INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL – CAMPUS PORTO ALEGRE
MESTRADO PROFISSIONAL EM INFORMÁTICA NA EDUCAÇÃO

THAÍS RAMOS VIEGAS

CONSPROG:
Uma proposta pedagógica para o ensino-aprendizagem de programação

Porto Alegre - RS
2017

THAÍS RAMOS VIEGAS

CONSPROG:
Uma proposta pedagógica para o ensino-aprendizagem de programação

Dissertação e produto de pesquisa apresentados ao Programa de Pós-graduação *Stricto Sensu* – Mestrado Profissional em Informática na Educação do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – Campus Porto Alegre, como parte dos requisitos para obtenção do título de Mestre em Informática na Educação.

Orientador: Fabio Yoshimitsu Okuyama.
Coorientadora: Sílvia de Castro Bertagnolli

Porto Alegre – RS
2017

CIP - Catalogação na Publicação

Viegas, Thaís Ramos

CONSPROG: UMA PROPOSTA PEDAGÓGICA PARA O ENSINO APRENDIZAGEM DE PROGRAMAÇÃO / Thaís Ramos Viegas. -- 2017.
225 f.
Orientador: Fabio Yoshimitsu Okuyama
Coorientadora: Silvia de Castro Bertagnolli

Dissertação (Mestrado) - Instituto Federal do Rio Grande do Sul, Campus Porto Alegre, Programa de Pós-Graduação em Informática na Educação, Porto Alegre, BR-RS,2017.

1. PROGRAMAÇÃO. 2. ENSINO-APRENDIZAGEM. 3. TAXONOMIA DE BLOOM. 4. REDES BAYESIANAS. I. Okuyama, Fabio Yoshimitsu, orient. II. Bertagnolli, Silvia de Castro, coorient. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica do Campus Porto Alegre / IFRS com dados fornecidos pelo(a) autor(a).

THAÍS RAMOS VIEGAS

CONSPROG:
Uma proposta pedagógica para o ensino-aprendizagem de programação

Dissertação e produto de pesquisa apresentados ao Programa de Pós-graduação *Stricto Sensu* – Mestrado Profissional em Informática na Educação do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – Campus Porto Alegre, como parte dos requisitos para obtenção do título de Mestre em Informática na Educação.

_____

Orientador:
Prof. Dr. Fabio Yoshimitsu Okuyama.
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – Campus Porto Alegre

_____

Coorientadora:
Profª. Drª. Silvia de Castro Bertagnolli
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – Campus Porto Alegre

Aprovada em 26 de setembro de 2017.
Local de defesa: Auditório 9º andar do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – Campus Porto Alegre, Rua Cel. Vicente, 281 – Centro – Porto Alegre. CEP: 90.030-041 – Rio Grande do Sul – Brasil.
Banca examinadora:

Profª Drª Adriana Justin Cerveira Kampff (PUCRS)
Profª Drª Josiane Caroline Soares Ramos (IFRS Campus Porto Alegre)
Profª Drª Márcia Häfele Islabão Franco (IFRS Campus Porto Alegre)

À minha filha por todo seu carinho e compreensão. Ao meu marido pela paciência, amor, companheirismo, e por sua capacidade de me trazer paz na correria de cada semestre. À minha família, pelo apoio e por sempre acreditarem em mim.

AGRADECIMENTOS

Primeiramente agradeço ao mundo espiritual por me permitir viver esse momento, aos amigos e familiares por entenderem minha ausência e mesmo assim emanarem boas vibrações para que eu seguisse adiante. Ao meu amado esposo Marcelo, que de forma especial e carinhosa me deu força e coragem, me auxiliando nos momentos de dificuldades. À minha filha Helena que iluminou de maneira especial os meus pensamentos. Aos meus pais e irmão, que com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida. Aos meus sogros e cunhada, sem os quais essa caminhada não seria possível.

Agradeço também aos amigos que fiz na “#tumaUnidaMPIE20152”, a todos os professores do curso, que me acompanharam e me auxiliaram sempre que preciso. Aos Professores Fabio Yoshimitsu Okuyama e Sílvia de Castro Bertagnolli, por seus ensinamentos, paciência e confiança ao longo das orientações.

RESUMO

A demanda por profissionais qualificados na área de Tecnologia da Informação (TI) cresce muito se comparada às taxas de formação dos cursos superiores e técnicos, restringindo o crescimento do setor. No ano de 2015 o índice de egressos nas graduações nesta área foi de 31%, um dos indicadores apontado para esse baixo coeficiente é a complexidade em aprender a programar, que resulta em altos índices de retenção em Disciplinas Introdutórias de Programação (DIP). Além disso, a quantidade elevada de estudantes nessas disciplinas faz com que o professor não consiga prestar acompanhamento individualizado. Neste sentido é necessário investigar meios que auxiliem professores e estudantes no processo de ensino-aprendizagem, podendo assim, reduzir o índice de retenção. Logo, a presente pesquisa apresenta uma proposta pedagógica para o ensino-aprendizagem de programação (ConsProg), desenvolvida a partir do levantamento bibliográfico sobre temas relacionados, seguida por uma pesquisa documental nos Projetos Pedagógicos dos Cursos Superiores em TI dos *Campi* do Instituto Federal de Educação Ciência e Tecnologia do Rio Grande do Sul (IFRS), principalmente em ementas das DIP. Os dados coletados nesses dois métodos de pesquisa foram necessários para conhecer o perfil dos cursos, bem como os objetivos dos mesmos nas DIP. Além disso, foi conduzido um estudo de caso em três etapas, com estudantes do curso Superior de Tecnologia em Sistemas para Internet (STSI) do IFRS - Campus Porto Alegre, nos anos 2016 e 2017. Na primeira etapa, realizou-se observações das aulas com o intuito de conhecer o perfil dos estudantes, bem como a metodologia de ensino adotada pelo professor da disciplina. Com base nestas observações, no estudo bibliográfico e na pesquisa documental, foi desenvolvida a ConsProg, fundamentada na Teoria Construtivista e na Taxonomia dos Objetivos Educacionais de Domínio Cognitivo. Essa proposta foi aplicada em uma nova turma, da mesma Instituição e curso, no segundo semestre de 2016. A aplicação da ConsProg, como foi proposta, auxilia o professor na organização da disciplina, sequenciamento dos conteúdos, planejamento da aula, elaboração de atividades que facilitem a identificação das dificuldades dos estudantes, análise das atividades de forma a verificar as possíveis carências no aprendizado e acompanhamento do desenvolvimento de aprendizagem da turma e de cada indivíduo. Com o intuito de sistematizar a análise das atividades, foi elaborada uma Rede Bayesiana (RB) que apresenta, de forma simplificada, informações sobre a aprendizagem de cada estudante de acordo com as atividades realizadas por eles. A RB agregada à proposta pedagógica foi utilizada na terceira etapa, ocorrida em 2017, mostrando-se adequada para o acompanhamento do desenvolvimento dos estudantes no processo de ensino-aprendizagem, apresentando uma redução de 16% no índice de retenção da disciplina.

Palavras-chave: Programação. Ensino-aprendizagem. Taxonomia de Bloom. Redes Bayesianas.

VIEGAS, Thaís Ramos. **CONSPROG:** Uma proposta pedagógica para o ensino-aprendizagem de programação. Porto Alegre, 2017. 225 f. Dissertação (Mestrado Profissional em Informática na Educação). Programa de Pós-Graduação e Pesquisa *Strictu Sensu* em Informática na Educação. Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS) Campus Porto Alegre, 2017.

ABSTRACT

The demand for qualified professional in area of Information Technology (IT) grows way beyond the conclusion rates the undergraduate and technical courses, for example, in 2015, only 31% of the students successfully graduated. One of the reasons for this low result is the complexity in learning the programing, which results in high reproof rate in Introductory Programming Disciplines (IPD). Besides that, the large number of students in the IPD can impair the teacher individual monitoring. In this sense, it's necessary investigate means to help teachers and students in the Teaching-learning process, being able reduce the reproof rates. Therefore, this research presents one pedagogical proposal of Teaching-learning of programming (ConsProg), developed from a literature review on related themes, followed by documentary research in curriculum of the graduation courses of IT in Federal Institute of Education, Science and Technology of Rio Grande do Sul (IFRS), mainly in IPD. The data collected in those two search methods was needed to know the profile of the course, and also objectives of IPD. In addition, was conducted one case study in three steps, with students of the graduation course of IT of the IFRS – Porto Alegre, in years 2016 and 2017. In first step, took place observation of the class to know the students profile and the methodology of teaching used in discipline by the teacher. With base in the observations, in the bibliographic study and in the documental research, was developed the ConsProg, based on the Constructivist Theory and on the Taxonomy of Educational Objectives: Cognitive Domain. That proposal was applied in a new class, in same one Institution and course, the second semester of 2016. The use of the ConsProg, as was proposed, assists the teacher in organization of discipline, planning of lesson, elaboration the activities for verification of the difficulties of students, and tracking the develop of learning the class or the student. For systematizing the analysis of activities, was elaborated one Bayesian Network (BN) that show information on the learning of each student, according to activities answered for them. The BN has been used in the third stage of the case study, in 2017, showing to be useful to follow-up of student's development in the process of Teaching-learning, contributing for reduction the reproof rates in IPD.

Keywords: Computer programming. Teaching-learning. Bloom's taxonomy. Bayesian Network.

VIEGAS, Thaís Ramos. **CONSPROG:** Uma proposta pedagógica para o ensino-aprendizagem de programação. Porto Alegre, 2017. 225 f. Dissertação (Mestrado Profissional em Informática na Educação). Programa de Pós-Graduação e Pesquisa *Strictu Sensu* em Informática na Educação. Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS) Campus Porto Alegre, 2017.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE GRÁFICOS

LISTA DE CÓDIGOS

LISTA DE APÊNDICE

LISTA DE SIGLAS E ACRÔNIMOS

ANSI	*American National Standards Institute*
AVA	Ambiente Virtual de Aprendizagem
CIP	Conceitos Introdutórios de Programação
CNBC	Catálogo Nacional de Cursos de Bacharelado
CNCST	Catálogo Nacional de Cursos Superiores de Tecnologia
DCN	Diretrizes Curriculares Nacionais
DIP	Disciplinas Introdutórias de Programação
ES	Ensino Superior
E1	Estudante 1
E2	Estudante 2
E3	Estudante 3
GTT	*Greedy Thick Thinning*
IFRS	Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul
INEP	Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira Legislação e Documentos
ISSO	*International Standars Organization*
LPI	Linguagem de Programação I
PPC	Projeto Pedagógico de Curso
RB	Rede Bayesiana
SBC	Sociedade Brasileira de Computação
STSI	Superior de Tecnologia em Sistemas para Internet
TC	Teoria Construtivista
TI	Tecnologia da Informação
TIC	Tecnologia da Informação e Comunicação
TOEDC	Taxonomia dos Objetivos Educacionais de Domínio Cognitivo
UFRGS	Universidade Federal do Rio Grande do Sul

SUMÁRIO

1 INTRODUÇÃO

A área de Tecnologia da Informação (TI) cresce constantemente, assim como a demanda por profissionais qualificados. Porém, dados apontam que a quantidade de egressos dos cursos superiores de TI não é suficiente para a demanda do mercado de trabalho (BRASSCOM, 2015), o que restringe o crescimento do setor. Apesar do aumento de interessados no ingresso em cursos superiores das áreas relacionadas a TI e TIC[1] ter crescido 70% de 2006 a 2013, apenas 31% dos estudantes se formam (INEP, 2015).

De acordo com a Brasscom (2015), a baixa taxa de concluintes pode estar relacionada à carga horária dos estudantes trabalhadores, ou até mesmo à falta da necessidade de terminar a graduação, uma vez que, segundo eles, os cursos técnicos garantem a entrada no mercado de trabalho de forma mais rápida. Outros autores como Lahtinen (2005) e Sevella et al. (2013), apontam que a alta taxa de evasão nos cursos superiores de TI está ligada a fatores educacionais, como o nível de complexidade em aprender a programar.

Lahtinen (2005) aponta que: o uso de IDE[2], a compreensão da estrutura da linguagem de programação e de sua sintaxe, são algumas das dificuldades que os estudantes encontram na aprendizagem de técnicas de programação. Já Sevella et al. (2013) apresenta algumas barreiras que os estudantes, novatos em programação, enfrentam como: a construção do código, encontrar erros no código, a compreensão dos erros durante a execução do programa, entre outros fatores. Em concordância com esses autores, acrescenta que o nível de abstração, bem como a necessidade de domínio sobre outras áreas, podem ser aspectos que dificultam o processo de ensino-aprendizagem (VIEGAS, 2015).

Neste sentido, tem-se o delineamento do problema de pesquisa: como identificar e minimizar as dificuldades de aprendizagem que os estudantes enfrentam nas DIP? Acredita-se que a elaboração de uma abordagem pedagógica, que tenha como cerne o estudante, que possibilite a verificação da necessidade de intervenções pedagógicas, que auxilie na identificação dos conteúdos a serem reforçados, e que contribua com o docente no acompanhamento do desenvolvimento de aprendizagem, possa auxiliar no processo educativo.

---

[1] Tecnologia da Informação e Comunicação

[2] *Integrated Development Environment* - Ambiente de Desenvolvimento Integrado

Com este cenário, a presente dissertação apresenta uma proposta pedagógica, baseada na Teoria Construtivista (TC) de Jean William Fritz Piaget e na Taxonomia de Objetivos Educacionais de Domínio Cognitivo (TOEDC) de Benjamin Seymour Bloom et al., para a prática docente em disciplinas relacionadas aos Conceitos Introdutórios de Programação[3] (CIP). Para isso, realizou-se um levantamento bibliográfico sobre diversos temas: ensino-aprendizagem, taxonomia de objetivos educacionais, trabalhos correlatos e conceitos introdutórios do ensino de programação.

Com o intuito de a vincular a proposta pedagógica com os elementos e objetivos da disciplina de programação, foi realizado uma pesquisa em documentos do Instituto Federal do Rio Grande do Sul (IFRS), sobre os cursos de TI ofertados na rede, com enfoque nos cursos superiores. Foram analisados documentos como: Projeto Pedagógico de Curso (matriz curricular, ementas das disciplinas do primeiro semestre do curso relacionadas à programação), chamadas da disciplina de Linguagem de Programação I do curso SSI do campus Porto Alegre, leis e decretos sobre educação, entre outros. A análise desse levantamento será abordada detalhadamente no capítulo 6, seção 6.1 - Cursos de TI e a disciplina de Linguagem de Programação I.

Após a pesquisa bibliográfica e documental, para criação e validação da proposta pedagógica, foi realizado um estudo de caso dividido em três etapas, nas turmas do primeiro semestre do curso Superior de Tecnologia em Sistemas para Internet (STSI) do IFRS campus Porto Alegre. Assim sendo, a primeira etapa ocorreu em 2016/1, com intuito de acompanhar as aulas e investigar como o trabalho docente, nesse universo, proporciona ensino-aprendizagem aos estudantes de programação. Uma vez que a proposta pedagógica tem o objetivo de auxiliar o professor, levando em conta as necessidades dos estudantes, foi aplicado um questionário, a fim de conhecer a visão dos estudantes sobre a disciplina de programação e a metodologia nela aplicada.

Por meio da análise qualitativa e quantitativa dos dados oriundos dessa primeira etapa, juntamente com a pesquisa bibliográfica e documental, foi definida a primeira versão da proposta pedagógica, denominada ConsProg. Essa proposta visa auxiliar o professor no desenvolvimento de materiais para suas aulas, que vão desde

[3] Neste trabalho, considera-se conceitos introdutórios de programação os seguintes conteúdos: variáveis, entrada e saída de dados, operadores aritméticos, operadores relacionais, operadores lógicos, estrutura de controle e estrutura de decisão.

a forma no qual ele pode ministrá-las, até o acompanhamento do aprendizado dos estudantes por meio de atividades.

Ao verificar as resoluções dessas atividades, por meio da proposta, o professor poderá adaptar suas aulas de acordo com as necessidades da turma, bem como prestar auxílio àqueles estudantes que apresentam maior dificuldade no desenvolvimento de aprendizagem dos CIP. Uma vez que ocorra adequação das aulas conforme a demanda da turma, ter-se-á uma disciplina que preza a qualidade do aprendizado dos estudantes, resultando em profissionais mais qualificados para o mercado de trabalho.

Na segunda etapa do estudo de caso, que ocorreu em 2016/2, foi aplicada a proposta pedagógica (ConsProg) em uma nova turma do primeiro semestre do curso SSI, no Campus Porto Alegre. A disciplina foi ministrada pelo mesmo professor da intervenção anterior, e nesse processo realizou-se o acompanhamento das aulas com o intuito de investigar a prática docente fazendo o uso da proposta pedagógica desenvolvida. Além disso, geraram-se dados por meio de um questionário, coletou-se dados no Ambiente Virtual de Aprendizagem (AVA) Moodle sobre as resoluções das atividades e das avaliações, a fim de verificar se houve mudanças no desenvolvimento da aprendizagem dos estudantes, em comparação com a primeira etapa.

Assim sendo, através do uso da ConsProg foi possível acompanhar o desenvolvimento de aprendizagem dos estudantes, uma vez que os materiais desenvolvidos permitem essa identificação de forma mais clara. Porém, por tratar-se de uma disciplina de primeiro semestre, as turmas possuem um número bastante elevado de estudantes, tornando a análise individual trabalhosa, demandando tempo de docente. Isso ocorre, pois, o professor deve verificar a relação entre os conteúdos, em cada atividade, para compreender em qual conceito o estudante demonstra maior dificuldade.

Essa relação deve ser analisada, uma vez que cada conteúdo estará relacionado com outros, podendo assim uma dificuldade no conteúdo “x” prejudicar o aprendizado de outro conteúdo. Desta forma, visando auxiliar o docente no processo de acompanhamento de turmas com número elevado de estudantes matriculados, realizou-se um mapeamento da relação entre CIP, através dos dados gerados na segunda etapa do estudo de caso.

Em 2017/1 realizou-se a terceira e última etapa do estudo de caso que deu origem à RB, que contempla a ConsProg. Para a construção desta RB, foram

utilizados os dados coletados dos questionários realizados pelos estudantes, e também um sistema para a estruturação e aprendizado de RB. Contou-se também com a análise da especialista da área de educação, bem como o da área de ciência da computação.

O uso da ConsProg, que contou com a RB nesta terceira etapa, possibilitou ao professor a identificação das dificuldades dos estudantes de forma mais clara, objetiva e rápida. A proposta pedagógica proporcionou ao docente mais segurança ao finalizar os conceitos dos discentes, visto que as informações apresentadas na RB condiziam com o que era apresentado nas atividades avaliativas. Além disso, durante todo o semestre o docente pôde identificar as dificuldades apresentadas pela turma, e assim retomar os conceitos necessário para suprir as necessidades dos estudantes.

Neste sentido, acredita-se que a ConsProg contribui no processo educativo, podendo proporcionar aos estudantes e professores uma experiência diferenciada e mais efetiva no ensino-aprendizagem de programação. Na seção a seguir apresenta-se os objetivos geral e específicos.

1.1 OBJETIVOS

O presente trabalho tem como objetivo principal criar uma abordagem pedagógica que possibilite um maior acompanhamento do processo de ensino-aprendizagem dos CIP. Além disso, a pesquisa tem o propósito de:

- Mapear correlações entre os CIP, mostrando as dependências entre os conteúdos, para a identificação do desenvolvimento da aprendizagem;
- Construir conectivos entre a teoria construtivista de Pieget, a taxonomia dos objetivos educacionais de domínio cognitivo de Bloom et al. e aos conteúdos introdutórios de programação;
- Construir uma Rede Bayesiana com o intuito de sistematizar a proposição do desenvolvimento de aprendizagem.

De acordo com Creswell (2010) quanto maior for o público atendido em um projeto de pesquisa, maior será a importância desse estudo. Nesse sentido, na próxima seção apresenta-se a justificativa, no qual mostra o quão considerável é a

Dissertação, uma vez que pretende auxiliar professores e estudantes no processo de ensino-aprendizagem de cursos superiores em Tecnologia da Informação.

1.2 JUSTIFICATIVA

No Brasil, de acordo com o Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP, 2015), existem em torno de 32.049 cursos de graduação, sendo 3.203 dentro da área geral de conhecimento ciências, matemática e computação. O MEC[4] apresenta os cursos de computação divididos em três categorias: bacharelados, licenciaturas e tecnólogos. No Catálogo Nacional de Cursos Superiores de Tecnologia são apresentados 14 tipos de cursos em TI (MEC, 2016), e nas Diretrizes Curriculares Nacionais para os cursos de graduação em Computação encontram-se quatro tipos de cursos bacharéis e um curso de licenciatura (MEC, 2012).

Existem, portanto, 19 tipos de cursos ligados à área de computação no Brasil. Em 2013, conforme divulgado pela Brasscom (2015), foi registrado em torno de 130.340 ingressantes nestes cursos, formando-se apenas 38.564. Levando em conta esses números, é observado que existe um índice em torno de 70% de retenção e/ou evasão, podendo com isso faltar profissionais qualificados para o mercado de trabalho.

Uma das possibilidades para tal fenômeno poderia ser a dificuldade no aprendizado de programação, de acordo com Lahtinen (2005) que apresenta uma análise das dificuldades dos estudantes e professores no ensino-aprendizagem de programação, aponta que a compreensão de conceitos abstratos pode ser um dos grandes problemas encontrados pelos estudantes na aprendizagem de programação. Um estudo mais recente, realizado por Souza et al. (2016), aponta três fatores como principais problemas no ensino-aprendizagem de programação: aprender os conceitos de programação, aplicar os conceitos de programação e a falta de motivação em realizar as atividades propostas. Com um olhar direcionado para o docente, Menezes e Nobre (2002) acreditam que o número elevado de estudantes

[4] Ministério da Educação

não permite um acompanhamento individualizado, dificultando que o professor promova um aprendizado mais homogêneo na turma.

Muitos são os estudos relacionados à dificuldade no ensino-aprendizagem de programação, porém Papert (1985), criador da linguagem LOGO, fez com que crianças de quatro anos conseguissem comandar uma tartaruga mecânica através de comandos. O LOGO também possui uma versão virtual (IDE), no qual o estudante consegue visualizar a tartaruga realizando os movimentos por ele programados, agregando o abstrato (linhas de comando) ao concreto (movimento da tartaruga). Neste sentido é necessário compreender o que está faltando no processo de ensino-aprendizagem, visto que segundo a visão de Papert (1985), é possível aprender a programar em qualquer idade.

De acordo com Deitel (2011, p.7), “os computadores processam dados sob o controle de conjuntos de instruções chamados programas de computador”, esses programas são desenvolvidos em uma linguagem específica. De uma forma mais simplificada, programar significa, nada mais, nada menos que comunicar-se com o computador, numa linguagem que tanto ele quanto o homem podem entender (Papert, 1985). Porém, observa-se que muitos estudantes enfrentam dificuldades em comunicar-se com o computador, e em compreender os aspectos da programação, como capacidade para planejar, desenvolver, testar e depurar os programas por eles criados (Boulay, 1989).

Desta forma, a disciplina de programação é desafiadora, uma vez que requer diversas habilidades e conhecimentos, com isso, cabe investigar o processo de ensino-aprendizagem sob a perspectiva da prática docente. Assim sendo, a presente proposta de dissertação mostra-se significativa, pois tem como objetivo principal criar uma abordagem pedagógica que possibilite um maior acompanhamento do processo de ensino-aprendizagem dos CIP. A próxima seção apresenta como o presente trabalho foi organizado.

1.3 ESTRUTURA DA DISSERTAÇÃO

O presente trabalho encontra-se organizado em capítulos, seções e subseções. No segundo capítulo é possível conhecer a história do IFRS campus Porto Alegre, bem como a criação do curso STSI e a estrutura da disciplina de programação. Esse

capítulo tem o intuito de contextualizar o objeto de estudo (disciplina de LPI), e também as abordagens pedagógicas adotadas pelo campus Porto Alegre em seus cursos, em especial no STSI.

O capítulo três possui três seções referentes à metodologia de pesquisa utilizada, a população dos indivíduos participantes e como foram feitas a geração e coleta de dados. No quarto capítulo apresentam-se trabalhos correlatos sobre ensino-aprendizagem nos cursos superiores de TI. Os dados utilizados para este, são artigos publicados nos anos de 2014, 2015 e 2016, em eventos, revistas e workshops da Sociedade Brasileira de Computação (SBC).

Já no quinto capítulo encontra-se a fundamentação teórica, possuindo cinco seções, sendo a primeira responsável por apresentar a TC utilizada para embasar a proposta pedagógica. A segunda seção traz a TOEDC, utilizada para criação e avaliação das atividades. Na terceira seção é possível conhecer os conceitos de programação utilizados na presente pesquisa. Seguindo, tem-se a seção quatro, apontando quais são os desafios e dificuldades encontrados pelos estudantes ao aprenderem os CIP. Na última seção deste capítulo, é apresentado conceitos sobre Redes Bayesianas, utilizadas na sistematização da análise das atividades.

O capítulo seis traz a análise dos dados e encontra-se subdividido em três seções: cursos de TI e a disciplina de Linguagem e Programação I; estudo de caso; e Processo de criação da Rede Bayesiana. A primeira seção traz o levantamento dos cursos de TI no IFRS, bem como as disciplinas relacionadas com o ensino de programação no primeiro semestre dos cursos e os conteúdos ministrados nas mesmas. Na segunda seção, encontram-se as análises das três etapas do estudo de caso, organizadas em três subseções (primeira, segunda e terceira etapa do estudo de caso).

Na sequência encontra-se o capítulo sete, possuindo quatro seções que explicam passo a passo como utilizar a ConsProg, trazendo exemplos utilizados na disciplina de LPI durante a segunda e terceira etapa do estudo de caso. Por último, tem-se o capítulo oito que apresenta as considerações finais.

2 CONTEXTUALIZAÇÃO

A educação por si só é entendida pela sociedade como papel importante na formação do ser humano. Mas é necessário compreender que esta educação advém de um contexto histórico, já que as “questões de educação são engendradas nas relações que os homens estabelecem ao produzir sua existência” (ARANHA, 2006, p.19). Deste modo, toda pesquisa relacionada à educação deve apresentar o contexto histórico da instituição em que se realiza a investigação, pois esta tem relação direta com os resultados obtidos.

Além disso, é possível através desse contexto compreender as práticas pedagógicas adotadas pela instituição. Nesse sentido, Reis Filho (1981) afirma que através do conhecimento histórico é possível a reflexão filosófica quanto ao conteúdo da realidade sobre o que se pensa, podendo assim descobrir as diretrizes das ações pedagógicas. Através da contextualização sobre o surgimento das escolas técnicas e dos Campi do IFRS, foi possível ter um delineamento do contexto institucional do campus Porto Alegre e as ações pedagógicas adotadas por ele, dando ênfase no curso STSI.

Na sequência tem-se uma seção sobre a criação do campus Porto Alegre, e esta possui duas subseções: O curso Superior de tecnologia em Sistemas para Internet; A disciplina de programação no curso STSI. A primeira subseção trata sobre o curso STSI, desde seu surgimento até suas modificações no PPC. E a segunda subseção fala sobre a disciplina de programação no curso STSI do campus Porto Alegre, apresentando sua ementa, carga horária e objetivos.

2.1 IFRS E A CRIAÇÃO DO CAMPUS PORTO ALEGRE

Em 1906 foi fundada, em Porto Alegre, a Escola de Comércio Porto Alegre, que futuramente deu origem à Universidade Federal do Rio Grande do Sul (UFRGS) por meio de seu curso geral. Com o decreto nº 7.566 de 23 de setembro de 1909, assinado por Nilo Peçanha, cursos profissionalizantes são criados com o intuito de qualificar a mão de obra (BRASIL, 1909). Nesta época a Escola Técnica de Comércio Porto Alegre, que ficava junto à Faculdade Livre de Direito de Porto Alegre, tinha, para Moll e colaboradores (2010), a metodologia de ensino essencialmente prática, atendendo

o decreto do Presidente da República que era “formar operários e contramestres, ministrando-se o ensino prático e os conhecimentos técnicos necessários aos menores que pretenderem aprender um ofício” (BRASIL, 1909, art. 2º).

De acordo com Moll e colaboradores (2010), em 1934 foi criada a Universidade de Porto Alegre, integrando a ela a Escola de Comércio Porto Alegre e a Faculdade Livre de Direito de Porto Alegre. Em 1942, de acordo com Kuenzer (2001), os cursos técnicos começam a ser considerados cursos de nível médio, porém alguns deles não davam o direito do estudante fazer o Ensino Superior (ES).

No ano de 1950 a Faculdade de Economia e Administração uniu-se a Escola Técnica de Comércio, passando a fazer parte da Universidade Federal do Rio Grande do Sul (MOLL, 2010). Nesta época, inicia uma nova fase, pois a “Universidade passou a ser administrada pelo Governo Federal [...]. A Faculdade de Economia e Administração e, respectivamente, a Escola de Comércio, agora denominada Escola Técnica de Comércio, passaram a integrar o sistema federal” (IFRS, 2011). Novos cursos técnicos foram sendo criados ao longo dos anos (1954 a 1963): Técnico de Administração, Técnico de Secretariado, Administração de Pessoal, Organização de Fundos, Psicotécnica e Técnica Orçamentária. Foi somente na década de 60 que a Escola Técnica passou a ter direção própria, e com isso novos cursos foram implantados com o passar dos anos: Operador de Computador, Transações Imobiliárias, Comercialização e Mercadologia, Segurança do Trabalho, Suplementação em Contabilidade e Suplementação em Transações Imobiliárias (IFRS, 2013).

Em 1996 cursos como Técnico em Biotecnologia e Técnico em Química, assim como os cursos Pós-Técnicos de Controle e Monitoramento Ambiental, Redes de Computadores e Suplementação em Processamento de Dados, iniciam na Escola Técnica de Comércio da UFRGS, que passou a se chamar de Escola Técnica da UFRGS (IFRS, 2016).

Uma vez que as instituições educacionais estão diretamente ligadas à política, os cursos técnicos sofreram diversas modificações ao longo dos anos, nesse sentido, “a Escola Técnica passa a ministrar, no ano de 1999, somente cursos de educação profissional, tendo como pré-requisito para ingresso a conclusão do ensino médio” (IFRS, 2016, p.9). Na busca de medidas para “ampliação do acesso à educação e da permanência e aprendizagem nos sistemas de ensino” (PACHECO, 2011, p.6), em

2008 o Presidente da República decreta e sanciona a Lei nº 11.892, de 29 de dezembro de 2008 que cria os Institutos Federais de Educação, Ciência e Tecnologia.

Por meio desta Lei (nº 11.892), de acordo com IFRS (2013, p. 6), “o Campus Porto Alegre do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS) nasce da desvinculação da Escola Técnica da Universidade Federal do Rio Grande do Sul (UFRGS)”. Atualmente, de acordo com dados do MEC (2016), a rede conta com 38 Institutos Federais e 644 Campi. Segundo o art. 6º da Lei 11.892, os IF têm como características a oferta de educação profissional e tecnológica desenvolvendo-a como processo educativo e investigativo, promovendo integração e verticalização da educação desde o ensino básico até o superior, constituindo-se como centros de excelência na oferta do ensino, entre outras (BRASIL, 2008).

Com isso, de acordo com Pacheco (2011, p.13), “o governo federal, através do Ministério da Educação, criou um modelo institucional absolutamente inovador em termos de proposta político-pedagógica”, tendo “suas bases em um conceito de educação profissional e tecnológica sem similar em nenhum outro país”. Logo, em função da reestruturação, a antiga Escola Técnica da UFRGS, atual IFRS Campus Porto Alegre, “passou a ter uma nova estrutura administrativa e pedagógica, necessária para atender as demandas que surgem com a criação de novos cursos técnicos e superiores” (IFRS, 2011).

Até o presente momento (primeiro semestre de 2017), o Campus Porto Alegre conta com 15 cursos técnicos, cinco cursos superiores (três tecnólogos e duas licenciaturas), um PROEJA, dois cursos *Latu Sensu*, um curso *Stricto Sensu*, e também cursos de extensão. Um dos tecnólogos é o de Sistemas para Internet, objeto desta pesquisa, no qual apresentar-se-á na próxima seção.

2.2 O curso Superior de Tecnologia em Sistemas para Internet

Com o intuito de formar profissionais que desenvolvam aplicações computacionais para a Internet, o curso foi implantado em 2010/2. Nessa época o curso era ofertado uma vez ao ano, no turno da manhã com 36 vagas para estudantes ingressantes. No ano de 2014 o curso passou a ser ofertado duas vezes ao ano, com 72 vagas para ingressos, 36 para o turno da manhã e 36 para o turno da noite, sendo

que a periodicidade da oferta é feita em turnos alternados: semestre par durante a manhã e semestre ímpar durante a noite.

O curso foi criado de acordo com o Catálogo Nacional de Cursos Superiores de Tecnologia (MEC, 2016), possuindo 2256 horas divididas em seis semestres. Assim sendo, o curso duração de três anos e de acordo com a Resolução 046, de 08 de maio de 2015[5], o estudante deve concluir o curso em no máximo seis anos ou doze semestres a contar de seu ingresso. Para isso, a estrutura curricular encontra-se organizada em 12 áreas, com maior concentração em desenvolvimento e engenharia de software, o que torna os CIP extremamente importantes, por serem a base dessa área.

Quadro 1 - Disciplinas ofertadas no curso STSI

1º Semestre	2º Semestre	3º Semestre	4º Semestre	5º Semestre	6º Semestre
Lógica de programação	Estrutura de dados I	Programação para web I	Programação web II	Empreendedorismo	Governança de TI
Inglês técnico	Banco de dados I	Engenharia de software II	Segurança e auditoria	Programação para web III	Validação e Verificação de Sistemas
Fundamentos da computação	Linguagem de programação II	Banco de dados II	Sistemas operacionais	Gestão de projetos	Probabilidade e Estatística
Construção de páginas Web I	Engenharia de software I	Web design	Engenharia de software III	Leitura e produção textual	Informática e Sociedade
Linguagem de programação I	Construção de páginas Web II	Redes de computadores I	Redes de computadores II	Metodologia da pesquisa	Trabalho de Conclusão (TC)
Interação humano computador				Optativa I	Técnicas de Apresentação
				Optativa II	Optativa III
					Optativa IV

Fonte: Adaptado de IFRS (2016, p. 27-30)

No Quadro 1 é possível visualizar as disciplinas ofertadas no curso STSI, organizadas por semestre, as células destacadas (em cinza) referem-se às disciplinas que utilizam os conhecimentos introdutórios de programação. Analisando o Quadro 1,

[5] Resolução referente aprovação da Organização Didática pelo Conselho Superior do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul. *Cf.* <http://ifrs.edu.br/wp-content/uploads/2017/07/20170301747344830d_versao_out_2016_dir_dev_estud_2_a.pdf > Acesso em 04 mar. 2017.

observa-se ainda que pelo menos 41% das disciplinas ofertadas no curso faz o uso dos conteúdos aprendidos em Linguagem de Programação I.

O PPC também apresenta os objetivos de cada ano do curso, no qual o primeiro visa à formação básica para o desenvolvimento de pequenas aplicações Web. Já no segundo ano a intenção é de que o estudante esteja capacitado para o desenvolvimento de sistemas Web baseados em camadas e orientados a objeto. No terceiro e último ano do curso trabalha-se questões de segurança e a utilização de *frameworks*. Para melhor compreensão da estrutura e organização da disciplina Linguagem de Programação I, na próxima seção esta será apresentada de forma mais detalhada.

2.3 A DISCIPLINA DE LINGUAGEM DE PROGRAMAÇÃO I NO CURSO STSI

No curso STSI, do IFRS campus Porto Alegre, a disciplina Linguagem de Programação I é ofertada no primeiro semestre, possuindo 80 horas aula (66 horas). De acordo com o PPC do curso, a disciplina é presencial, e deve “apresentar o paradigma da programação estruturada, com uso da Linguagem C – Padrão ANSI (*American National Standasds Institute*), com o objetivo de desenvolver o raciocínio na elaboração de soluções a problemas de solução algorítmica” (IFRS, 2013, p.24).

Ministrada pelo mesmo professor desde 2011/2, a ementa não apresenta mudanças significantes na segunda versão do o PPC (IFRS, 2013), permanecendo com a mesma linguagem de programação e conteúdos programáticos. Neste sentido, a disciplina sofreu mudanças apenas em sua carga horária, de 72 para 80 horas aula. Os conteúdos no qual o professor deve trabalhar no semestre são:

> fundamentos da construção de programas utilizando linguagem C ANSI. Conceitos de variáveis, variáveis homogêneas (vetores e matrizes) e variáveis heterogêneas (registros). Operadores e expressões matemáticas e lógicas. Estruturas de controle de programação. Funções, procedimentos, variáveis locais e globais, passagem de parâmetros por valor e por referência e tratamento de arquivos. (IFRS, 2013, p.24)

No final do semestre, os estudantes recebem um conceito: A (Conceito Ótimo), B (Conceito Bom), C (Conceito Regular), D (Conceito Insatisfatório) ou E (Falta de Frequência), este conceito é formado pelas atividades avaliativas da disciplina. No caso da disciplina de Linguagem de Programação, são realizados provas e trabalhos práticos. Logo, para que o estudante seja considerado aprovado, este deve ter

conceito "A", "B" ou "C" e poderá matricular-se nas disciplinas seguintes. Caso o estudante atinja conceitos "D" ou "E", estes estão reprovados, e só podem efetuar matrícula nas disciplinas que não exigem a aprovação em Linguagem de Programação I como requisito.

Essa disciplina é ofertada de 2010 a 2013 anualmente e semestralmente desde 2014, as turmas possuem em média 40 estudantes, sendo que 36 destes são ingressantes e os demais repetentes. Em um levantamento realizado por Wolff et al. (2016) a respeito da situação dos estudantes ingressantes no curso STSI, aponta que os estudantes que ingressam no turno da manhã por vestibular têm maior probabilidade de evadir ou não concluir o curso. Acredita-se que este fato ocorra uma vez que os estudantes possam ter conseguido emprego e assim não consigam estudar no turno da manhã, porém estes mesmos não renovam suas matrículas em turno diferenciado (WOLF et al., 2016). Na sequência ter-se-á o capítulo 3 que apresenta os procedimentos metodológicos utilizados por esta pesquisa.

3 PROCEDIMENTOS METODOLÓGICOS

Um projeto de pesquisa, segundo Creswell (2010), é a forma no qual se planeja e se procede na pesquisa, que pode ser desde suposições até métodos detalhados. Indiferente da área que se estude, é importante determinar a metodologia, pois esta guia o pesquisador e o auxilia na forma em que chegará aos resultados. Nesse sentido, o presente capítulo possui três seções que visam apresentar os métodos de pesquisa utilizados, quais são os indivíduos pesquisados, como foi feita a geração e coleta de dados, e também como se realizou a análise desses dados.

3.1 METODOLOGIA DE PESQUISA

De natureza aplicada e com objetivos exploratórios, a pesquisa iniciou através do levantamento bibliográfico sobre os diversos temas relacionados ao projeto. Para Fonseca (2002), o levantamento bibliográfico é uma pesquisa feita em documentos publicados como: livros, revistas, artigos, entre outros; que tem o intuito de embasar o pesquisador em sua análise. Nesse contexto, buscaram-se trabalhos correlatos sobre o ensino-aprendizagem de programação, em anais de eventos da SBC[6], relacionados à educação. Além disso, buscou-se compreender como ocorre a construção do conhecimento, por meio da TC e da TOEDC. Uma vez que o objetivo principal foi a criação de uma abordagem pedagógica que possibilite um maior acompanhamento do processo de ensino-aprendizagem dos CIP, investigou-se também sobre como desenvolver atividades que auxiliassem na identificação do desenvolvimento cognitivo dos estudantes sobre os CIP.

Assim sendo, utilizou-se a teoria de Bloom et al. a respeito dos objetivos educacionais de domínio cognitivo. Para Bloom et al. (1972), é possível que a TOEDC sirva de apoio ao professor, tanto na construção de suas atividades, quanto na análise das respostas dos estudantes. Por meio dessas respostas, o professor consegue verificar qual dificuldade o estudante possui.

Levando em conta que a pesquisa trata do ensino-aprendizagem de programação, nos levantamentos bibliográficos também se incluíram estudos a respeito de CIP. Visto que a pesquisa está sendo aplicada no IFRS campus Porto

[6] Sociedade Brasileira de Computação

Alegre, e a linguagem de programação escolhida para a criação dos materiais foi a mesma já utilizada na disciplina: Linguagem C padrão ANSI.

Adicionalmente, utilizou-se uma pesquisa documental, que de acordo com Fonseca (2002, p.32), “a pesquisa documental recorre a fontes mais diversificadas e dispersas, sem tratamento analítico, tais como: tabelas estatísticas, jornais, revistas, relatórios, documentos oficiais, [...]". Nesse sentido, buscou-se em ementas das disciplinas de programação, os PPC superiores em TI dos IFRS, informações para adequar a proposta pedagógica às mesmas. Ainda com o intuito de construir uma proposta pedagógica propícia às necessidades dos estudantes e professores dos cursos superiores de TI do IFRS, foi realizado estudo de caso no campus Porto Alegre.

De acordo com Creswell (2010, p. 86), um estudo de caso é uma metodologia “na qual o investigador explora [...] múltiplos sistemas delimitados (casos) ao longo do tempo, por meio da coleta de dados detalhada em profundidade envolvendo múltiplas fontes de informação (p. ex., observações, entrevistas [...]) e relata uma descrição do caso [...]”. Esse estudo foi dividido em três etapas: Observação sem o uso da proposta pedagógica – primeira etapa do estudo de caso; Observação com a proposta sendo utilizada – segunda etapa do estudo de caso; Relato do docente sobre a utilização da proposta pedagógica – terceira etapa do estudo de caso. Todos eles foram realizados no curso STSI do IFRS campus Porto Alegre, na disciplina de Linguagem de Programação I.

A primeira etapa foi no semestre 2016/1, no qual realizou-se observações durante as aulas, um questionário fechado (Apêndice A) e coleta de dados no AVA Moodle. O objetivo desta etapa do estudo de caso foi compreender a relação do estudante com a disciplina de LPI, as dificuldades por eles encontradas na disciplina, bem como a metodologia utilizada pelo professor. Durante todo esse processo, a proposta pedagógica foi sendo construída, bem como as atividades que foram aplicadas na segunda etapa do estudo de caso.

Em 2016/2 ocorreu a segunda etapa, no qual também se realizou o acompanhamento das aulas, porém agora com o intuito de verificar se a proposta pedagógica proporcionou aos estudantes um desenvolvimento de aprendizado mais efetivo. Para uma análise quantitativa, foi feito um novo questionário fechado (Apêndice B) e coleta de dados no AVA Moodle, para ilustrar a análise qualitativa sobre as observações.

Com o intuito de sistematizar as análises das atividades dos estudantes, construiu-se uma RB elaborada a partir dos dados da primeira e segunda etapas estudo de caso. Essa RB foi aplicada em um terceiro estudo de caso (2017/1), que teve o propósito de verificar se a proposta pedagógica em conjunto com a RB pode contribuir com o professor no acompanhamento de seus estudantes, uma vez que as turmas são numerosas. Assim sendo, o professor da disciplina fez o uso da ConsProg com a RB e posteriormente entregará um relatório sobre essa sistematização. Na próxima seção, é possível verificar quem são indivíduos da pesquisa, quais foram os critérios de inclusão e exclusão dos mesmos.

3.2 POPULAÇÃO E AMOSTRA

Como já foi relatado na seção anterior, o presente projeto de pesquisa está sendo aplicado na disciplina de LPI, do curso STSI, do IFRS campus Porto Alegre (no Apêndice D encontra-se o modelo do termo de aceite institucional entregue ao diretor do campus). A disciplina é realizada no primeiro semestre do ano no turno da noite, e o segundo semestre no turno da manhã. Até o momento participaram da pesquisa em torno de 90 indivíduos, os quais estão divididos em três amostras enumeradas de 1 a 3 para facilitar compreensão.

A primeira corresponde a etapa um do estudo de caso, aplicada no semestre 2016/1, contando com 43 estudantes. No decorrer do mesmo, alguns estudantes não compareceram às aulas, evadindo a disciplina. Sendo assim, no último dia do semestre, foram contabilizados 20 estudantes que não evadiram.

Já a segunda amostra, trata-se da etapa dois do estudo de caso, no qual corresponde ao semestre 2016/2, iniciando com 46 estudantes. Assim como na amostra anterior, muitos foram evadindo a disciplina, restando 24 no final do semestre.

A terceira etapa estudo de caso realizado em 2017/1 iniciou com 43 estudantes na turma, restando 27 no final. Diferente dos outros estudos de caso, o sujeito participantes desta foi o professor da disciplina, o qual entregou um relatório com o parecer da utilização da ConsProg, e disponibilizou os dados do AVA Moodle para análise. Na sequência será apresentada a forma na qual foram feitas a geração e coleta de dados.

3.3 GERAÇÃO E COLETA DE DADOS

Durante a pesquisa foram gerados e coletados dados oriundos do levantamento bibliográfico, da pesquisa documental e do estudo de caso. Nesse sentido, a presente seção possui três subseções que apresentam como foram realizadas a geração e coleta dos dados utilizados na pesquisa. A primeira subseção fala sobre a pesquisa bibliográfica, a segunda sobre a pesquisa documental, e a terceira traz informações sobre as três etapas do estudo de caso realizados nos semestres: 2016/1; 2016/2; 2017/1.

3.3.1 Pesquisa bibliográfica

De acordo com Fonseca (2002, p. 31) “qualquer trabalho científico inicia-se com uma pesquisa bibliográfica, que permite ao pesquisador conhecer o que já se estudou sobre o assunto”. Neste sentido, através do levantamento bibliográfico foi possível conhecer o estado da arte, os desafios e dificuldades no ensino-aprendizagem de programação e a contextualização do objeto de pesquisa. Além disso, construiu-se a fundamentação teórica utilizando as teorias construtivistas de Piaget, a teoria da taxonomia dos objetivos educacionais de domínio cognitivo do Bloom et al. (1972), uma vez que se acredita que haja relação entre essas teorias e o ensino-aprendizagem de programação.

“Procurando referências teóricas publicadas com o objetivo de recolher informações ou conhecimentos prévios sobre o problema a respeito do qual se procura a resposta” (FONSECA, 2002, p. 31-32), investigou-se por trabalhos correlatos. Este foi escrito por meio da seleção de artigos em repositórios de eventos, congressos e workshops da Sociedade Brasileira de Computação de 2014 a 2016, que fossem relacionados ao ensino. A explicação de como foram selecionados os artigos e análise dos mesmos pode ser apreciado no capítulo 4 - Trabalhos correlatos, no qual buscou-se trabalhos similares ao da presente pesquisa. Esse levantamento completo está organizado em: Apêndice E, Apêndice F, Apêndice G, Apêndice H, Apêndice I, Apêndice J, Apêndice K, Apêndice L, e Apêndice M.

O conhecimento a respeito das dificuldades e desafios encontrados pelos estudantes em DIP tornou-se necessário, visto que a pesquisa teve o intuito de criar

uma proposta pedagógica que permita ao docente identificar esses obstáculos de forma mais rápida e clara. Logo, realizou-se a coleta dessas informações em artigos científicos e revistas da área de ensino de programação. O resultado pode ser verificado na seção 5.4 - Ensino-aprendizagem de programação: desafios e dificuldades.

Uma vez que a pesquisa é de natureza aplicada e a mesma ocorre em uma Instituição de Ensino, conhecer o contexto da mesma se faz importante para compreender as metodologias utilizadas por ela. Com isso, buscaram-se livros e artigos que apresentassem a história do IFRS campus Porto Alegre, desde quando a mesma era Escola Técnica de Comércio Porto Alegre, sendo que a contextualização pôde ser verificada no Capítulo 2.

Com o intuito de fundamentar a proposta metodológica, investigaram-se autores que trabalhassem com o ensino de forma “cumulativa”, uma vez que os conteúdos de programação se interligam, sendo sempre necessário o conhecimento de conceitos anteriores para compreender o atual. Sendo assim, optou-se pela TC de Piaget (seção 5.1) que embasou sobre a construção do conhecimento. Porém, uma disciplina não trabalha apenas com conceitos, requer o acompanhamento dos estudantes através de atividades (avaliativas ou não). Pensando nisso, decidiu-se agregar à TC, a TOEDC de Bloom et al., a fim de acompanhar o desenvolvimento do estudante de uma forma mais detalhada, podendo identificar as dificuldades do mesmo de forma mais específica. Assim sendo, a TOEDC fundamentou a forma no qual o professor deve criar e avaliar suas atividades (seção 5.2).

Por se tratarem de teorias complexas, requerendo do professor uma análise detalhada de cada estudante, as mesmas foram aplicadas em uma RB, sendo necessário estudos bibliográficos sobre o tema (seção 5.5). Na próxima seção é apresentado como foram coletados os dados na pesquisa documental.

3.3.2 Pesquisa documental

Através de pesquisas em PPCs (matrizes curriculares e ementas dos cursos de TI) dos Campi do IFRS, foram coletadas informações gerais sobre os cursos a fim de conhecer o perfil destes. O principal intuito desse levantamento era reunir dados relacionados à disciplina de Programação ministrada no primeiro semestre dos cursos

superiores. Ainda constam dados a respeitos destes cursos em documentos do Ministério da Educação: CNCST e Catálogo Nacional de Cursos de Bacharelado (CNCB), bem como no INEP sobre o censo da educação superior. Esses dados foram utilizados para compreender a construção dos PPC, por meio do órgão que regulamenta, controla e avalia os cursos.

Para conhecer o perfil das turmas sobre aprovação, reprovação e evasão da disciplina de LPI do curso STSI do IFRS campus Porto Alegre, analisou-se os cadernos de chamadas de 2011 a 2016. O intuito dessa coleta também foi possuir dados quantitativos para verificar se houve ou não melhora nesses índices com a utilização da ConsProg. Na próxima seção será apresentado como foram gerados e coletados dados no estudo de caso.

3.3.3 Estudo de caso

Durante a pesquisa realizou-se um estudo de caso em três etapas: 2016/1, 2016/2, e 2017/1. A primeira etapa, que ocorreu em 2016/1, teve o intuito de gerar dados sobre: as metodologias utilizadas pelo professor durante as aulas, o perfil dos estudantes da disciplina, como eram aplicadas e acompanhadas as atividades (avaliativas ou não) pelo professor, e com isso o que informações podia coletar a respeito do aprendizado dos estudantes. Esses dados foram gerados por meio de observação das aulas durante a disciplina, e registrados em um diário de campo. No último dia letivo do semestre realizou-se um questionário fechado com os sujeitos participantes da pesquisa em que se pode conhecer a opinião dos mesmos sobre a disciplina e a metodologia aplicada, bem como o perfil dos mesmos.

Além disso, o professor da turma (2016/1) disponibilizou o acesso aos materiais utilizados durante as aulas no AVA Moodle para os estudantes. Com essas informações foi possível conhecer o que havia sido planejado para as aulas, e as atividades aplicadas ao longo do semestre. Também se pode verificar dados AVA a respeito da entrega das atividades práticas e questionários teóricos. Os dados oriundos dessa primeira etapa foram utilizados como base para desenvolver a proposta pedagógica ConsProg, levando em conta as necessidades dos estudantes.

Em 2016/2 realizou-se a segunda etapa do estudo de caso, no qual o professor da disciplina utilizou a ConsProg em suas aulas. Durante estas, observações foram

feitas e registradas no diário de campo, no qual gerou dados qualitativos a respeito da aplicação da ConsProg. Assim como na primeira etapa do estudo de caso, o professor liberou o acesso ao AVA Moodle para que fossem coletados dados sobre as resoluções das atividades, materiais de aula e cronograma da disciplina.

Com a intenção de conhecer a opinião dos sujeitos participantes a respeito do semestre, realizou-se na última aula um questionário fechado, no qual continha: perguntas a respeito do semestre, da metodologia, das dificuldades e sobre a dedicação dos mesmos. Os dados gerados e coletados durante o semestre foram importantes para avaliar a proposta pedagógica. Essas informações também foram relevantes para definir as relações e dependências dos conteúdos a fim de facilitar a identificação das dificuldades enfrentadas pelos estudantes.

No semestre 2017/1 foi realizada a terceira etapa do estudo de caso, que consistiu na reaplicação da proposta pedagógica com os ajustes que foram necessários. Neste semestre o professor contou com o mapeamento de relações e dependências dos conteúdos. Diferente dos outros semestres, não foram feitas observações pela pesquisadora durante as aulas, visto que nesta última aplicação a intenção é de conhecer a visão do professor da disciplina quanto a utilização da ConsProg.

Os dados do AVA Moodle também foram coletados, em especial o resultado dos questionários, que serviram de conjunto de dados de entrada para o treinamento da RB. Um questionário fechado foi aplicado pelo professor da disciplina, para que fosse possível conhecer o perfil dos estudantes quanto a seus conhecimentos prévios, experiências com programação e disponibilidade de tempo. No final, o docente entregou um relatório sobre a aplicação da ConsProg. Assim, os dados gerados e coletados foram utilizados para analisar a aplicação da proposta, bem como contribuir para a construção da RB.

Além disso, contou-se também com os diários de classe dos três semestres analisados, referentes a disciplina LPI. Ademais, analisou-se os cadernos de chamadas desde 2011, a fim de investigar o perfil dos estudantes quanto a aprovação, reprovação e evasão nos diferentes turnos em que se oferta a disciplina. Sendo assim, averiguaram-se um total de dez cadernos de chamada. O capítulo a seguir apresenta os trabalhos correlatos investigados na pesquisa bibliográfica.

4 TRABALHOS CORRELATOS

Através da pesquisa bibliográfica foi possível fazer o levantamento sobre o que foi pesquisado nos últimos três anos (2014 a 2016) no Brasil a respeito de ensino-aprendizagem de programação. A busca de trabalhos correlatos foi realizada nos anais dos eventos, workshops, revistas e congressos da Sociedade Brasileira de Computação (SBC). Durante o ano, a SBC promove junto a outras instituições em torno de 24 eventos de computação, possuindo também duas revistas. Dentro desses, seis eventos e uma revista são específicos da área de educação:

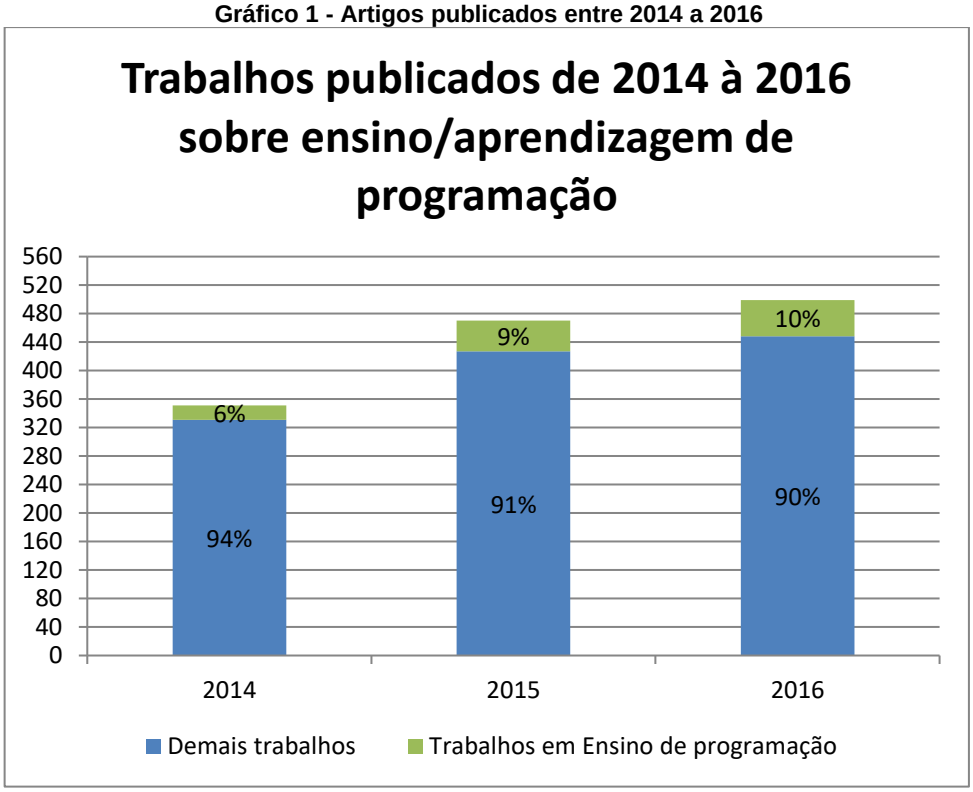
- DesafIE – Desafios da Computação aplicada à Educação
- WEI – Workshop sobre Educação em Computação
- WIE – Worksop de Informática na Escola
- JAIE – Jornada de Atualização em Informática na Educação
- SBIE – Simpósio Brasileiro de Informática na Educação
- CBIE – Congresso Brasileiro de Informática na Educação
- RBIE – Revista Brasileira de Informática na Educação

Inicialmente fez-se o levantamento nos títulos dos artigos publicados nesses eventos e na revista, fazendo o uso das palavras-chave: ensino e/ou aprendizagem de programação. Aqueles que apresentaram em seu título que o trabalho tratava-se de um estudo à nível de ensino fundamental, foram descartados, uma vez que a pesquisa ocorreu no IFRS e o mesmo não contempla esse nível de ensino. Logo, selecionou-se 114 trabalhos dos 1320 publicados nesses eventos, no período mencionado anteriormente[7].

Os artigos selecionados foram organizados por ano, e categorizados por evento, com isso foi possível obter dados como os que podem ser observados no Gráfico 1. Nele é possível identificar que trabalhos relacionados ao ensino/aprendizagem de programação representam 6% dos trabalhos publicados no ano de 2014, tendo um acréscimo de 3% no ano de 2015, e atingindo 10% no ano de 2016. Assim sendo, pode-se afirmar que o tema em questão vem crescendo gradualmente no decorrer dos anos. O quadro organizado com o primeiro levantamento de trabalhos correlatos encontra-se nos apêndices (2014 - Apêndice E; 2015 - Apêndice F; 2016 - Apêndice G).

[7] No ano de 2014, as páginas que possuem os anais dos eventos WEI e DesafIE estão fora do ar, não sendo possível verificar se havia trabalho relacionado a está pesquisa

Gráfico 1 - Artigos publicados entre 2014 a 2016



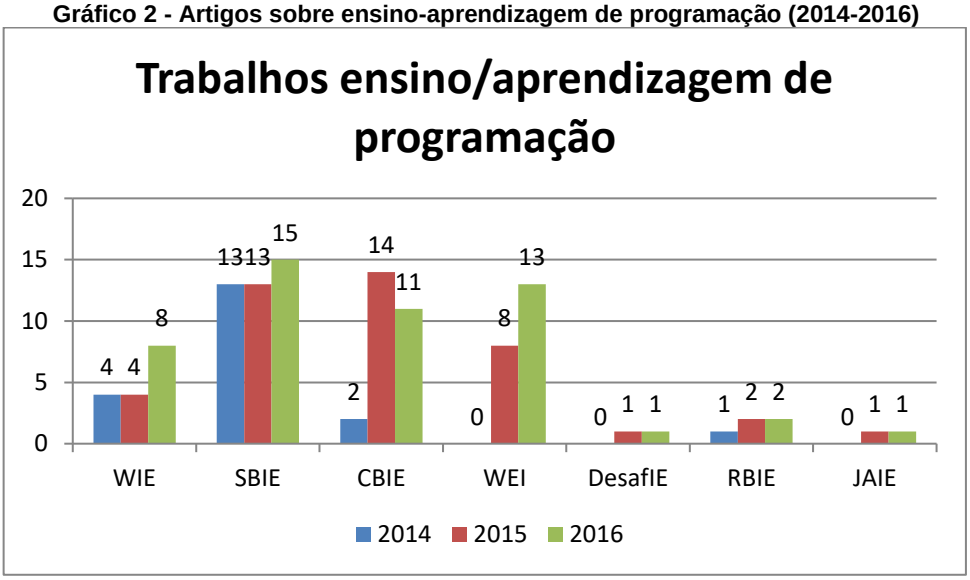
Fonte: A autora

Além do que já foi exposto anteriormente, o Gráfico 1 também mostra que a quantidade de trabalhos publicados na área de informática na educação também apresenta um crescimento significativo. Uma vez que os dados foram separados por ano e categorizados por evento, foi possível verificar qual dos eventos da SBC tem maior publicação sobre o tema pesquisado. Logo, observando o Gráfico 2 pode-se afirmar que o SBIE e o CBIE são os eventos que publicaram mais artigos relacionados com a presente pesquisa.

Durante os três anos analisados, o SBIE apresentou 41 trabalhos sobre ensino/aprendizagem de programação, enquanto que o CBIE possui 27 publicações sobre o mesmo. Pode-se dizer com isso, que o SBIE seria o evento mais propício para a publicação de artigos sobre o tema proposto nessa dissertação.

Apesar de se ter selecionado apenas 8,6% artigos dos 1320 analisados, muitos ainda se distanciam do tema dessa pesquisa. Assim sendo, avaliou-se os resumos dos 114 artigos, foram descartados aqueles que não tratavam de uma pesquisa relacionada ao ES. Restaram, então, 50 trabalhos (3,8% dos 1320) na segunda filtragem.

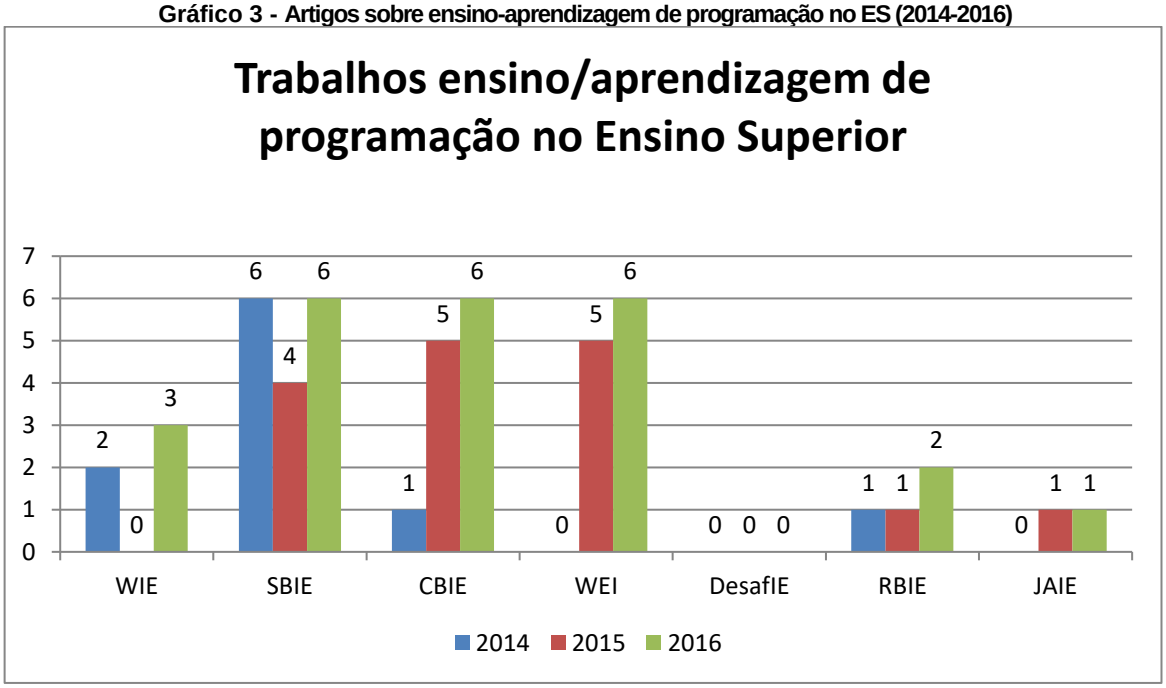
Gráfico 2 - Artigos sobre ensino-aprendizagem de programação (2014-2016)



Fonte: A autora

Os dados oriundos dessa nova análise continuaram separados por ano, e classificados por eventos. Os quadros resultantes desse levantamento podem ser conferidos nos apêndices: Apêndice H – 2014; Apêndice I – 2015; Apêndice J - 2016. Pode-se com esses dados recriar o Gráfico 2, suprimindo aqueles descartados nessa verificação mais detalhada. Logo, ao visualizar o Gráfico 3, pode-se verificar que o SBIE e o CBIE ainda apresentam maior número de publicações correlatas a essa pesquisa, porém, o WEI se aproxima do CBIE, com apenas um trabalho a mais.

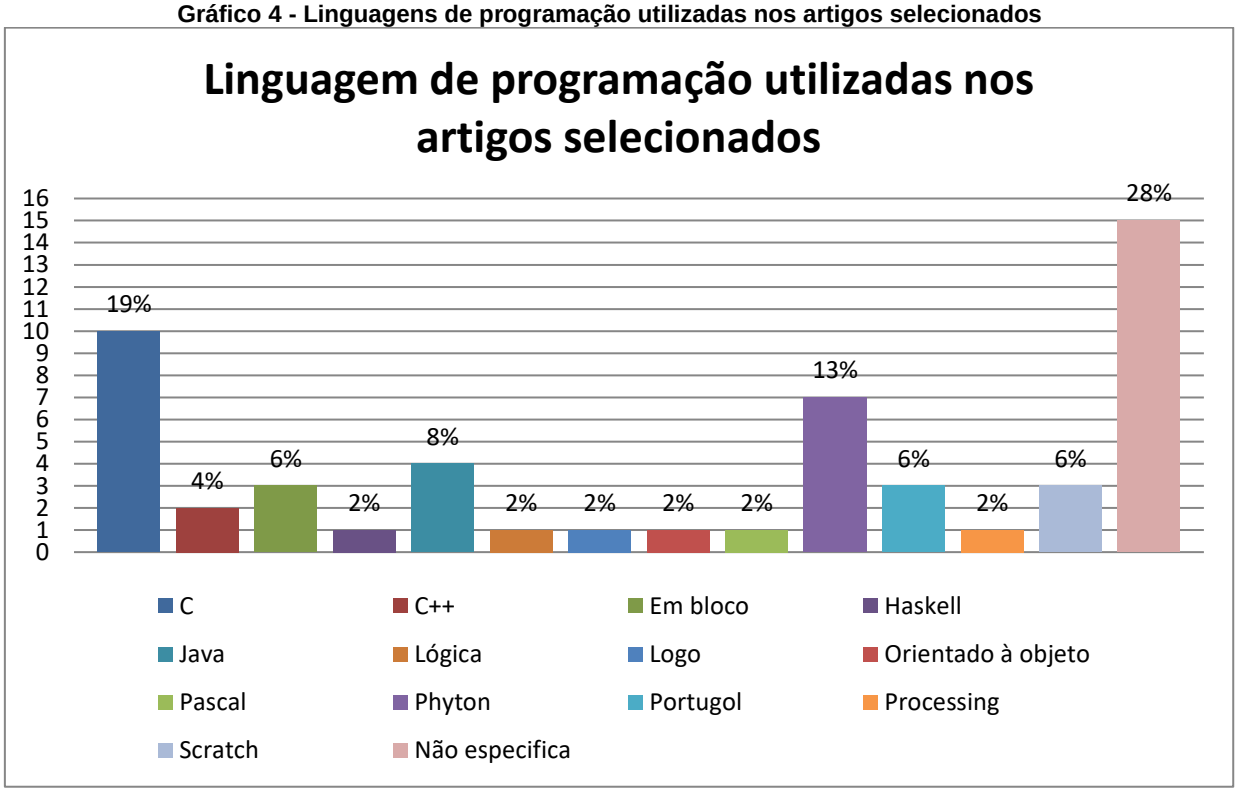
Gráfico 3 - Artigos sobre ensino-aprendizagem de programação no ES (2014-2016)



Fonte: A autora

Com o intuito de verificar se existem trabalhos semelhantes já publicados, e também para aprofundar o conhecimento a respeito do estado da arte no ensino/aprendizagem de programação de nível superior, foram lidos os 50 trabalhos. Seguindo a mesma organização dos dados anteriores, esses foram separados por ano, classificados por evento e acrescido da linguagem de programação no qual o artigo trata, bem como o(s) objetivo(s) e também se a pesquisa foi aplicada numa disciplina introdutória de programação, nos semestres iniciais do curso.

Gráfico 4 - Linguagens de programação utilizadas nos artigos selecionados



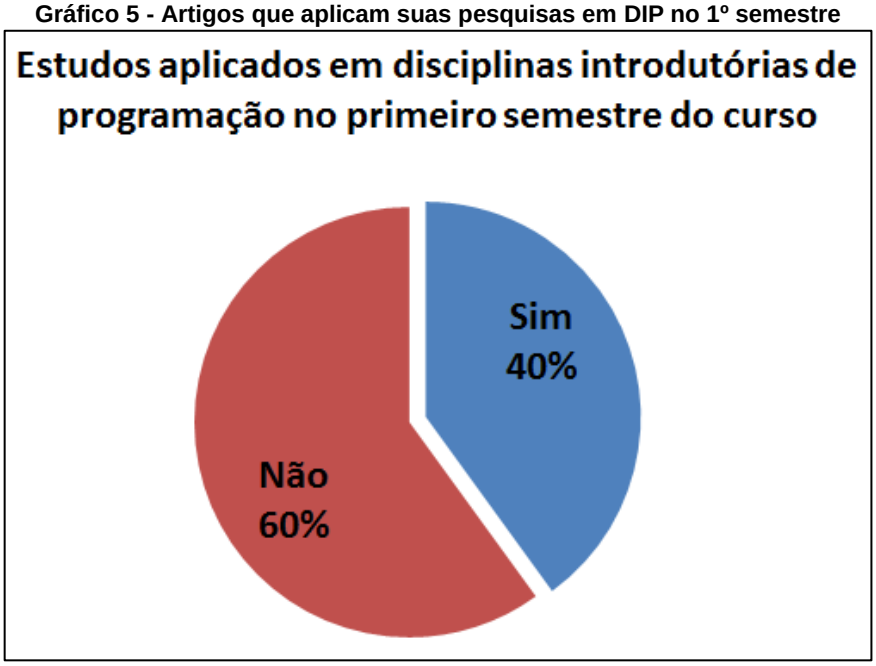
Fonte: A autora

Com o conhecimento das linguagens analisadas nos artigos, foi possível criar o Gráfico 4, que apresenta quais linguagens foram utilizadas nas pesquisas. É possível visualizar nesse gráfico que 28% dos artigos não especificam a linguagem de programação. Já nos 72% que especificam a linguagem, pode-se averiguar que as mais citadas foram: C representando 19% dos trabalhos, Phyton com 13% e Java com 8%. Vale ressaltar que a presente pesquisa está relacionada ao ensino-aprendizagem de programação, utilizando-se da linguagem de programação C, que se encontra de acordo com os estudos atuais, o que a torna bastante relevante.

Nessa terceira análise dos artigos foi verificado se as pesquisas eram aplicadas em disciplinas introdutórias de programação. O Gráfico 5 apresenta esse resultado,

no qual 40% desses trabalhos realizam suas pesquisas diretamente nas disciplinas de programação. Já os outros 60% publicam dados relacionados a pesquisas bibliográficas, estudos de casos em projetos de extensão, minicursos, etc.

Gráfico 5 - Artigos que aplicam suas pesquisas em DIP no 1º semestre



Fonte: A autora

Para melhor compreensão, os dados relacionados ao Gráfico 5 serão divididos em duas seções. A primeira seção irá aprofundar sobre os 60% dos artigos que não aplicam suas pesquisas em disciplinas de programação, e na segunda seção, apresentar-se-á aqueles que fazem essa aplicação.

4.1 ARTIGOS QUE NÃO APLICAM SUAS PESQUISAS EM DISCIPLINAS INTRODUTÓRIAS DE PROGRAMAÇÃO

Numa análise mais detalhada desses 60% (Gráfico 5) que não aplicam suas pesquisas em disciplinas introdutórias de programação, em 2014 têm-se apenas três trabalhos. Estes apresentam estudos referentes à linguagem de programação Phyton e C, destacando-se o trabalho de Barbosa et al. (2014), que investigam sobre a influência da linguagem de programação no ensino-aprendizagem de CIP, apontando a linguagem Phyton como a mais favorável para o mesmo.

Já em 2015 encontram-se 11 artigos que não foram aplicados em uma disciplina introdutória de programação. A maioria destes não especifica a linguagem

de programação que pesquisam, e tratando-se de levantamentos bibliográficos. O artigo que auxiliou com dados para o desenvolvimento da ConsProg foi o de Gomes et al. (2015). Os autores apresentam as dificuldades que os estudantes encontram ao aprender CIP, apontando como principal dificuldade o reconhecimento das mensagens de erro geradas pelo compilador e como devem corrigi-las.

Seguindo a investigação dos artigos que não aplicam suas pesquisas em disciplinas introdutórias de programação, em 2016 foram selecionados 24 trabalhos. Dentre esses, 16 são pesquisas relacionadas à ferramenta Scratch e outras que também utilizam programação em bloco. Investiga-se também sobre o uso de jogos e *gamification*, como fatores motivacionais para o ensino-aprendizagem de programação, e a criação de ferramentas que possuem *feedback* sobre as atividades realizadas pelos estudantes.

Dentre esses 16 trabalhos, o de Raabe et al. (2016) destaca-se por verificar a influência dos enunciados no desempenho na resolução das atividades introdutórias de programação. De acordo com os autores, “[...] é importante que o enunciado seja contextualizado em um contexto conhecido, que as informações sejam apresentadas de forma explicita, com indícios do processo de resolução e com exemplificação” (RAABE et al., p.81, 2016). Na próxima seção ter-se-á os trabalhos que aplicam suas pesquisas em disciplinas introdutórias de programação, representando 40% dos trabalhos selecionados, como visto no Gráfico 5.

4.2 ARTIGOS QUE APLICAM SUAS PESQUISAS EM DISCIPLINAS INTRODUTÓRIAS DE PROGRAMAÇÃO

Os trabalhos aplicados em disciplinas de programação, no primeiro semestre do curso, representam 40% dos 50 selecionados. No ano de 2014 encontram-se sete trabalhos, sendo um deles semelhante à pesquisa do presente trabalho. Intitulado “Integração de uma Metodologia de Ensino Presencial de Programação com um Sistema Tutor Inteligente”, González & Tamariz (2014) apresenta uma proposta metodológica para auxiliar o ensino presencial de programação, por meio de um ambiente virtual denominado “HALYEN”.

A metodologia apresentada pelos autores, faz o uso de uma abordagem pedagógica desenvolvida por Delgado et al. (p.1, 2004), essas propõem o uso de “[...]

recursos didáticos voltados para o desenvolvimento da capacidade de abstração, do raciocínio lógico, da solução de problemas e da autonomia cognitiva". De acordo com González & Tamariz, o uso da abordagem pedagógica de Delgado et al. (2004) agregado ao ambiente virtual, mostrou-se promissor, uma vez que os estudantes se sentiram motivados e apresentaram resultados positivos no aprendizado de programação.

No ano de 2015, foram identificados cinco artigos que fizeram a aplicação da pesquisa em disciplinas de programação. Esses trabalhos apresentam alternativas para o ensino-aprendizagem de programação, como: ampliação de espaços de aprendizagem que vão além do horário estipulado na disciplina, fazendo o uso de enunciados que se aproximem dos interesses dos estudantes; trabalhar com dinâmicas de grupo, programação por blocos e kits de Arduino; aumentar a participação e motivação por meio das teorias de Gamificação; desenvolver e aplicar ferramentas de auxílio de resolução de atividades.

Em 2016, oito trabalhos fazem parte dos 40% selecionados (Gráfico 5). O artigo de Carvalho et al. (2016) mostrou ser semelhante à presente pesquisa, uma vez que propuseram reformular a metodologia utilizada na disciplina de programação. A reformulação consistiu na troca da linguagem de programação para Phyton e na reestruturação dos conteúdos ministrados.

Durante todo o processo de seleção de artigos, foi possível observar que muito se pesquisa sobre ensino-aprendizagem de CIP. A maioria dos trabalhos relata e traz dados sobre as dificuldades encontradas pelos estudantes ao aprenderem a programar, alguns tentam propor alternativas que facilitem esse aprendizado, indo desde propostas de alteração da linguagem de programação, até o desenvolvimento de aplicativos com uso de inteligência artificial. Na próxima seção serão apresentadas algumas considerações a respeito dos trabalhos correlatos.

4.3 CONSIDERAÇÕES

O presente trabalho se assemelha aos artigos selecionados dos eventos educacionais da SBC. Assim como a maioria deles, a pesquisa trata sobre ensino-aprendizagem de programação e faz o uso da Linguagem C ANSI (Gráfico 4), que foi

a linguagem mais utilizada entre eles. Neste sentido, pode-se dizer que a escolha dessa é relevante.

Outro ponto em que há semelhança é no que diz respeito às dificuldades dos estudantes em aprenderem a programar, no qual é apontado o nível de abstração um dos principais fatores que dificultam a aprendizagem. Além disso, os estudos relatam que é necessário encontrar métodos de ensino para que os elevados índices de retenção nessas disciplinas sejam reduzidos. Logo, os autores dos artigos sugerem o uso de: aplicativos, gamificação, programação em bloco, Arduino, entre outras metodologias, para tornar o ensino de CIP mais atrativa e menos abstrata. Porém não fundamentam essas propostas com teorias educacionais, a fim de propor algo menos técnico e conteudistas.

Com isso pode-se observar que a proposta pedagógica ConsProg, resultante da presente pesquisa, mostra-se diferenciada, uma vez que traz teóricos como Piaget e Bloom et al. para fundamentá-la. Além disso, a mesma sugere um ensino voltado ao estudante, em que o desenvolvimento das aulas deve se dar de acordo com o aprendizado da turma. Mais do que isso, a proposta busca, através de uma RB, apontar as possíveis dificuldades dos estudantes e assim o professor poderá planejar suas aulas com o intuito de amenizar essas carências tanto da turma quanto no individual.

Vale ressaltar que essas teorias foram escolhidas por meio de um estudo bibliográfico, no qual se buscou fundamentações adequadas para o propósito das disciplinas de Linguagem de Programação. A proposta pedagógica vai além de uma simples metodologia de ensino que apresenta ferramentas para ensino-aprendizagem de programação, uma vez que busca explicar em que momento o estudante apresentou baixo rendimento e como trabalhar com isso. Sendo assim, a ConsProg também auxilia o professor em diagnosticar esse baixo rendimento, e podendo ele criar estratégias para recuperar os conteúdos[8] nos quais os estudantes apresentaram dificuldades. O próximo capítulo traz a fundamentação teórica que sustenta a presente de dissertação, bem como a proposta pedagógica desenvolvida.

[8] De acordo com a LDBEN (BRASIL, 1996, art. 13) o professor é incumbido de “estabelecer estratégias de recuperação para os alunos de menor rendimento”.

5 FUNDAMENTAÇÃO TEÓRICA

Para melhor organização, o presente capítulo encontra-se dividido em cinco seções, no qual a primeira aborda a construção do conhecimento segundo a TC de Jean William Fritz Piaget. Já a segunda seção apresenta a TOEDC, desenvolvida por Bloom et al.

A pesquisa trata-se do ensino-aprendizagem de programação, contudo, o conceito de programação não se dará de forma genérica, visto que o local que a pesquisa está sendo aplicada trabalha com a Linguagem de Programação C, far-se-á o uso da mesma. Por conseguinte, a terceira seção traz os conceitos introdutórios da Linguagem de Programação C. Na quarta seção é possível conhecer alguns dos desafios e dificuldades, encontradas por estudantes e professores, no ensino-aprendizagem dos CIP. Por último, tem-se a seção que embasa a sistematização da análise das atividades, ou seja, apresenta os conceitos relacionados RB.

5.1 A CONSTRUÇÃO DO CONHECIMENTO SOB A PERSPECTIVA DE PIAGET

Com o intuito de criar uma abordagem pedagógica para o ensino-aprendizagem de programação, utilizou-se a TC de Piaget para fundamentá-la. Acredita-se que a TC seja adequada para disciplinas relacionadas com programação, uma vez que o aprendizado dos conteúdos destas desenvolve-se ao longo das aulas, no qual um conteúdo está diretamente relacionado com o outro. O Construtivismo vê o ensino-aprendizagem como patamares em que normalmente o sujeito (estudante) parte de um conhecimento empírico e através de coordenações adquire novos conhecimentos, atingindo assim um novo patamar por meio da reflexão (PIAGET, 1995).

Essa reflexão proporciona ao sujeito combinações que refletirá no aprendizado de novos conceitos, que partirão, mais uma vez, do conhecimento empírico. Logo, o sujeito colhe no patamar anterior informações para o patamar posterior, havendo assim um “ato mental de reconstrução e reorganização sobre o patamar superior daquilo que foi assim transferido do inferior” (PIAGET, 1995, p.6).

Neste sentido, se torna imprescindível compreender como ocorre a construção do conhecimento para se criar uma proposta adequada ao ensino-aprendizagem dos CIP. Entendendo a forma em que esse processo acontece, o professor pode criar

possibilidades favoráveis ao aprendizado dos estudantes. Assim sendo, de acordo com Piaget (1983), durante o desenvolvimento cognitivo, o sujeito passa por três estágios: o primeiro é o sensório-motor (do nascimento até dois anos); o segundo, é dividido em dois subestágios, a preparação para operações lógico-concretas (de dois a sete anos) e as operações lógico-concretas (dos sete até onze anos); e o terceiro estágio é a lógica formal (a partir dos onze anos).

Para Piaget, estágio é a forma no qual ocorre a “[...] organização da atividade mental, sob seu duplo aspecto: por um lado, motor ou intelectual, por outro, afetivo [...]” (PIAGET, 1983, p. XII). É no terceiro estágio que o sujeito vai ter condições cognitivas de programar, uma vez que não dependerá mais do concreto para compreender que algo é possível, e conseguirá relacionar o que é possível ao necessário (PIAGET, 1983). Em outras palavras, a partir deste estágio o sujeito age de forma consciente e lógica em relação a suas ideias (PIAGET, 1983), características estas necessárias para programar.

Assim sendo, um estudante de ES - conforme a estrutura escolar brasileira na faixa etária a partir dos 17 anos (BRASIL, 1996) - teria condições cognitivas para o aprendizado de programação, uma vez que, em teoria, já atingiu o estágio lógico-formal. De acordo com a teoria de Piaget (1983), a lógica formal é quando o sujeito, a partir dos onze anos, consegue ultrapassar o real, pensar logicamente e também abstrair conceitos. Contudo, não ocorre um simples despertar e o sujeito está apto para o estágio lógico formal, existe todo um processo que ocorre desde o nascimento e continua durante toda sua existência.

Esse processo está diretamente ligado ao desenvolvimento das funções cognitivas, sendo estas, habilidades e processos que se estruturam ao logo da vida do ser humano. Neste sentido, Piaget (1983) afirma que não se pode determinar uma idade exata para cada estágio, porém a ordem delas sempre se mantém, ou seja, "[...] para atingir um certo estágio, [...] é necessário ter construído as pré-estruturas, as subestruturas preliminares que permitem progredirmos mais" (idem, p.215). Esta teoria se aplica diretamente à programação, pois um estudante não conseguirá aprender e aplicar os conceitos de condicionais, se o mesmo não conseguiu, pelo menos, compreender variáveis. Esse tipo de relação será detalhado na seção 5.3 - Introdução aos conceitos de programação.

Retomando, é válido pensar que as turmas em níveis de graduação têm uma variação de idade muito grande, que podem ir dos 17 até, quem sabe, 80 anos. Nesse

contexto, de acordo com os estágios, todos estudantes, apesar da diferença de idade, estariam no estágio lógico-formal, e em tese pensariam logicamente igual. Porém é necessário compreender que cada sujeito tem seu próprio tempo e experiência que podem levar em conta dois aspectos:

> psico-social, quer dizer tudo que [...] se aprende por transmissão familiar, escolar, educativa em geral; e depois, [...] psicológico, [...] que é o desenvolvimento da inteligência, [...] o que não foi lhe ensinado, mas ela deve descobrir sozinha (PIAGET, p.211, 1983).

Além disso, o estágio lógico-formal pode iniciar por volta dos 11 anos, e as operações do pensamento não são igual a de um adulto, entre os 12 ou 15 anos, o sujeito começa a "raciocinar sobre os enunciados verbais, proposicionais;" torna-se capaz de "manipular hipóteses, de raciocinar a partir do ponto de vista do outro" (PIAGET, 1983, p.220). Mas, apesar da teoria, Piaget (1983) acredita que se pode ter muitas variações nas idades destes estágios, que podem ser ocasionados por efeito da sociedade em que o sujeito vive. Para compreender os fatores que podem levar a essas variações, Piaget (p. 224, 1983) dividiu em quatro: "primeiro fator: a hereditariedade, a maturação interna. [...] segundo fator: a experiência física, a ação dos objetos. [...] terceiro fator: transmissão social, o fator educativo. [...] quarto fator [...] equilibração [...]".

O quarto fator, equilibração, é fundamental, pois será através do equilíbrio entre o confronto daquilo que o sujeito já conhece, com aquilo que ele está conhecendo, que favorecerá o desenvolvimento cognitivo do mesmo (PIAGET, 1983). Nesse sentido, deve-se cuidar para que não haja aceleração nesse processo, pois de acordo com Piaget,

> muita aceleração corre o risco de romper o equilíbrio. O ideal da educação não é aprender ao máximo, maximalizar resultados, mas é antes de tudo aprender a aprender; é aprender a se desenvolver e aprender a continuar a se desenvolver depois da escola (1983, p. 225).

Sabendo disso, é importante que o professor respeite o processo cognitivo de seu estudante, para que o mesmo consiga se apropriar do conteúdo e assim haja equilibração para o aprendizado do próximo conteúdo. Quando um estudante se apropria de um determinado conceito, isso fica muito evidente, Piaget (1983) dá o exemplo dos números inteiros, uma vez que a criança os compreende isso se torna estável e não se modifica mais durante toda a vida. Logo, Piaget afirma que

> podemos caracterizar os estágios numa população dada por uma cronologia, mas essa cronologia é extremamente variável; ela depende da experiência anterior dos indivíduos, e não somente de sua maturação, e depende principalmente do meio social que pode acelerar ou retardar o aparecimento de um estágio, ou mesmo impedir sua manifestação (p.235, 1983).

Outro elemento importante para o aprendizado é a práxis, que para Piaget (1983) não é apenas a ação de movimentos, mas a ação voluntária em sua totalidade, consciente de seu resultado. Por meio das práxis o estudante coordena e regula suas ações, porém isto não se desenvolve apenas pela aprendizagem ou pelo condicionamento das associações, mas por esquemas de associações cumulativas que consistirá em assimilações (Piaget, 1983). De acordo com Piaget, assimilação baseia-se em associar um "[...] objeto ou situação a um esquema anterior aumentando assim esse esquema" (1983, p.245).

Toda ação é oriunda de uma natureza afetiva, ou seja, deve haver necessidade e satisfação para que o sujeito assimile um objeto a um esquema. A coordenação desses esquemas resulta, segundo Piaget (1983), na inteligência. Contudo, ainda existe a "resposta à ação dos objetos sobre os esquemas, sincronizando com a assimilação dos objetos aos esquemas", que Piaget (1983, p. 251) chamou de acomodação. Sendo assim, o aprendizado ocorre na equilibração, que é o processo de auto regulação do sujeito entre a assimilação e a acomodação, resultando na adaptação do mesmo ao meio.

Pode-se dizer com isso que para ocorrer a construção do conhecimento precisamos de três elementos fundamentais: equilibração, práxis e afetividade. Através da equilibração o estudante terá a assimilação e a acomodação do objeto. Nas práxis ele irá coordenar suas ações para que ocorra a equilibração. Já na afetividade, o estudante por meio da necessidade e da satisfação irá exercer a práxis. Para Piaget toda necessidade

> tende a incorporar as coisas e pessoas à atividade própria do sujeito, isto é ‘assinalar’ o mundo exterior às estruturas já construídas, e a reajustar estas últimas em função das transformações ocorridas, ou seja, ‘acomodá-las’ aos objetos externos (PIAGET, 1995, p.17).

Sendo assim, “o conhecimento resulta da interação entre sujeito e objeto que são mais ricas do que aquilo que os objetos podem fornecer por eles” (PIAGET, 1995, p.87). Portanto, para que ocorram os três elementos: equilibração, práxis e afetividade, é necessário haja interação entre sujeito (estudante) e objeto

(programação). Uma vez que a TC é bastante complexa, sentiu-se a necessidade de uma teoria que pudesse qualificar o aprendizado dos estudantes de uma forma mais específica, logo, fez-se o uso da TOEDC que será apresentada na seção seguinte.

5.2 AVALIAÇÃO E CRIAÇÃO DE ATIVIDADES SEGUNDO A TAXONOMIA DE BLOOM

Taxonomia é um termo bastante utilizado na biologia, refere-se normalmente a uma classificação científica de animais, plantas etc., de acordo com características sistemáticas (BLOOM et al., 1972). Com isso, Bloom et al. (1972) desenvolveram uma taxonomia de objetivos educacionais constituída por três partes: domínio cognitivo, afetivo e psicomotor. Essa mesma divisão é vista na TC, no qual o desenvolvimento do sujeito se dá através do cognitivo, afetivo e motor (PIAGET, 1983).

Mesmo tendo as três partes, Bloom et al. (1972) aprofundaram-se no cognitivo e no afetivo, deixando um pouco de lado o psicomotor, pois na escola secundaria e ES (objetos de estudo dos mesmos) haviam poucos estudos a esse respeito, assim, acreditaram não ser muito útil o aprofundamento sobre o mesmo. Para a presente pesquisa, utilizou-se o domínio cognitivo, uma vez que a ideia de afetividade é abordada por Piaget (1983) no sentido de afetar-se, sensibilizar-se com a situação, ou seja, quando há interação entre sujeito e objeto como um todo.

O domínio cognitivo foi desenvolvido na taxonomia dos objetivos educacionais a fim de auxiliar professores com um modelo relativamente preciso de análise dos resultados educacionais na área cognitiva, que engloba memorização, pensamento e solução de problemas (BLOOM et al., 1972). Os conceitos da TOEDC não remetem a uma classificação, mas a uma hierarquia com o objetivo de auxiliar na identificação do desenvolvimento cognitivo do indivíduo. Para isso, criam-se princípios organizacionais que levam em conta o comportamento esperado do estudante, ou seja, a forma no qual se espera que o estudante pense, haja e sinta na construção do conhecimento. Logo, como uma fonte de reflexão a respeito dos problemas educacionais, a taxonomia dos objetivos educacionais de domínio cognitivo proporciona ao professor orientações para o desenvolvimento de técnicas de ensino e avaliação (BLOOM et al., 1972).

Ao se utilizar as orientações da taxonomia de Bloom, deve-se saber que a solução de um problema não ocorre no “vácuo”, mas baseado em conhecimentos anteriores e das vivencias do estudante, Piaget (1975, p.387) afirma que “as estruturas não estão pré-formadas dentro do sujeito, mas constroem-se à medida das necessidades e das situações”. Sendo assim, Bloom et al. (1972) traz algumas decisões que devem ser tomadas toda vez em que haverá objetivos cognitivos, ou seja, de aprendizagem, que são:

> Qual porção de conhecimento que se deve exigir do estudante? Em que medida necessita o estudante aprender precisamente esse conhecimento? Qual a melhor forma de organizar o conhecimento com vistas a sua aprendizagem? Como podem os conhecimentos exigidos serem significativos para o estudante? (BLOOM, 1972, p.32).

Levando em conta essas questões, é esperado que o professor tenha como resultado de seus estudantes o conhecimento, porém apenas este não é considerado como único, uma vez que o estudante deve conseguir aplicar o novo conhecimento em novos problemas. O processo do estudante conseguir aplicar um determinado conhecimento em novos problemas, Piaget (1995) chamou de abstração reflexionante, que é constituída por diversos patamares que vai desde a necessidade da presença do objeto até a apropriação total do mesmo.

Esse mesmo processo chamado de abstração reflexionante por Piaget (1995), Bloom et al. (1972, p.34) empregaram a expressão “capacidade e habilidades intelectuais”, que significa “buscar em suas experiências anteriores informações e técnicas apropriadas para examinar e solucionar novos problemas” (idem). Para Bloom et al., a habilidade é oriunda dos processos mentais, que requer do estudante uma organização e reorganização de seus conhecimentos a fim de solucionar um problema novo, e a capacidade intelectual refere-se à combinação do conhecimento com a habilidade.

Com o intuito de o estudante atingir a capacidade e habilidade intelectual, a taxonomia dos objetivos educacionais de domínio cognitivo (BLOOM et al., 1972) é composta por seis níveis: conhecimento, compreensão, aplicação, análise, síntese e avaliação. Esses níveis são assim organizados, “no sentido de se obter evidência na medida em que os comportamentos desejados foram desenvolvidos pelo aluno” (BLOOM et al., 1972, p.11), e corresponde a uma hierarquia cumulativa que vai do mais simples ao mais complexo, sendo o nível anterior pré-requisito do posterior. Fica

claro nesse momento que o conhecimento é trabalhado de forma construtiva, com o cuidado de dificultar gradativamente o entendimento do conceito (objeto) para que o estudante (sujeito) se aproprie do mesmo.

Logo, para cada nível, Bloom et al. (1972) indicam verbos que podem expressar o que é esperado do estudante em determinada atividade. No Quadro 2 é possível observar alguns verbos organizados por níveis e sua utilização facilita na criação das atividades e avaliação dos estudantes, permitindo assim um acompanhamento mais efetivo da aprendizagem dos mesmos.

Quadro 2 - Lista de verbos de acordo com cada nível da Taxonomia

CONHECIMENTO	COMPREENSÃO	APLICAÇÃO	ANÁLISE	SÍNTESE	AVALIAÇÃO
Definir	Compreender	Aplicar	Verificar	Relatar	Julgar
Apresentar	Interpretar	Concluir	Distinguir relações	Improvisar	Estimar
Reconhecer	Predizer	Empregar	Compreender inter-relações	Integrar	Indicar
Ordenar	Alterar	Solucionar	Analisar	Projetar	Comparar
Relacionar	Selecionar				Avaliar
Listar	Explicar				Identificar

Fonte: Bloom et al., (1972)

Ao analisar os níveis criados por Bloom et al. (1972) na TOEDC, é possível verificar que esta foi baseada na ação do sujeito, que é vista por Piaget (1983) como interação entre o objeto (conteúdo) e o sujeito (estudante). Logo, acredita-se que cada verbo sugerido na taxonomia pode ser uma ou um conjunto de ações do estudante, buscando seu desenvolvimento cognitivo, podendo resultar na aprendizagem do conteúdo. Além disso, cada nível pode ser relacionado à teoria de Piaget (1995) quanto à abstração reflexionante.

Uma vez que o primeiro nível da taxonomia depende do concreto, caracteriza-se, para Piaget (1995), numa abstração empírica, no qual o sujeito depende do objeto para retirar dele características e informações para as ações. Neste caso, o estudante necessita relacionar o novo conhecimento com algo já vivido para conseguir resolver um dado problema. Assim sendo, se o estudante vai aprender variáveis, que de acordo com Deitel (2011) “é uma posição na memória onde um valor pode ser armazenado para ser utilizado por um programa”, o professor poderá fazer uma

analogia com o cotidiano do estudante para facilitar o entendimento. Um exemplo a se utilizar pode ser a comparação das variáveis com gavetas, onde se guardam objetos, que posteriormente poderão ser utilizados. Logo, o estudante poderá construir relações entre o que conhece e aquilo que se deseja aprender, que se apoia na teoria de Piaget (1995, p. 6) “[...] sobre todas as atividades cognitivas do sujeito (esquemas ou coordenações de ações, operações, estruturas etc.) para delas retirar certos caracteres e utilizá-los para outras finalidades (novas adaptações, novos problemas etc.) ”.

Quando o estudante, por meio de suas coordenações inferirem propriedades ao objeto, ele está num outro patamar, mais elevado que o primeiro, denominado por Piaget (1995) de abstração pseudoempírica, na taxonomia, acredita-se que ele esteja no nível compreensão ou aplicação. Fazendo o uso de exemplo na programação, é quando o estudante deve fazer um programa para dividir dois valores e cria todas as variáveis como inteiro, ao executar o mesmo o resultado não corresponde ao esperado (5 / 2 = 2). Com isso, o estudante faz reflexão, e por meio de pequenas mudanças (altera a variável do resultado para real) consegue atingir o resultado esperado, porém ele ainda depende do concreto para verificar a necessidade destas alterações. Segundo Piaget (1995, p.274-275) reflexão é o “[...] ato mental de reconstrução e reorganização sobre o patamar superior daquilo que foi assim transferido do inferior”.

Já os níveis análise, síntese e avaliação, requerem um nível de abstração mais elevado, podendo estar de acordo com a abstração reflexiva, pois requer que o estudante (sujeito) faça relações entre os conteúdos (objetos). Para Piaget a abstração reflexiva

> apoia-se sobre as formas e sobre todas as atividades cognitivas do sujeito (esquemas ou coordenações de ações, operações, estruturas, etc) para delas retirar certos caracteres e utilizá-los para outras finalidades [...] (PIAGET, 1995, p.6)

Logo, ao se pensar no ensino de programação, quando o estudante desenvolve um programa, demanda que o mesmo faça o uso de diversos conceitos de programação (estrutura de um programa, variáveis, tipos de dados, operações matemáticas, etc), sendo necessária a coordenação entre eles. Assim sendo, Piaget afirma que:

> todo reflexionamento de conteúdos (observáveis) supõe a intervenção de uma forma (reflexão), e os conteúdos assim transferidos exigem a construção de novas formas devidas à reflexão. Há, assim, pois uma alternância ininterrupta de reflexionamentos → reflexões → reflexionamentos; e (ou) de conteúdos → formas → conteúdos reelaborados → novas formas, etc., de domínios cada vez mais amplos, sem fim e, sobretudo, sem começo absoluto (PIAGET, 1995, p. 276-277).

É importante ressaltar que para cada conteúdo (objeto) de programação, o estudante (sujeito) perpassa pelos patamares da abstração, assim sendo, a cada novo conceito se fazem necessários exemplos concretos que integram aos esquemas anteriores o novo conhecimento. Lembrando que para "assimilar um objeto a um esquema é, pois, simultaneamente tender a satisfazer uma necessidade e conferir uma estrutura cognitiva à ação" (PIAGET, 1983, p.246). Acredita-se assim que os níveis criados por Bloom et al. (1972) podem auxiliar o professor na criação de seus materiais didáticos, desde seus conteúdos até a avaliação de seus estudantes, uma vez que se preocupa na construção do conhecimento de forma hierárquica a fim de auxiliar no desenvolvimento cognitivo do estudante. Na próxima seção é descrita cada nível, apresentando exemplos na área de programação.

5.2.1 Níveis hierárquicos da TOEDC

A TOEDC, desenvolvido por Bloom et al. (1972, p.6) possui "objetivos vinculados à memória [...] e ao desenvolvimento de capacidades e habilidades intelectuais". Organizado de forma hierárquica, possui seis níveis que são organizados de acordo com a complexidade, indo do mais simples aos mais complexos. Para melhor entendimento, as subseções a seguir apresentam cada uma delas.

5.2.1.1. Conhecimento

Refere-se à verificação da memorização dos conceitos, no qual o estudante deve adquirir e armazenar informações que serão utilizadas quando o mesmo necessitar. Neste nível é provável que o estudante sinta dificuldades no entendimento, uma vez que normalmente não estão familiarizados com o novo conteúdo, e lhes "pareçam complexos e abstratos e adquiram significações somente em relação com situações e problemas concretos" (BLOOM et al., 1972, p.62). Sendo assim, é

importante trazer exemplos concretos, sejam eles físicos ou que faça parte do cotidiano do estudante, a fim de que o mesmo se aproprie deste conhecimento.

Este primeiro nível é considerado o mais importante e o mais longo, já que vai servir de base para todos os outros níveis (BLOOM et al., 1972). Logo, é necessário que o professor não exceda na quantidade de informação, pois o estudante pode ter dificuldades em reter esse conhecimento, podendo resultar num aprendizado deficitário. Desta forma, para avaliar a capacidade de conhecimento, deve-se verificar se o estudante consegue reconhecer de forma precisa os elementos aprendidos quando soluciona as atividades propostas.

Assim, quando o professor ensina variável e tipos de dados, pode relacionar aos conhecimentos matemáticos dos estudantes, como o exemplo apresentado no Quadro 3.

Quadro 3 - Exemplo matemático já conhecido pelos estudantes

Em um cálculo matemático podemos somar bananas com maçãs? Não, pois deve-se respeitar cada tipo de informação, e somar apenas os iguais.

Fonte: A autora

Após fazer relações com as experiências dos estudantes, o professor poderá trazer os conceitos da linguagem de programação, lembrando em ensinar aquilo que é essencial para o momento. Ao criar as atividades para este nível, pode-se avaliar:

- Conhecimento das terminologias técnicas (Que tipo de dados se armazena numa variável cujo tipo de dado é um inteiro (`int`);
- Padrões de organização (Como se declara uma variável?); conhecimento de convenções (A variável ‘salario’ deve pertencer a qual tipo de dado?).

5.2.1.2. Compreensão

Neste nível o estudante deve conseguir entender os conceitos de forma literal, dentro de uma comunicação não muito explícita, utilizando os mesmos em contextos diferentes. É necessário, portanto, que o estudante interprete as informações, podendo ainda organizar suas ideias numa outra linguagem. Espera-se também, neste nível, que o estudante consiga ir além das informações dadas na atividade.

Para Bloom et al. (1972, p.175), esse nível “representa o nível mais baixo de entendimento [...] tal que o indivíduo conhece o que está sendo comunicado [...] sem necessariamente relacioná-la a outro material [...]”. Relacionando com a teoria de Piaget (1995), pode-se dizer que neste nível o estudante faz uma abstração pseudoempírica, no qual ele não apenas retira característica do objeto, mas consegue atribuir propriedades a ele de acordo com a coordenação das ações. Em outras palavras, o estudante utiliza seu conhecimento para resolver problemas semelhantes ao que ele já conhece.

Ao fazer o uso desse nível na programação, o professor pode questionar seus estudantes, fazendo com que eles interajam e pensem sobre o que está sendo tratado. Neste sentido, Bloom et al. (1972, p.76) explicam que este nível “refere-se àqueles *objetivos, comportamentos* ou *respostas* que representam um entendimento da mensagem literal contida em uma comunicação”. Utilizando-se do mesmo exemplo anterior, variável e tipos de dados, o professor pode verificar se o estudante atingiu esse nível por meio de atividades que envolvam interpretação (Reconhecer as variáveis necessárias para um determinado programa em um dado enunciado).

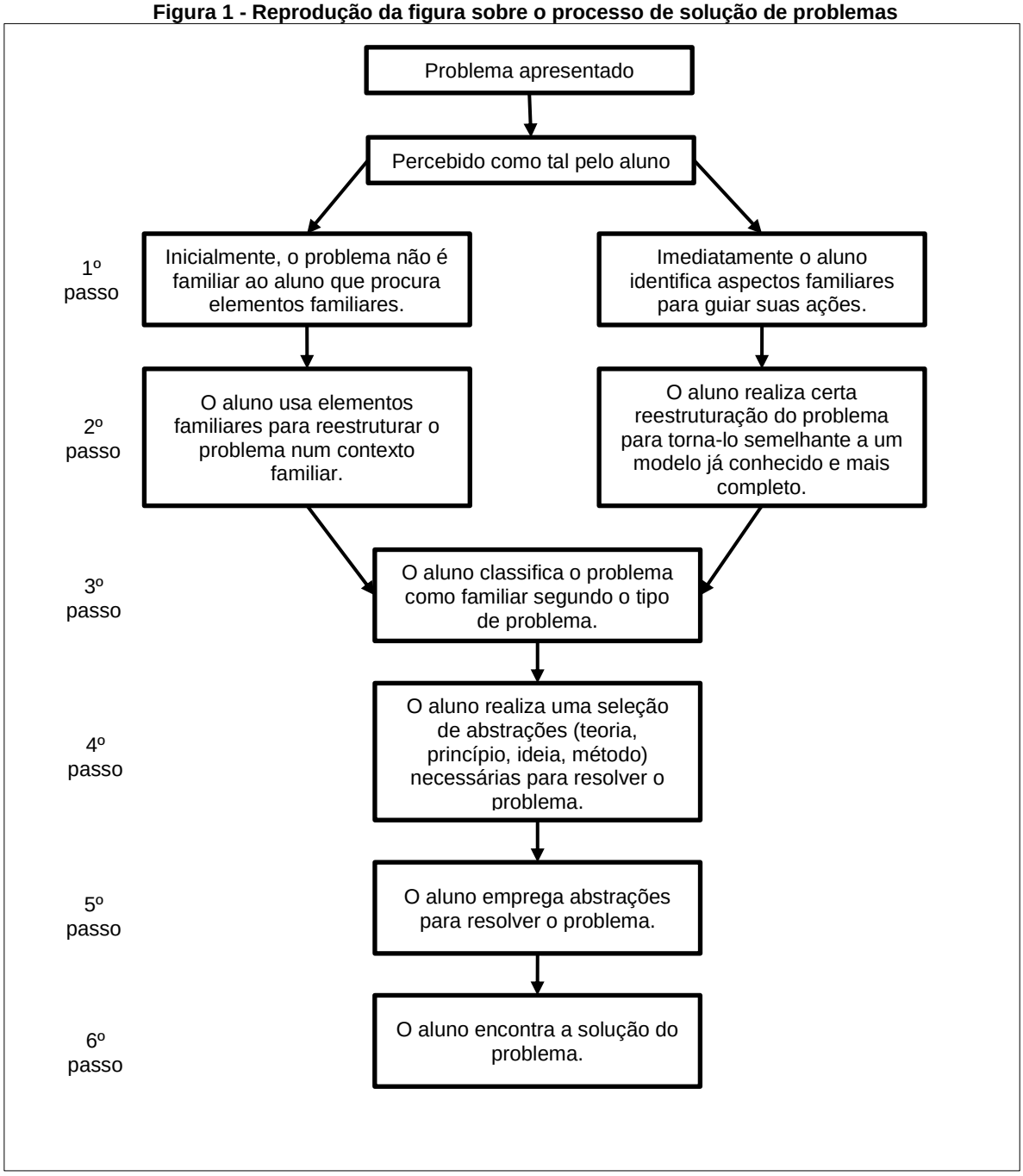
5.2.1.3. Aplicação

O nível de abstração a partir daqui aumenta consideravelmente, pois os estudantes devem demonstrar compreensão dos conceitos aprendidos. Consequentemente será necessário que os estudantes tenham prática em classificar e reestruturar situações, para que consiga aplicar a solução correta. De acordo com Bloom et al. (1972, p.103), “na aplicação, o aluno deve usar corretamente a abstração em uma situação na qual ela não estiver de modo algum especificada”.

Neste sentido, para que o estudante demonstre que atingiu o nível de aplicação, o mesmo deve ir além da compreensão. Enquanto que na compreensão o estudante simplesmente reconhece o conceito e demonstra seu uso quando necessário, na aplicação ele deve pôr em prática esses mesmos conceitos em situações novas. De acordo com a teoria de Piaget (1995), pode-se dizer que a partir desse nível o estudante começa a fazer reflexão, que se refere ao “[...] ato mental de reconstrução e reorganização sobre o patamar superior daquilo que foi assim transferido do inferior” (idem, p. 274-275).

Este nível é problemático, pois se deve ter o cuidado para que o estudante não seja treinado, ou seja, apenas ser reprodutor daquilo que em teoria aprendeu. Sendo assim, é esperado que o estudante na disciplina de LPI consiga desenvolver seus próprios blocos de código, e não apenas repetir aqueles apresentados pelo professor. Ao desenvolver atividades para este nível, o professor verificar o processo de solução do problema fazendo o uso do fluxograma apresentado na Figura 1.

Figura 1 - Reprodução da figura sobre o processo de solução de problemas



Fonte: Bloom et al. (1972, p. 104)

De acordo com Bloom et al. (1972, p.103), “se o processo se assemelha-se mais às conexões [...] 1 a 4 integram a ‘aplicação’, mas não fazem parte da ‘compreensão’ [...] melhor representada por um problema que inicia na etapa 5”. Logo, ao analisar uma atividade relacionada ao nível aplicação, o estudante deve apresentar a solução desde o 1º passo do fluxograma (Figura 1) até o 6º passo. Porém, se o mesmo apresentar a solução do problema iniciando no 5º passo (Figura 1), o mesmo não caracteriza, de acordo com Bloom et al. (1972), ter atingido o desenvolvimento cognitivo do nível de aplicação para aquele conteúdo.

Tratando-se de atividades de programação, um exemplo a ser usado para esse nível seria apresentar um problema ao estudante e verificar se ele consegue definir quais são as variáveis necessárias e seus tipos de dados para solucioná-lo. Pode-se dizer também, que é esperado nesse nível que o estudante consiga apresentar a solução das atividades estruturando um bloco de comandos referente ao conteúdo que está sendo ensinado no momento.

5.2.1.4. Análise

Como quarto nível tem-se a análise, considerada uma prévia da avaliação, é a habilidade em usar os conteúdos aprendidos, de maneira que o estudante percebe as relações internas existentes e como estão organizados esses conteúdos. Assim, o estudante utiliza essa percepção para comunicar o significado, servindo em si como técnica de comunicação capaz de estabelecer o resultado final de uma comunicação.

Segundo Bloom et al. (1972, p.123) a análise serve “[...] como uma ajuda para o alcance de maior compreensão, ou como uma etapa na avaliação do material. ”, sendo que o material é próprio conteúdo. Por isso, Bloom et al. (1972) salienta que a habilidade de analisar é objetivo de importância em qualquer área, sendo que, muitas vezes, os professores desejam essa habilidade para: identificar quais materiais são importantes, perceber as implicações não explicitamente formuladas, verificar as indicações sobre técnicas e recursos empregados.

Ademais, a análise é um nível mais complexo de habilidade que a compreensão, pois a compreensão está relacionada com o entendimento de um significado, enquanto que a análise lida com o entendimento e com a forma como essa comunicação é realizada. Por isso, se dois programas realizam a mesma

atividade, a habilidade de perceber que os programas fazem a mesma atividade por um estudante é chamada de compreensão, enquanto que analisar as relações entre as partes do programa e avaliar a forma como se dá a construção e a técnica utilizada em cada uma dessas partes é chamada de análise. Assim, é por meio da análise que o estudante poderá expressar opiniões sobre a excelência da comunicação, ou seja, do programa.

5.2.1.5. Síntese

Quinto nível da TOEDC, a partir daqui o estudante tem mais liberdade, podendo desenvolver uma conduta criadora. De acordo com Bloom et al. (1972, p. 143), “o produto da síntese é ainda singular ou único pela liberdade em que se concede ao indivíduo para propor suas ideais, [...] não está rigorosamente predeterminado pelos requisitos da tarefa”. É esperado ainda, que o estudante tenha a habilidade em combinar conteúdos, ou seja, unir partes, combinando-as para chegar ao resultado esperado.

Tratando-se de programação, no nível síntese, é esperado que o estudante consiga fazer um programa completo, fazendo o uso dos diversos conceitos aprendidos durante o semestre. Um exemplo simples seria um programa para somar dois valores, o estudante necessitaria conhecer e compreender, aplicar, analisar e sintetizar alguns conteúdos como: estrutura de um programa, variáveis, tipos de dados, operadores aritméticos e entrada e saída de dados. Com isso, aplicar os comandos necessários para este programa, analisando se os comandos escolhidos serão os mais apropriados, e assim sintetizar seu programa.

Sintetizar o programa significa a forma no qual o estudante irá fazer seu programa, uma vez que isso depende diretamente de sua lógica. Caso o estudante não tenha atingido o nível síntese, ele estará apenas replicando aquilo que o professor deu como exemplo, não fazendo uso do comportamento criador esperado nesta.

Como sexto e último nível, é apresentada a avaliação, que apesar de estar no topo hierárquico, Bloom et al. (1972) acreditam que ela não é, necessariamente, o estágio final do pensamento, pode ser que seja a preparação para o aprendizado de um novo conhecimento. Logo, ela foi colocada no topo uma vez que combina todos os outros níveis, e nele é esperado que o estudante julgue levando em conta critérios

e padrões determinados por ele ou que lhes são dados. Este nível é detalhado no próximo tópico.

5.2.1.6. Avaliação

Considerado como estágio final, Bloom et al. (1972, p. 157) definem esta camada “como o processo de julgamentos acerca do valor de ideias, trabalhos, soluções, métodos, materiais, [...] realizados com um determinado propósito”. Neste momento, é esperado que o estudante tenha se apropriado do conteúdo a tal ponto que consiga julgar os resultados de forma consciente, tendo uma percepção integral de todos conceitos em que se apoia (BLOOM et al., 1972). Vale ressaltar que esse julgamento não se trata de uma simples opinião, pois esta, de acordo com Bloom et al. (1972, p.158), “não chega a ser totalmente consciente, e o indivíduo pode não ter uma percepção integral dos fundamentos ou bases em que as apoia”.

Para que o estudante possa avaliar devem haver critérios, sejam eles definidos pelo próprio estudante, ou por outro indivíduo, para um julgamento tanto quantitativo quanto qualitativo (BLOOM et al., 1972). Combinando e envolvendo todos os outros níveis, a camada de avaliação não é necessariamente a camada final, ela pode ser a preparação para um novo conhecimento. A TOEDC trata-se da construção do conhecimento, de forma que os níveis não são isolados, inter-relacionam-se, assim como a abstração reflexionante que “é um processo que permite construir estruturas novas, em virtude da reorganização de elementos tirados de estruturas anteriores [...]” (PIAGET, 1995, p. 193).

Assim sendo, pode-se dizer que o estudante ao avaliar qualitativamente ou quantitativamente de forma consciente, de acordo com a TOEDC, este estará em um patamar de abstração bastante elevado, podendo ser caracterizado como abstração refletida. Piaget (1995, p.205) define a abstração refletida como sendo “resultado da abstração reflexionante, assim que se torna consciente”. Neste sentido, o estudante é capaz de desenvolver as atividades sem a necessidade do concreto, uma vez que o mesmo já se apropriou das teorias e conhece o resultado das mesmas (VIEGAS, 2015).

Ao desenvolver um programa, é possível verificar se o estudante atingiu o nível avaliação ao verificar se o mesmo consegue: indicar problemas na lógica de um

código (utilizar uma variável antes de declará-la); comparar códigos julgando o mais adequado de acordo com as regras de programação. Para isso, o professor pode apresentar na atividade um enunciado de um programa e possíveis soluções para o mesmo, assim, o estudante deve avaliar e apontar a solução mais adequada, ou a solução correta. Na sequência, apresentar-se-á os conceitos de programação utilizadas nos estudos de caso da presente pesquisa.

5.3 INTRODUÇÃO AOS CONCEITOS DE PROGRAMAÇÃO

Os conceitos de programação abordados nesta seção são referentes a linguagem de programação C, uma vez que esta é a linguagem utilizada na disciplina de Linguagem de Programação I do Curso Superior de Sistemas para Internet, disciplina esta que faz parte da presente pesquisa. Através dos dados levantados no Projeto Pedagógico de Curso (IFRS, 2013) do curso em questão do IFRS Campus Porto Alegre, foram verificados quais conteúdos são trabalhados na disciplina para que estes fossem aqui apresentados. Neste sentido, a disciplina de Linguagem de Programação I tem como objetivo ensinar:

> Conceitos de variáveis, variáveis homogêneas (vetores e matrizes) e variáveis heterogêneas (registros). Operadores e expressões matemáticas e lógicas. Estruturas de controle de programação. Funções, procedimentos, variáveis locais e globais, passagem de parâmetros por valor e por referência e tratamento de arquivos (IFRS, 2013, p.24).

Nesses objetivos também é definida a linguagem de programação a ser utilizada: C ANSI. De acordo com Deitel (2011), a linguagem C é uma das mais populares em desenvolvimento de software e foi padronizada em 1989 pela ANSI através da *International Standards Organization* (ISO). Assim como muitas linguagens de programação, a C é considerada de alto nível, ou seja, “permitem aos programadores escrever instruções que se parecem com o idioma inglês comum e contêm as notações matemáticas normalmente usadas” (DEITEL, 2011, p. 12). Outra característica dessa linguagem é a programação estruturada, que “produz programas mais fáceis de entender (do que a programação não-estruturada) e, portanto, mais fáceis de testar, depurar, modificar e até de se provar sua correção do ponto de vista matemático” (idem, p. 162).

Tendo em vista a contextualização da linguagem C ANSI, esta seção não abordará todos os conteúdos previstos nos objetivos da disciplina, mas àqueles que fazem parte da pesquisa. Esses conteúdos foram selecionados através da pesquisa documental apresentada no Capítulo 6, seção 6.1, que traça os conteúdos ensinados nas DIP, nos cursos superiores de TI do IFRS. Neste sentido, os conteúdos abordados nesta seção serão: tipo de dados, variáveis, entrada e saída de dados, expressões aritméticas, operadores lógicos, estruturas de decisão, estruturas de repetição, vetores, matrizes e definição de funções.

Os tipos de dados estão diretamente ligados com as variáveis, pois estas são espaços na memória que armazenam os dados de um programa e ao criá-las é necessário definir seu tipo. Isso é necessário para informar ao computador como o dado da variável deverá ser manipulado, por exemplo, ao definir uma variável com o tipo inteiro, o computador saberá que podem ser realizadas operações matemáticas inteiras sobre o valor armazenado. Logo, para fazer um programa que calcule a soma de dois valores, o estudante precisa utilizar variáveis numéricas, e ainda levar em conta se esses valores serão inteiros ou reais. Caso o estudante declare a variável como inteira, esta não poderá armazenar valores reais, porém se o mesmo declarar a variável como real, esta poderá armazenar valores inteiros. Neste sentido, para cada tipo de programa, o estudante deve analisar o mesmo e definir as variáveis e seus tipos de acordo com a solução do problema. No Quadro 4 são apresentados alguns tipos de dados, juntamente com a descrição e exemplos de dados que poder ser armazenados.

Quadro 4 - Tipos de dados

Tipo	Descrição	Exemplo
Inteiro	Valor numérico sem ponto decimal	23, 89, -10, 0, 450
Real	Valor numérico com ponto decimal	23.7, 45.8, 34.3, 20.0
Caractere	Um ou mais caracteres, que podem ser letras, números ou caracteres especiais	ana, joão, “romeu e julieta”, rh@hotmail.com
Lógico	Valor lógico	verdadeiro ou falso

Fonte: Okuyama *et al.* (2014, p. 47)

Essas informações não são visualizadas pelos usuários dos programas, pois o código fonte criado pelo programador é compilado e transformado em um programa. Logo, para que haja interação entre o usuário e o sistema computacional é necessário programar a entrada e saída de dados, que são funções na linguagem C ANSI. Para a saída de dados é utilizada a função `printf`, que mostrará na tela do computador as informações nela contida. No Quadro 5 é apresentado um exemplo de programa

na linguagem C ANSI a fim de ilustrar a utilização do comando de saída `printf`. O Quadro 5 está dividido em duas partes (código e resultado na tela) no qual a última coluna representa o que seria mostrado na tela para o usuário do programa, e na primeira coluna são os comandos necessários para que apresente as informações na tela.

Quadro 5 - Exemplo programa utilizando `printf`

Código	Resultado na tela

`/* Primeiro programa em C */`
`main( )`
`{`
`printf  ("Bem-vindo ao C!\n");`
`}`
	Bem-vindo ao C!

Fonte: Deitel (2011, p. 37)

A entrada de dados é utilizada para ler informações digitadas pelo usuário através da função `scanf`, e para fazer o uso desta informação, é necessário armazená-la em uma variável. Observe que apesar de se tratar de um novo conceito, é necessário que o estudante saiba variáveis para poder desenvolver o programa. O Quadro 6 apresenta uma possível solução para um programa que realize a soma de dois valores inteiros digitados pelo usuário.

Quadro 6 - Exemplo programa utilizando *scanf*

Código	Dados digitados	Resultado na tela
`main()`		
`{`		
`int inteiro1,  inteiro2,  soma;`		
`printf("Entre com o primeiro inteiro:");`		Entre com o primeiro inteiro:
`scanf("%d", &inteiro1);`	45	
`printf("Digite o segundo valor:");`		Digite o segundo valor:
`scanf("%d", &inteiro2);`	72	
`soma = inteiro1 + inteiro2;`		
`printf("A soma e %d/n",  soma);`		A soma e 117
`}`		

Fonte: Deitel (2011)

Analisando o Quadro 6 é possível observar que além do conceito de variável, o estudante precisa saber como utilizar a função de saída de dados. Além disso, o exemplo trata da soma de dois valores inteiros, que se trata do conteúdo expressões aritméticas. Esse conceito segue a lógica matemática e “devem ser escritas no formato linear [...], expressões como "a dividido por b" devem ser escritas como a/b

de forma que todos os operadores e operandos apareçam em uma única linha" (DEITEL, 2011, p.50).

No Quadro 7 são apresentadas as regras de precedência utilizados na linguagem de programação C ANSI. Calculando da esquerda para direita, as regras de precedência nesta linguagem seguem as mesmas regras matemáticas, no qual "a + b / c" é diferente de "(a + b) / c".

Quadro 7 – Regras de precedência na linguagem C ANSI

Operador	Operação	Ordem de cálculo (precedência)
( )	Parênteses	Calculado em primeiro lugar. Se houver parênteses aninhados, a expressão dentro do par de parênteses mais interno é calculada em primeiro lugar. No caso de vários pares de parênteses "no mesmo nível" (i.e., que não estejam aninhados), eles são calculados dão esquerda para a direita.
* / %	Multiplicação Divisão Resto(módulo)	Calculados em segundo lugar. No caso de vários operadores, eles são calculados da esquerda para direita.
+ -	Adição Subtração	Calculados por último. No caso de vários operadores, eles são calculados da esquerda para direita

Fonte: Deitel (2011, p.52)

Os conceitos de operadores lógicos e os operadores relacionais são utilizados tanto em estruturas de decisão, quanto nas estruturas de repetição, pois servem para fazer as comparações necessárias para estas estruturas. No Quadro 8 é possível observar os operadores relacionais utilizados na linguagem de programação C ANSI. Além dos operadores relacionais, é possível utilizar os operadores lógicos para fazer mais de uma comparação, e são eles: "`&&`" (conjunção), "`||`" (disjunção) e "`!`" (negação).

Quadro 8 - Operadores relacionais

Operador	Tipo	Exemplo
>	Maior	a>0
<	Menor	a<0
>=	Maior igual	a>=0
<=	Menor igual	a<=10
!=	Diferente	a!=11
==	Igual	a==0

Fonte: Okuyama (2014, p.117)

As estruturas de decisão (`if...else, switch`), como o próprio nome diz, são utilizadas para tomar uma decisão: "[...] em um programa, por exemplo, para determinar se o grau de uma pessoa em uma prova é maior ou igual a 60 e, se for

imprimir a mensagem ‘Parabéns! Você passou.’” (DEITEL, 2011, p.54). O código para representar essa tomada de decisão pode ser visualizado no Código 1, acrescido de uma possível decisão caso a nota seja menor que 60.

Código 1 - Código de estrutura de decisão

```
if (media >= 60){
      printf(“Parabéns! Você passou.”);
}
else{
      printf(“Você não passou.”);
}
```

Fonte: A autora

Observando o Código 1, é possível identificar além dos conceitos de estrutura de decisão, os conceitos de saída de dados (`printf`) e variáveis (`media`), o que deixa explicito que os conteúdos de programação são cumulativos. Neste sentido, tanto a TOEDC de Bloom et al., quanto a TC de Piaget são indicadas para a prática docente neste tipo de ensino. Na sequência têm-se as estruturas de repetição, que “possibilitam repetir determinados trechos de códigos para solução de problemas que exijam repetição dos processos” (OKUYAMA et al., 2014, p.124).

Logo, para imprimir na tela uma sequência de números que inicie em um e termine em dez, é indicado fazer o uso das estruturas de repetição. No Código 2 é apresentada uma possível solução para o exemplo em questão, note que fazendo o uso do comando `for` reduz o número de linhas de comandos, pois se não o utilizasse, o programa teria que repetir o comando `printf` dez vezes.

Código 2 - Comando for com incremento

```
int main(int argc, char** argv){
  int i;
  for (i = 1; i <= 10; i++){
     printf(“%d\n” , i);
  }
  return (EXIT_SUCCESS);
}
```

Fonte: A autora

Com isso, pode-se dizer que na medida em que o estudante aprende novos conceitos, ele aprimora sua programação dando a seus programas eficiência e manutenibilidade. Neste sentido, têm-se os conceitos de vetores e matrizes que, segundo Okuyama et al. (2014, p.130), “são tipos de dados uniformes e que possuem uma quantidade predeterminada de elementos referenciados por um mesmo nome”.

Em outras palavras, é como criar uma única variável e poder armazenar nela mais de um dado, pois esta variável possui índices que controla esse armazenamento.

A diferença entre vetor e matriz está em suas dimensões, ou seja, a matriz possui “n” linhas e colunas, já o vetor possui apenas uma dimensão (uma linha com “n” colunas). Sendo assim, esses conceitos requerem um nível de abstração superior, uma vez que o estudante terá que “imaginar” como esses dados são guardados na memória do computador. Caso o estudante tenha que desenvolver um programa que leia quatro valores inteiros do usuário, utilizando esse novo conceito, ele reduz o número de linhas de código, bem como facilitar a alteração do programa.

No Código 3 observa-se uma possível solução para o programa mencionado no parágrafo anterior, sem o uso de vetor. Note que o mesmo repete várias vezes o comando em destaque, e também o excesso de variáveis (`valor1, valor2, valor3, valor4`).

Código 3 - Exemplo sem vetor

```
int main(int agrc, char** argv){
   int valor1, valor2, valor3, valor4;
   printf(“Digite um valor inteiro: ”);
   scanf(“%d”, &valor1);
   printf(“Digite um valor inteiro: ”);
   scanf(“%d”, &valor2);
   printf(“Digite um valor inteiro: ”);
   scanf(“%d”, &valor3);
   printf(“Digite um valor inteiro: ”);
   scanf(“%d”, &valor4);
   return (EXIT_SUCCESS);
}
```

Fonte: A autora

Ao comparar o Código 3 com Código 4, fica bastante visível que o uso do vetor permite a redução de linhas de código, deixando-o mais legível e simples.

Código 4 - Exemplo com vetor

```
int main(int agrc, char** argv){
    int valor [4];
    int i;
    for (i = 0; i < 4; i++){
      printf(“Digite um valor inteiro: ”);
      scanf(“%d”, &valor[ i ]);
    }
    return (EXIT_SUCCESS);
}
```

Fonte: A autora

Com o intuito de deixar o código ainda mais legível e possibilitar o reuso de código, tem-se o conceito de funções, que para Okuyama et al. (2014, p.139) “são blocos de códigos de um programa que podem ser utilizados várias vezes e, se organizados corretamente, podem ser utilizados em diversos programas”. Assim sendo, com os conceitos apresentados nessa seção, acredita-se que o estudante terá base para construir pequenos sistemas computacionais e atingir os objetivos apresentados na ementa da disciplina (IFRS, 2013, p.24) que é “desenvolver o raciocínio na elaboração de soluções à problemas de solução algorítmica”. Na próxima seção serão abordados os desafios e dificuldades encontrados por professores e estudantes nas disciplinas introdutórias de programação.

5.4 ENSINO-APRENDIZAGEM DE PROGRAMAÇÃO: DESAFIOS E DIFICULDADES

Ao iniciar um curso, seja técnico ou superior, os estudantes devem se matricular em todas as disciplinas ofertadas no primeiro semestre, neste sentido, estudam CIP. De acordo com Lahtinen (2005) a disciplina de programação é considerada uma das disciplinas mais desafiadoras dos cursos de TI, uma vez que requer compreensão correta de conceitos abstratos. Normalmente esses estudantes nunca tiveram contato com a programação, e são denominados “novatos em programação” (SEVELLA et al., 2013).

Segundo Winslow (1996), um estudante pode levar em torno de dez anos para se tornar um programador especialista. Acredita-se que este tempo é tão longo devido à complexidade que é programar, uma vez que demanda múltiplas habilidades e conhecimentos: desde memorização e compreensão, até o pensamento lógico e capacidade de abstração de problemas. Além disso, são necessários fazer uso de ferramentas de desenvolvimento de programação, que na sua maioria são complexas e mostram os erros apenas em inglês.

Nesse sentido, o estudante terá que ter conhecimentos básicos da língua inglesa para poder diferenciar os erros emitidos pelo compilador, bem como os códigos da linguagem de programação. Para ilustrar, no Quadro 9 tem-se o exemplo de uma atividade simples de somar dois valores, bem como a resolução da atividade de forma matemática e também na Linguagem de Programação C.

Quadro 9 - Exemplo de atividade para somar dois valores

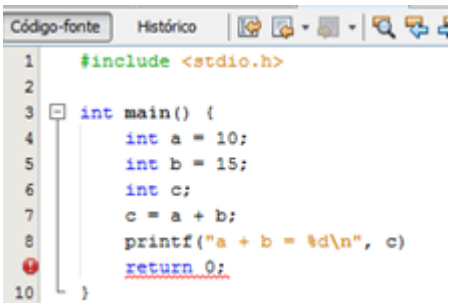
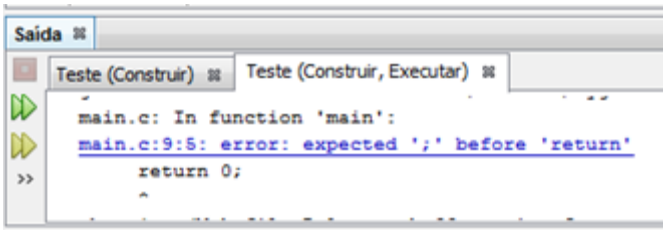
Sabendo que “a” vale 10 e “b” vale 15, mostre o resultado da soma dos dois valores:	
Matematicamente	Linguagem de programação C

x = a + b
x = 10 + 15
x = 25
	`#include <stdio.h>`
	`int main(){`
	`  int a=10;`
	`  int b=15;`
	`  int c;`
	`  c=a+b;`
	`  printf(“a + b = %d”, c);`
	`  return 0;`
	`}`

Fonte: A autora

No Quadro 10 é apresentado o código com erro, para mesma atividade do exemplo do Quadro 9, o erro gerado pelo compilador, e também a tradução da mensagem gerada pelo compilador. É importante observar que mesmo que o estudante consiga traduzir o erro, se ele não tiver domínio dos conceitos de programação, não irá conseguir compreendê-lo. No exemplo do Quadro 10, o erro é gerado devido a falta do símbolo de ponto e vírgula “;” (linha 9). Para solucionar esse problema, basta o estudante inserir o símbolo faltante.

Quadro 10 - Exemplo de erro de compilação

Código fonte	
Erro	
Tradução	main.c: Na função ‘main’: main.c: 9:5: erro: esperado ‘;’ antes ‘return’ return 0; ^

Fonte: A autora

Esse exemplo de erro é bastante comum ocorrer, porém é apenas um de muitos outros que o estudante se depara ao programar. Muitos desses erros são oriundos de uma programação mal formulada e/ou de uma lógica mal estruturada; já outros são da falta de conhecimento das regras de programação e/ou de como cada comando funciona. Apesar de cada instituição de curso superior ter seu próprio Projeto Pedagógico de Curso, incluindo as ementas para cada disciplina, um estudante de curso superior em Sistemas para Internet, na disciplina de Programação (em linguagem C, neste caso), normalmente precisa dominar alguns conhecimentos para ser aprovado na disciplina, como: tipo de dados, variáveis, entrada e saída de dados, expressões aritméticas, operadores lógicos, estruturas de decisão, estruturas de repetição, vetores, matrizes e definição de funções.

Outro ponto importante, e que muitas vezes acaba por tornar a aprendizagem de programação um pouco mais complexa, é que para cada um dos itens listados anteriormente, é necessário que o estudante compreenda tanto o conceito, quanto a aplicação do mesmo no código. Além disso, vale ressaltar que na programação um novo conceito é, normalmente, dependente do conceito anterior, no qual afirma Renumol et al. (2010, p.17) “que a programação é uma interação entre processos cognitivos menores e maiores e precisa de várias habilidades cognitivas”. Sendo assim, a aprendizagem de uma nova linguagem (comandos de programação), o conteúdo cumulativo, o uso de uma língua estrangeira (língua inglesa), as necessidades de raciocínio lógico, utilização de ferramentas complexas, entre outros aspectos, tornam o ensino-aprendizagem de programação uma tarefa bastante difícil.

Nesse contexto, Sevella et al. (2013), fizeram uma pesquisa com o intuito de verificar quais são as principais barreiras enfrentadas pelos estudantes ao aprender os conceitos de programação. A hipótese dos autores era de que as dificuldades nas quais os estudantes enfrentam seriam devido a barreiras conceituais. Porém, após verificação e análise dos resultados, os autores afirmaram que as dificuldades se encontram na construção dos códigos de programação.

Durante a pesquisa de Sevella et al. (2013), foi observado que os participantes tiveram dificuldade para aplicação dos códigos básicos de programação como: lógica para solução dos problemas, entrada e saída de dados, estruturas condicionais, estruturas de repetição, etc. O mesmo foi observado por Lahtinen (2005, p.17), no qual afirma que “o maior problema dos programadores novatos não é compreensão de conceitos básicos, mas sim aprender a aplicá-los” e acrescenta que “quanto mais

prática e concreta for o problema e o material de aprendizagem, mais aprendizagem se terá" (idem).

Neste sentido, compreender o comportamento cognitivo dos estudantes durante a aprendizagem de CIP pode auxiliar os professores a aprimorar suas estratégias pedagógicas (RENUMOL et al., 2010). Na seção seguinte apresentar-se-á alguns conceitos sobre Redes Bayesianas que foram necessários para fundamentar a sistematização da análise das atividades.

5.5 REDES BAYESIANAS

Oriunda do Teorema de Bayes, as (RB) são comumente utilizadas para propor probabilidades estatísticas quando envolvem dados imprecisos. Em seu teorema Bayes relacionou causa e efeito com o intuito de compreender o efeito e assim conhecer a probabilidade da causa (LUGER, 2013).

Logo, o Teorema de Bayes permite que seja calculada a probabilidade de uma hipótese h_i ser verdadeira ao ser observado a existência de uma evidência em particular. Por se tratar de uma inferência estatística, à medida que se observa evidências num caso específico, pode-se atualizar a crença sobre a hipótese, ou seja, recalcula-se a probabilidade da hipótese h_i do conjunto H dada à evidência E fazendo uso da seguinte equação:

$$p(h_i|E) = \frac{p(E\ |h_i)\ .p(h_i)}{\sum_{k=1}^{n} p(E|h_k)\ .p(h_k)},$$

sendo que $p(h_i|E)$ é a probabilidade de a hipótese h_i seja verdadeira dada a evidência E, $p(E\ |h_i)$ é a probabilidade de observar a evidência E quando h_i é verdadeira, $p(h_i)$ é a probabilidade de que h_i seja verdadeira em geral, n é número de hipóteses possíveis.

Para que seja possível fazer uso do Teorema de Bayes: "primeiro, todas as probabilidades nas relações da evidência com as diversas hipóteses devem ser conhecidas, bem como as relações probabilísticas entre as partes da evidência" (LUGER, 2013). Além disso, Luger (2013) reforça que outro processo ainda mais difícil

de determinar é que “todas as relações entre evidência e hipóteses devem ser estimadas ou amostradas empiricamente”.

Assim, as RB podem ter como representação gráfica uma estrutura de grafo acíclico e direcionado, sendo assim chamado de Rede Bayesiana de Crenças (LUGER, 2013). Nesse grafo, uma variável é constituída como um nó, e os arcos constituem a influência causal do pai para com o filho. Associada a cada variável tem-se uma tabela de probabilidades, sendo que os valores das variáveis podem ser representados por meio de escalas discretas ou contínuas. Conforme Korb e Nicholson (2004), as RB são direcionadas ao tratamento de variáveis discretas, pois as variáveis contínuas podem ser facilmente discretizadas por meio de categorizações. Além disso, Neapolitan (2004) informa que a utilização de variáveis contínuas é realizada quando a quantidade de categorias (necessárias para devida representação do estado da variável) torna-se muito grande.

Portanto, o processo de criação de uma rede bayesiana pode começar pela identificação de hipóteses e pelo particionamento do conjunto de evidências. Em seguida, estimam-se as probabilidades das relações entre hipóteses e evidências. Ao final, faz-se um conjunto de testes para validar a rede criada. Contudo, observa-se que a identificação das hipóteses e das evidências deve ser feito por especialistas no assunto, pois demanda conhecimento prévio sobre o assunto em questão.

Já a estrutura da RB pode ser feita manualmente por meio de especialistas, pode ser aprendida por meio de algoritmos de aprendizado ou pela combinação de ambas. Para os algoritmos poderem aprender a estrutura da RB, é necessário a utilização de um conjunto de dados, em que cada entrada representa um caso contendo os valores para cada uma das variáveis.

Dentre os algoritmos existentes para aprendizado de estrutura destacam-se:

- Bayesian Search: proposto por Cooper (1992), faz uso de uma heurística do tipo subida em encosta.
- PC: proposto por Spirtes (1991), este algoritmo permite a utilização de variáveis discretas e contínuas.
- Greedy Thick Thinning: proposto por Cheng (1998), o algoritmo é inspirado no Bayesian Search.
- Tree Augmented Naive Bayes: proposto por Friedman (1997), foi inspirado no Bayesian Search.

Além da estimação da estrutura da RB, também é necessário realizar a estimação dos parâmetros da rede, ou seja, estimar as probabilidades de cada nó da rede. Para calcular essas probabilidades, são utilizados o mesmo conjunto de dados da etapa de estimação de estrutura. No caso dos dados de entrada estarem completos, pode-se calcular os parâmetros por duas soluções: estimação por verossimilhança ou por estimação bayesiana. Caso os dados estejam incompletos, um algoritmo de *missing* é utilizado para estimar os valores faltantes, isso é realizado por meio da observação dos dados existentes.

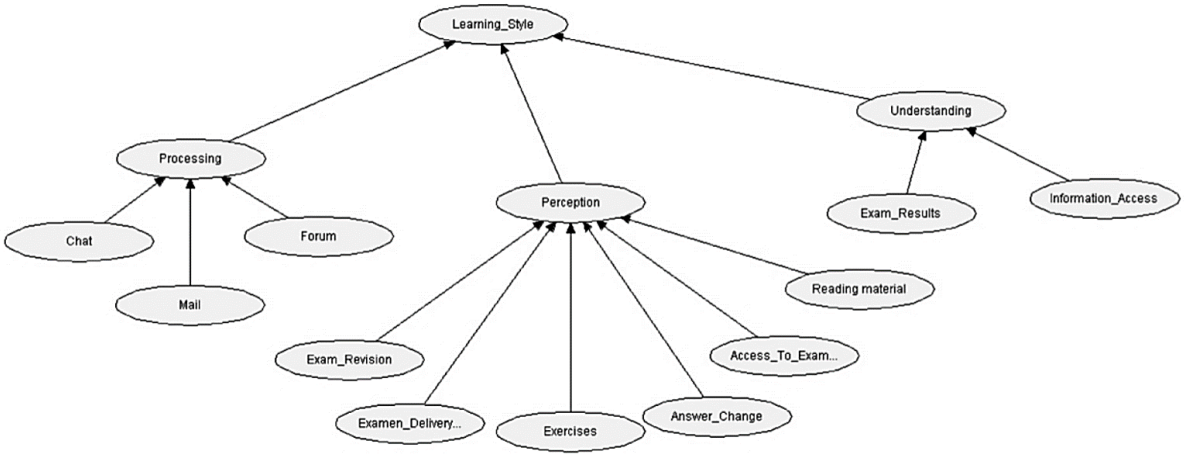
Posterior a construção da rede, passa-se a inferência de probabilística, a qual permite que o usuário especifique evidências para as variáveis da rede. Assim, os algoritmos de inferências podem atualizar as probabilidades dos nós filhos.

Considerando-se que a RB demanda que as probabilidades sejam completas e atualizadas, em áreas do conhecimento em que novas descobertas são feitas constantemente, atualizar as relações entre hipóteses e evidências pode ser um problema intratável (LUGER, 2013). Isso se deve ao fato de que, para atualizar essas probabilidades, despende-se muito tempo na geração de amostragens estatísticas e empíricas. Já nos casos em que o conjunto de dados a serem gerados é muito extenso, a geração estatística pode se tornar impeditiva.

5.5.1 Uso da RB na educação

Fazendo o uso dos estios de aprendizagem, Garcia et al. (2007) modelaram uma RB para a identificação do estilo de aprendizagem de estudantes através de dados coletados em um sistema web para educação. A estrutura desse trabalho pode ser verificada na Figura 2. Os fatores utilizados por Garcia et al. (2007) para determinar o estilo de aprendizagem do discente foram: (i) se o estudante revisa as atividades e quanto tempo leva nessa revisão; (ii) quanto tempo o estudante leva para terminar uma atividade e entregá-la; (iii) quantidade de vezes que o estudante muda suas respostas em uma atividade; (iv) tipo de material de leitura que o estudante tem preferência (concreto ou abstrato); (v)número de exemplos de um dado tópico que o estudante lê; e, por fim, (vi) número de exercícios que um estudante faz de um determinado tópico.

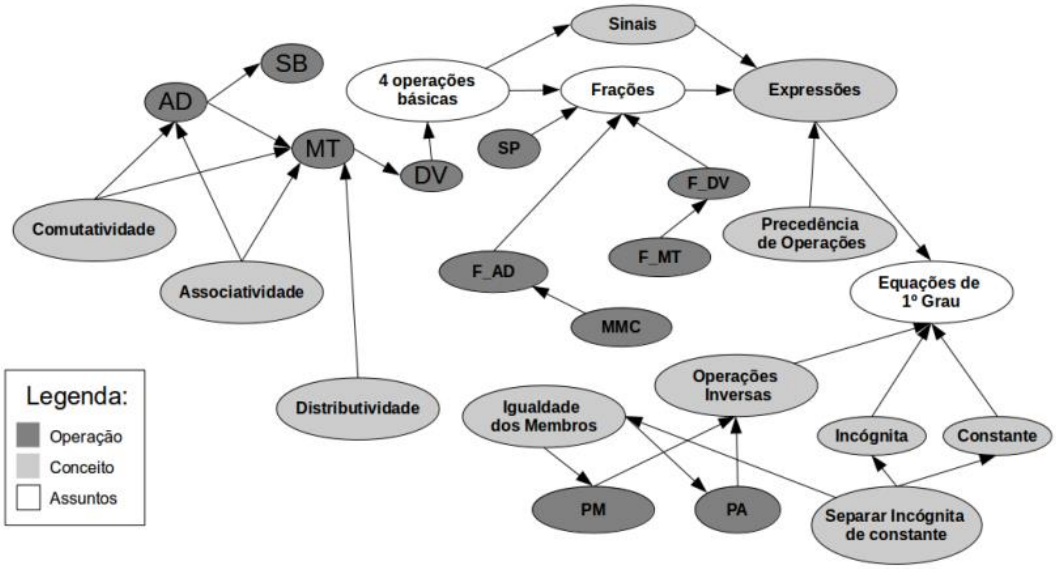
Figura 2 - Estrutura Rede Bayesiana desenvolvida por Garcia et al. (2007)



Fonte: Garcia et al. (2007, p. 800)

Seffrin e Jaques (2015), professores de matemática criaram um mapa conceitual para posteriormente criar a RB. O intuito desta era de avaliar o conhecimento algébrico dos estudantes, e sua estrutura pode ser observada na Figura 3.

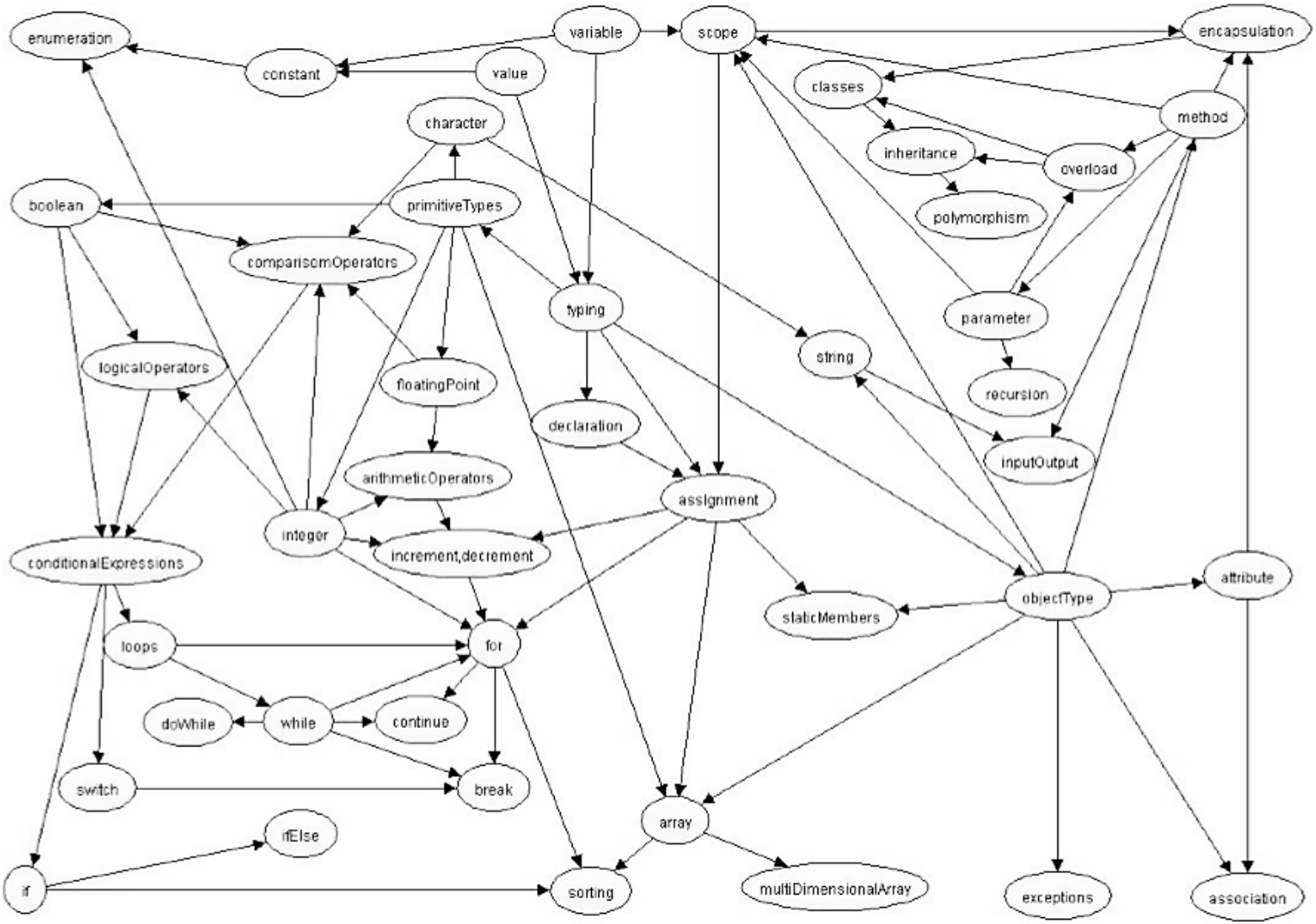
Figura 3 - Estrutura Rede Bayesiana desenvolvida por Seffrein e Jaques



Fonte: Seffrin e Jaques (2015, p.987)

Já no trabalho de Vier et al. (2015), a RB foi construída por meio de entrevistas com professores com experiência no ensino de programação. Esta RB levou em conta os componentes curriculares das disciplinas introdutórias de programação, bem como os conceitos vistos na disciplina subsequente, nos quais foram utilizados para definir os nós da rede. A proposta desta RB foi a de construir um modelo a respeito dos conhecimentos dos aprendizes, a mesma pode ser visualizada na Figura 4.

Figura 4 - Estrutura Rede Bayesiana proposta por Vier *et al.* (2015)



Fonte: Vier *et al.* (2015, p.54)

No capítulo seguinte apresenta-se a análise de dados gerados e coletados por meio do levantamento bibliográfico e documental, bem como no estudo de caso. Estes dados foram analisados de forma mista, ou seja, qualitativamente e quantitativamente.

6 ANÁLISE DE DADOS

A presente pesquisa utiliza a abordagem mista para análise de dados, que segundo Creswell (2010), é aquela que associa as formas qualitativa e quantitativa.

6.1 CURSOS DE TI E A DISCIPLINA DE LINGUAGEM DE PROGRAMAÇÃO I

Buscando conhecer os cursos Superiores de TI existentes nos Campi do IFRS, as metodologias que utilizam, e principalmente os conteúdos trabalhados nas disciplinas de Programação no primeiro semestre dos cursos, realizou-se a pesquisa documental. Através dos dados coletados nos sites de todos os campi[9], bem como do IFRS[10], buscou-se desenvolver uma proposta pedagógica condizente com a necessidade da disciplina de Linguagem de Programação I, necessariamente aquelas que ensinam linguagem de programação C.

Até o momento, primeiro semestre de 2017, o IFRS possui 17 Campi, sendo 12 já implantados: Bento Gonçalves, Canoas, Caxias do Sul, Erechim, Farroupilha, Feliz, Ibirubá, Osório, Porto Alegre, Restinga, Rio Grande e Sertão. Os outros 5 campi possuem cursos, porém encontram-se em implantação, que são: Alvorada, Rolante, Vacaria, Veranópolis e Viamão.

Foi possível verificar, de forma geral, que os campi possuem cursos de TI de nível médio nas modalidades: integrado, concomitante e subsequente. Possuem também cursos de nível superior e um mestrado profissionalizante. Uma vez que a criação dos IF tem o intuito da verticalização dos cursos (IFRS, 2015), é possível observar que a rede tenta atender esse requisito.

---

[9] Bento Gonçalves – http://www.bento.ifrs.edu.br
Canoas – http://www.canoas.ifrs.edu.br
Caxias do Sul – http://www.caxias.ifrs.edu.br
Erechim – http://www.erechim.ifrs.edu.br
Farroupilha – http://www.farroupilha.ifrs.edu.br
Feliz – http://www.feliz.ifrs.edu.br
Ibirubá – http://www.ibiruba.ifrs.edu.br
Osório – http://www.osorio.ifrs.edu.br
Porto Alegre – http://www.poa.ifrs.edu.br
Restinga – http://www.restinga.ifrs.edu.br
Rio Grande – http://www.riogrande.ifrs.edu.br
Sertão – http://www.sertao.ifrs.edu.br
Alvorada, Rolante, Vacaria, Veranópolis, Viamão – http://www.expansao.ifrs.edu.br
[10] http://www.ifrs.edu.br

No Quadro 11 apresenta-se os 18 cursos técnicos, existentes nos Campi do IFRS, de acordo com suas modalidades.

Quadro 11 - Levantamento dos cursos de técnicos dos Campi do IFRS

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL				
Campi		**Técnico**		
		Integrado	**Concomitante**	**Subsequente**
1	Bento Gonçalves	Informática para internet	-	-
2	Canoas	Informática	-	-
3	Caxias do Sul	-	-	-
4	Erechim	-	Informática	-
5	Farroupilha	Informática	-	-
6	Feliz	Informática	-	-
7	Ibirubá	Informática	-	-
8	Osório	Informática	-	Informática para internet
9	Porto Alegre	Redes	-	-
10	Restinga	Informática para internet	Redes	Redes
11	Rio Grande	Informática para internet	-	-
12	Sertão	Manutenção e suporte em Informática	Manutenção e suporte em Informática	-
Quantidade		10	3	2
Campi em Implantação		**Técnico**		
		Integrado	**Concomitante**	**Subsequente**
1	Alvorada	-	-	-
2	Rolante	-	-	-
3	Vacaria	Multimídia	-	Manutenção e suporte em Informática
4	Veranópolis	Informática	-	-
5	Viamão	-	-	-
Quantidade		2	0	1
TOTAL GERAL		**12**	**3**	**3**

Fonte: A autora

Representando 35% têm-se as graduações e pós-graduação de TI do IFRS. É possível afirmar, analisando o Quadro 12, que apenas os já implantados é que possuem esses níveis de ensino. Dos 12 Campi já implantados, apenas dois destes não possuem cursos superiores de TI, e apenas o campus Porto Alegre tem pós-graduação na área até o momento da pesquisa.

Quadro 12 - Cursos de graduação e pós-graduação dos Campi do IFRS

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL				
Campi		**Superior**	**Especialização**	**Mestrado**
1	Bento Gonçalves	Tecnologia em Análise e Desenvolvimento de Sistemas	-	-
2	Canoas	Tecnologia em Análise e Desenvolvimento de Sistemas	-	-
3	Caxias do Sul	-	-	-
4	Erechim	-	-	-
5	Farroupilha	Tecnologia em Análise e Desenvolvimento de Sistemas	-	-
6	Feliz	Tecnologia em Análise e Desenvolvimento de Sistemas	-	-
7	Ibirubá	Ciência da Computação	-	-
8	Osório	Tecnologia em Análise e Desenvolvimento de Sistemas	-	-
9	Porto Alegre	Tecnologia em Sistemas para Internet	-	Informática na educação
10	Restinga	Tecnologia em Análise e Desenvolvimento de Sistemas	-	-
11	Rio Grande	Tecnologia em Análise e Desenvolvimento de Sistemas	-	-
12	Sertão	Tecnologia em Análise e Desenvolvimento de Sistemas	-	-
Quantidade		10	0	1
Campi em Implantação		**Superior**	**Especialização**	**Mestrado**
1	Alvorada	-	-	-
2	Rolante	-	-	-
3	Vacaria	-	-	-
4	Veranópolis	-	-	-
5	Viamão	-	-	-
Quantidade		0	0	0
TOTAL GERAL		**10**	**0**	**1**

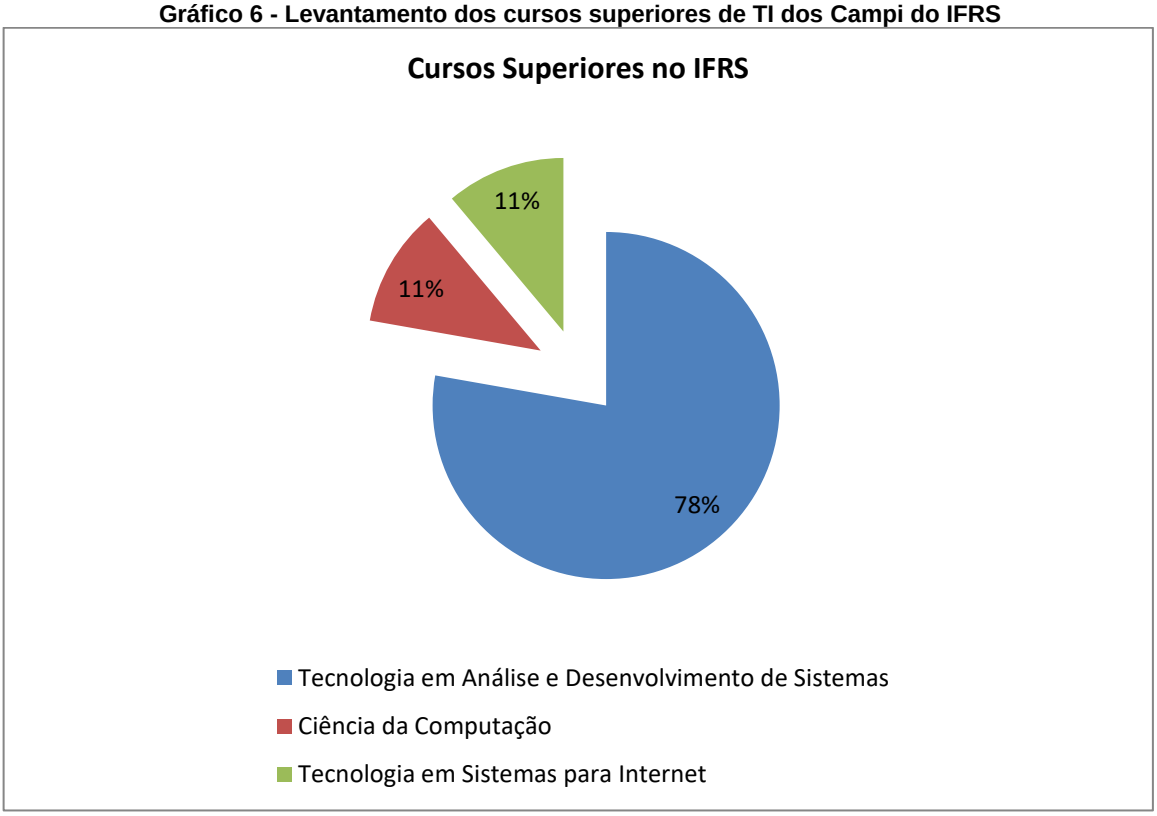
Fonte: A autora

Uma vez que o foco deste estudo se dá no ES, apenas estes foram levados em conta para os próximos levantamentos de dados. Neste sentido, utilizaram-se os PPC de cada curso, de cada campus, que estão disponíveis no site de cada uma das instituições.

Com total de dez cursos superiores, apenas um não corresponde a um tecnólogo, sendo ele bacharelado em Ciência da Computação. No Gráfico 6 é verificado que o curso mais representativo nos Campi do IFRS corresponde ao ADS

(Tecnologia em Análise e Desenvolvimento de Sistemas), podendo ser encontrado em oito Campi. Assim como o bacharelado em Ciência da Computação, o Tecnólogo em Sistemas para Internet é encontrado em apenas um campus.

Gráfico 6 - Levantamento dos cursos superiores de TI dos Campi do IFRS



Fonte: A autora

Foi possível identificar também através do levantamento documental nos PPCs dos Campi, que apesar de todos os cursos tecnólogos serem criados a partir do Catálogo Nacional de Cursos Superiores de Tecnologia (CNCST) e pertencerem à mesma rede de ensino, cada um deles apresentam uma matriz curricular distinta. O CNCST foi lançado pelo Ministério da Educação (MEC) em 2006, e serve de guia a respeito do perfil de competências do tecnólogo. Logo, é notável que a matriz curricular fique a critério de cada campus.

De acordo com o MEC, o tecnólogo em ADS

> analisa, projeta, desenvolve, testa, implanta e mantém sistemas computacionais de informação. Avalia, seleciona, especifica e utiliza metodologias, tecnologias e ferramentas da Engenharia de Software, linguagens de programação e bancos de dados. Coordena equipes de produção de softwares. Vistoria, realiza perícia, avalia, emite laudo e parecer técnico em sua área de formação.. (MEC, 2016ª, p. 52)

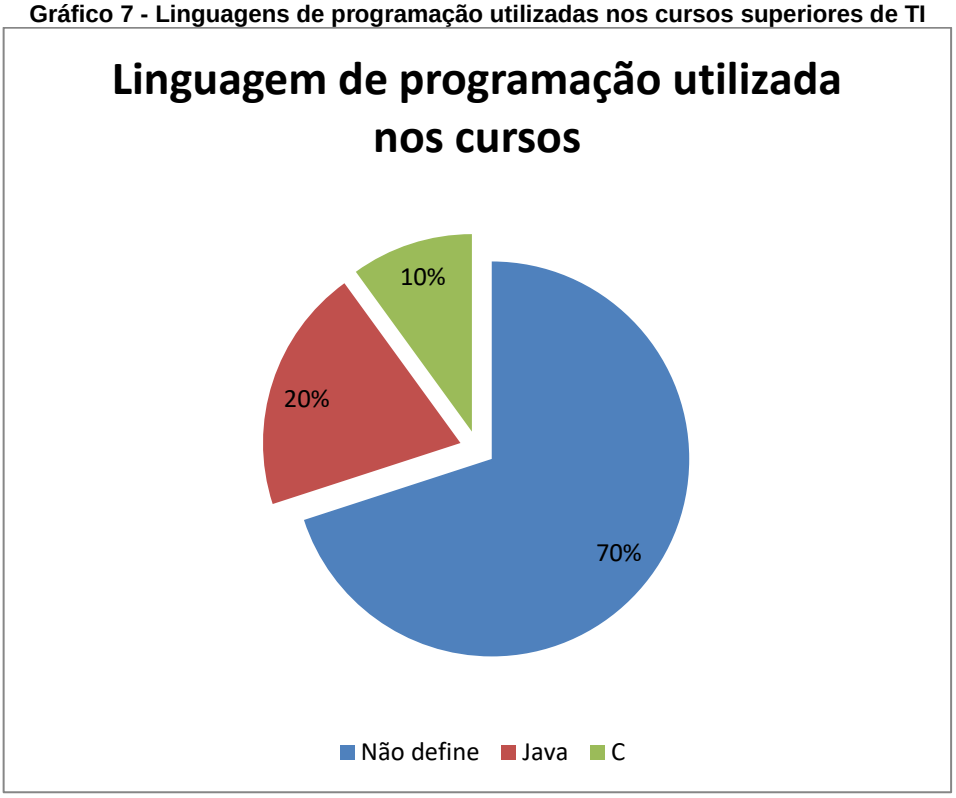
Já o tecnólogo em Sistemas para Internet (SI),

> projeta, desenvolve, testa, implanta, mantém, avalia e analisa páginas para sites de Internet e intranets, sistemas de comércio eletrônico e aplicativos para plataformas móveis para a Internet. Avalia, especifica, seleciona e utiliza metodologias e ferramentas adequadas para o desenvolvimento das aplicações. Elabora e estabelece diretrizes para a criação de interfaces adequadas à aplicação de acordo com características, necessidades e público-alvo. Vistoria, realiza perícia, avalia, emite laudo e parecer técnico em sua área de formação. (MEC, 2016, p. 63)

Indiferente do curso superior de TI, a linguagem de programação se faz presente. No Gráfico 13, observa-se que todos os campi trabalham com disciplinas relacionadas à linguagem de programação no primeiro semestre dos cursos, porém a carga horária e a forma como abordam, varia bastante.

Foi verificada também qual linguagem de programação é adotada nessas disciplinas introdutórias. Observa-se no Gráfico 7, que a maioria dos campi não define a linguagem de programação que utilizam, deixando, normalmente, para o grupo de professores escolherem. Acredita-se que a escolha esteja de acordo com a necessidade do curso e do mercado de trabalho.

Gráfico 7 - Linguagens de programação utilizadas nos cursos superiores de TI



Fonte: A autora

Quadro 13 - Disciplinas que contemplam CIP no primeiro semestre

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO GRANDE DO SUL							
Campi		Curso	Linguagem de programação	Lógica	Algoritmos	Laboratório algoritmos	CH TOTAL
1	Bento Gonçalves	Tecn. em Análise e Desen. de Sistemas[11]		x	X	x	120
2	Canoas	Tecn. em Análise e Desen. de Sistemas[12]			X		66
3	Farroupilha	Tecn. em Análise e Desen. de Sistemas[13]	x	x	X		90
4	Feliz	Tecn. em Análise e Desen. de Sistemas[14]	x	x			120
5	Ibirubá	Ciência da Computação[15]			X		66
6	Osório	Tecn. em Análise e Desen. de Sistemas[16]	x		X		66
7	Porto Alegre	Tecnologia em Sistemas para Internet[17]	x	x			132
8	Restinga	Tecn. em Análise e Desen. de Sistemas[18]	x				100
9	Rio Grande	Tecn. em Análise e Desen. de Sistemas[19]		x			135
10	Sertão	Tecn. em Análise e Desen. de Sistemas[20]	x	x			120

Fonte: A autora

[11] IFRS, Campus Bento Gonçalcez. **Projeto Pedagógico do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistema**. Coord. Maurício Covolan Rosito; Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Bento Gonçalves: 2015.

[12] IFRS, Campus Canoas. **Matriz curricular 0208/2012 Curso superior de tecnologia em análise e desenvolvimento de sistemas.** Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Canoas: 2012.

[13] IFRS, Campus Farroupilha. **Projeto Pedagógico do Curso Superior de Tecnologia em Sistemas para Internet**. Coord. Hugo André Klauck; Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Farroupilha: 2016.

[14] IFRS, Campus Feliz. **Projeto Pedagógico do Curso Superior de Tecnologia em Sistemas para Internet**; Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Feliz: 2014.

[15] IFRS, Campus Ibirubá. **Projeto Pedagógico do Curso Superior de Ciência da Computação**; Coord. Tiago Rios da Rocha; Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Ibirubá: 2015..

[16] IFRS, Campus Osório. **Projeto Pedagógico do Curso Superior de Análise e Desenvolvimento de Sistemas**; Coord. Bruno Chagas Alves Fernandes; Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Osório: 2016.

[17] IFRS, Campus Porto Alegre. **Projeto Pedagógico do Curso Superior de Tecnologia em Sistemas para Internet**. Coord. Tanisi Pereira de Carvalho; Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Porto Alegre: 2013.

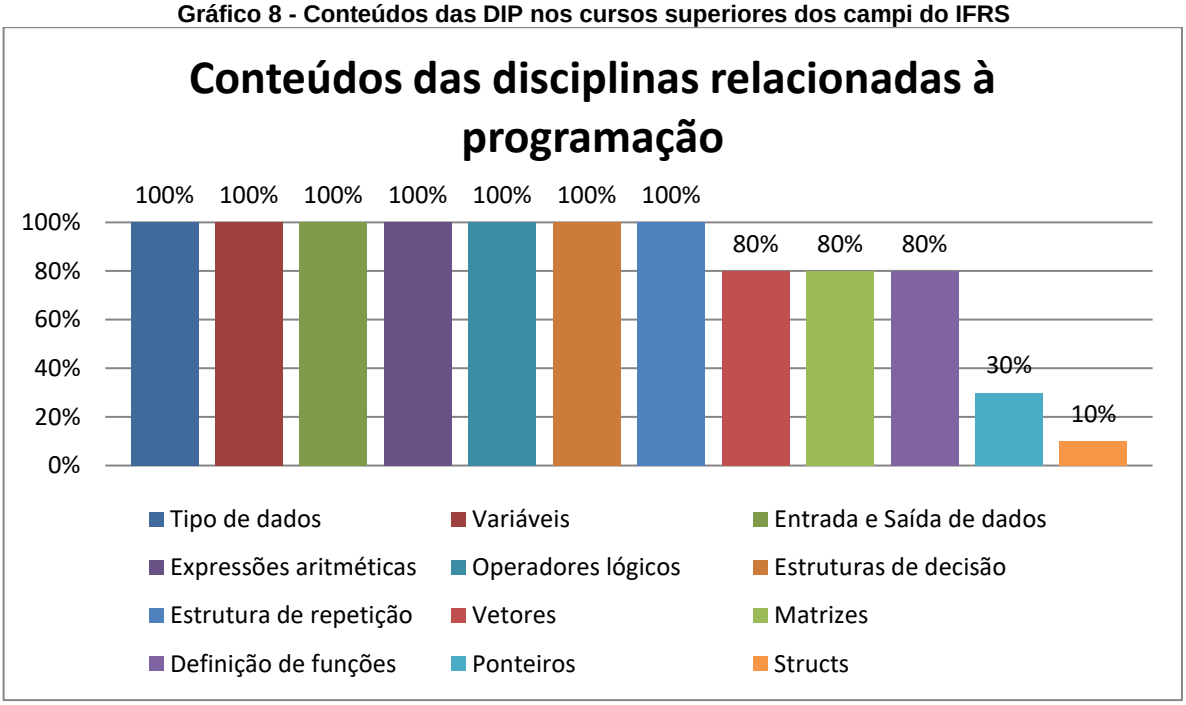
[18] IFRS, Campus Restinga. **Projeto Pedagógico do Curso Superior de Tecnologia em Sistemas para Internet**.; Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Restinga: 2016.

[19] IFRS, Campus Rio Grande. **Projeto Pedagógico do Curso Superior de Tecnologia em Sistemas para Internet**. Coord. Tiago Lopes Telecken; Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Rio Grande: 2013.

[20] IFRS, Campus Sertão. **Projeto Pedagógico do Curso Superior de Tecnologia em Sistemas para Internet**; Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Sertão: 2014.

Além de conhecer os cursos, o levantamento documental tem o intuito de embasar a proposta pedagógica, identificando as necessidades das disciplinas relacionadas à introdução a programação. Para isto, analisaram-se as ementas dessas disciplinas, a fim de verificar os conteúdos previstos para essas, o que pode ser observado no Gráfico 8.

Gráfico 8 - Conteúdos das DIP nos cursos superiores dos campi do IFRS



Fonte: A autora

Ao visualizar o Gráfico 8, é possível afirmar que a maioria dos campi opta em trabalhar com os conceitos: tipo de dados, variáveis, entrada e saída de dados, expressões aritméticas, estruturas de decisão e estruturas de repetição. Alguns dos campi ainda ensinam sobre vetores, matrizes e definição de funções.

Nesse sentido, de acordo com o Gráfico 8, foram descartados alguns conteúdos para que a proposta pedagógica esteja relacionada com as ementas das disciplinas de programação. Assim sendo, a ConsProg juntamente à RB, utilizam os conteúdos apresentados no Gráfico 8 que possuem 100% de representatividade.

Na próxima seção, apresenta-se o estudo de caso, realizado no IFRS campus Porto Alegre. Uma vez que o mesmo ocorreu em três semestres, e em cada um deles com uma turma diferente, a seção encontra-se dividida em três subseções.

6.2 ESTUDO DE CASO

O estudo de caso ocorreu no IFRS Campus Porto Alegre, com turmas do primeiro semestre do curso Superior Tecnológico em Sistemas para Internet, na disciplina de programação. Para melhor compreensão, a presente seção possui subseções que apresenta a análise de cada etapa do estudo de caso, organizados de acordo com o semestre aplicado: subseção 6.2.1 trata da observação sem a proposta – primeira etapa; subseção 6.2.2 traz a observação com o uso da proposta pedagógica ConsProg – segunda etapa; 6.2.3 fala sobre o relato do docente sobre o uso da ConsProg – terceira etapa.

6.2.1 Primeira etapa do estudo de caso

Esta subseção apresenta a análise de dados da primeira etapa estudo de caso, que ocorreu no primeiro semestre de 2016, no turno da noite. O intuito deste foi de observar as aulas, acompanhar o desenvolvimento dos estudantes e também a metodologia utilizada pelo professor na disciplina.

No decorrer das aulas, pode-se verificar que muitas vezes os estudantes não conseguiam acompanhar as explicações, e na maioria das vezes se omitiam, permitindo assim que o professor seguisse com o conteúdo. Analisando o cronograma da disciplina, verificou-se que o professor organizou os conteúdos previstos na ementa, no qual, muitas vezes, continha mais de um conceito a ser exposto pelo mesmo em um encontro. Um exemplo disso é o conteúdo programático da aula 6, em que ensina os comandos de estruturas de repetição `for`, `while` e `do while`.

Apesar de todos estes comandos pertencerem ao conceito de estruturas de repetição, cada um deles tem suas especificidades, o que pode gerar confusão nos estudantes, visto que são três técnicas diferentes. Cada uma dessas técnicas é utilizada para algumas funções específicas, sendo assim, ao apresentar os comandos em uma só aula, o estudante não terá a possibilidade de praticar cada uma delas e acomodar esse conceito. De acordo com Bloom et al. (1972), é necessário que o estudante perpasse pelos níveis para desenvolver o cognitivo e assim ter um aprendizado mais efetivo.

No Quadro 14 apresentam-se três possíveis respostas, para um programa que deve imprimir na tela do usuário números inteiros de 0 a 10. Os exemplos mostram o mesmo resultado para o usuário, porém as técnicas utilizadas para programar são diferentes. Enquanto que o comando `for` possui todos os elementos de controle de repetição em uma única linha, o `while` e o `do while` têm estes elementos dispersos e permite a construção de condições mais complexas, podendo essas serem por meio de várias estruturas de comando como os `if`.

Quadro 14 - Exemplos utilizando estruturas de repetição

Comando `for`

```
l.01  int main(int argc, char** argv){
l.02      int i;
l.03      for (i = 0; i < = 10; i++ ){
l.04          printf("%d\n" , i );
l.05      }
l.06      return (EXIT_SUCCESS);
l.07  }
```

Comando `while`

```
l.01  int main(int argc, char** argv){
l.02      int i = 0;
l.03      while ( i <= 10){
l.04          printf("%d\n" , i );
l.05          i++;
l.06      }
l.07      return (EXIT_SUCCESS);
l.08  }
```

Comando `do while`

```
l.01  int main(int argc, char** argv){
l.02      int i = 0;
l.03      do{
l.04          printf("%d\n" , i );
l.05          i++;
l.06      } while ( i <= 10);
l.07      return (EXIT_SUCCESS);
l.08  }
```

Fonte: A autora

Ao utilizar o comando `for` (Quadro 14) para desenvolver o programa mencionado, o estudante cria uma variável inteira `i`, na sequência inclui a linha de comando da estrutura de repetição, no qual informa que a variável `i` deve iniciar em zero (`i = 0`), depois informa a condição em que deve repetir enquanto ela for verdadeira (`i <= 10`), e por fim o contador que irá controlar a repetição (`i ++`). Apesar de parecer simples, o estudante necessita ter domínio de alguns conceitos anteriores: variáveis, operadores lógicos e operadores matemáticos. Então, levando

em conta que o mesmo tenha domínio dos conceitos anteriores e sabe como aplicá-los, ainda é necessário compreender como irá funcionar esse bloco de comandos, uma vez que o estudante estaria acostumado a escrever os códigos linha a linha, ou seja, utilizaria apenas uma linha de comando repetidas vezes (Quadro 15).

Quadro 15 - Exemplo de solução sem estruturas de repetição

```
printf("0");
printf("1");
...
printf("10");
```

Fonte: A autora

Durante as aulas em que o professor apresentava mais de um conceito ou comando mais complexo, pode-se observar que alguns estudantes se mostravam confusos, o que pode intervir na abstração desses conceitos. De acordo com Piaget (1983) não se deve acelerar o aprendizado, uma vez que poderá romper com o equilíbrio, no qual “eis pois o estado de fato: há variações na velocidade e duração do desenvolvimento” (1983, p. 223). Neste sentido, caso o estudante não se aproprie de um determinado conceito e consiga aplicá-lo, o mesmo terá dificuldades em aprender o próximo, uma vez que seu desenvolvimento cognitivo pode não estar preparado para isto.

Neste sentido, é importante que o professor retome os conceitos anteriores de forma sucinta, a fim de situar o estudante daquilo que será necessário para aprender o novo conteúdo, já que “a passagem da ação ao pensamento [...] não se realiza sob forma de uma revolução brusca, mas, pelo contrário, de uma diferenciação lenta e laboriosa, que se relaciona às transformações da assimilação” (PIAGET, 1983, p. 13).

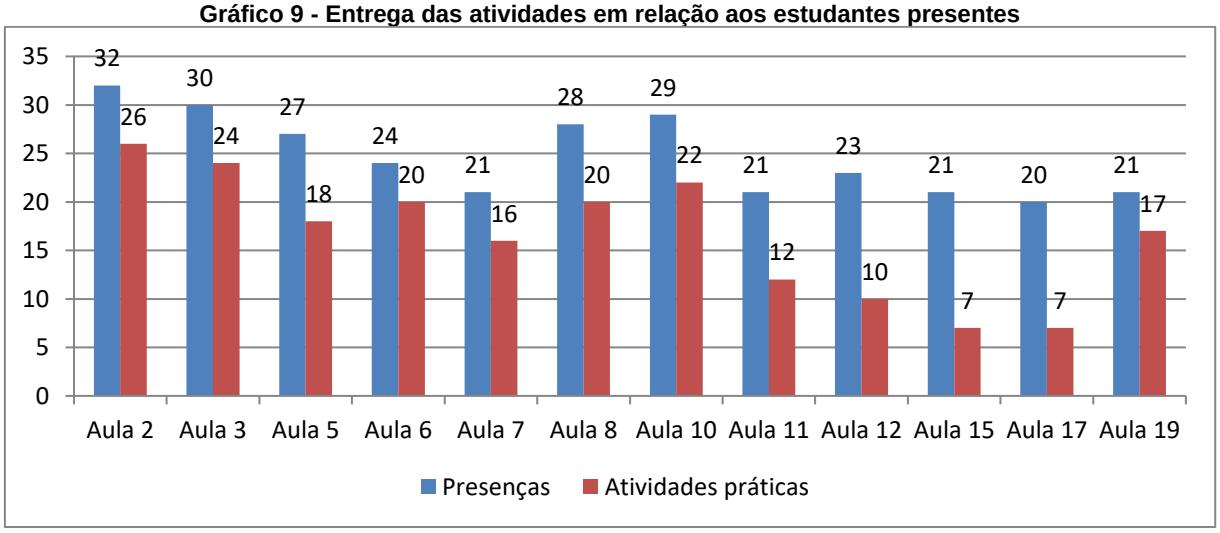
Apesar do professor não retomar os conceitos, os estudantes tinham acesso ao material apresentado em aula, bem como atividades e vídeo de apoio, através do AVA Moodle. Assim, em cada aula o professor liberava o acesso aos slides que seriam trabalhados, aos questionários teóricos, exercícios práticos de programação e algumas respostas de atividades da aula anterior. Logo, as aulas eram divididas em três momentos: correção das atividades práticas da aula anterior, ensino do novo conteúdo, e atividades do novo conteúdo.

Primeiramente o professor corrigia algumas atividades da aula anterior, de acordo com as dúvidas da turma, durante este momento pode-se observar que poucos estudantes questionavam, e o docente tinha que pedir constantemente a participação

deles. Na segunda parte da aula, o professor passava o novo conteúdo, com alguns exemplos práticos (códigos e trechos de códigos). E por fim, os estudantes tinham espaço para realizar as atividades e contar com o auxílio do professor, contudo, muitos iam embora, e os que ficavam dificilmente pediam ajuda ou tiravam dúvidas. Por ficar observando no fundo do laboratório, era possível visualizar a cada início de aula que muitos estudantes não haviam realizado as atividades, e alguns ainda comentavam entre eles que nem sabiam como começar, e mesmo assim não pediam auxílio ao docente.

Neste contexto, acredita-se que o acompanhamento dos estudantes por meios das atividades seja importante para verificar as possíveis dificuldades dos mesmos, ou pelo menos verificar que eles não estão realizando as tarefas e assim tentar compreender o que está ocorrendo. O Gráfico 9 apresenta dados obtidos no AVA Moodle referente à entrega de atividades juntamente com a quantidade de estudantes presentes nestes dias durante o semestre 2016/1 na disciplina Linguagem de Programação I. Com isso é possível dizer que desde o início do semestre alguns estudantes já apresentavam dificuldades, uma vez a entrega das atividades eram menores que a quantidade de estudantes presentes (Gráfico 9).

Gráfico 9 - Entrega das atividades em relação aos estudantes presentes



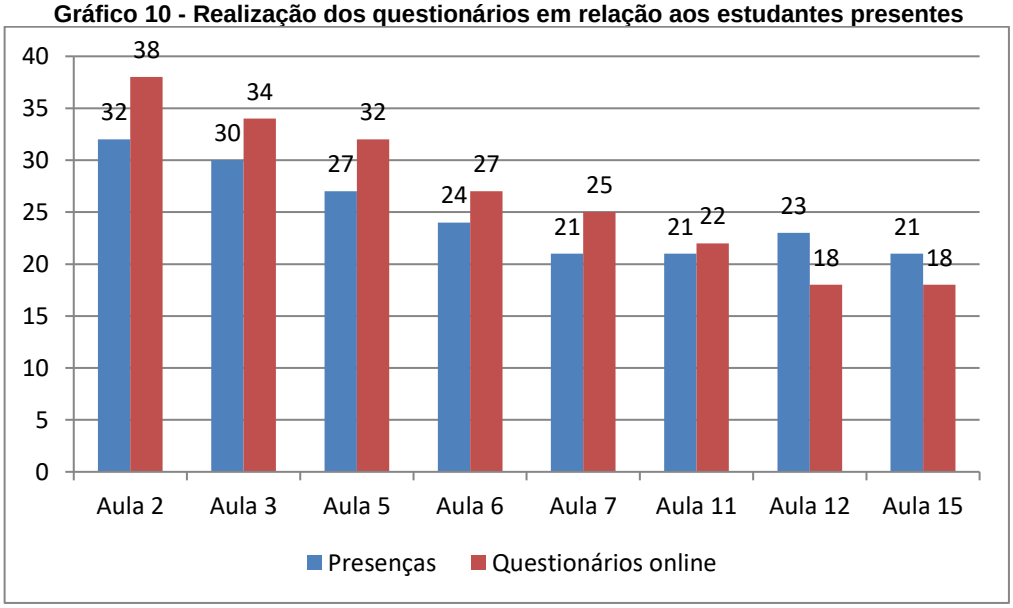
Fonte: IFRS campus Porto Alegre

Destaca-se que o prazo de entrega da resolução das atividades estipulado pelo professor era de 10 dias, sendo que o estudante tinha parte da aula em que a atividade era disponibilizada para resolver os exercícios. Além disso, na aula seguinte o professor realizava a correção de alguns desses exercícios e sanava dúvidas dos

estudantes, quando os mesmos pediam. Após essa correção, o estudante ainda tinha três dias para enviar as suas resoluções pelo AVA Moodle.

Por exemplo, no dia 3 de março de 2016 (aula 02), o professor distribui a atividade sobre variáveis e tipos de dados. Nesse dia, os estudantes puderam realizá-las no laboratório com acompanhamento do professor. Na aula seguinte no dia 10 de março de 2016 (aula 03), o professor corrigiu alguns desses exercícios de variáveis e tipos de dados, respondendo a perguntas dos estudantes. Consequentemente, os estudantes puderam enviar as suas respostas até o dia 13 de março de 2016 pelo AVA Moodle. Neste sentido, os estudantes poderiam tirar suas dúvidas, ter mais três dias para realizar as tarefas e submetê-las posteriormente, porém mesmo assim, apenas 26 dos 32 estudantes fizeram a entrega. Observa-se ainda no Gráfico 9, que a taxa de entrega em relação aos presentes diminui no decorrer do semestre.

Gráfico 10 - Realização dos questionários em relação aos estudantes presentes



Fonte: IFRS campus Porto Alegre

Já os questionários eram respondidos até por aqueles que não compareciam as aulas (Gráfico 10), podendo estar relacionado ao fato dos estudantes terem habilidade e capacidade para os níveis de conhecimento e compreensão, não possuindo ainda características do nível de aplicação que é necessário ao desenvolver os programas computacionais. Essa informação pode ser melhor compreendida ao comparar os dados do Gráfico 9 com o do Gráfico 10, visto que mais estudantes conseguiram responder o questionário do que realizar as atividades de programação. Portanto, os estudantes demonstraram já terem memorizado os

conceitos, mesmo que ainda não relacionados com outros conteúdos, mas não haviam demonstrado como utilizar esses nas novas situações presentes nos exercícios práticos.

Vale ressaltar que tanto as atividades práticas quanto os questionários elaborados pelo professor não possuíam um aumento gradativo de dificuldades. As atividades práticas eram criadas e disponibilizadas no AVA Moodle para que os estudantes fizessem a entrega. Os questionários tratavam-se de questões teóricas, em sua maioria, e não havia o cuidado de criá-las obedecendo graus de dificuldades. Assim, acredita-se que ao desenvolver exercícios mais pedagógicos, no qual dê a chance de todos os estudantes se saírem bem em algum nível, pode motivá-los a seguir buscando o conhecimento que lhes falta.

Além disso, através dessas atividades, ficaria mais evidente para o docente o nível de aprendizado em que o estudante se encontra, consequentemente poderá auxiliá-lo, respeitando seu tempo de assimilação. Uma vez que “o domínio cognitivo obedece a uma ordem hierárquica, e cada uma das classes de capacidades e habilidades envolve exigências relativas às classes de nível inferior” (BLOOM et al., 1972, p. 103), e que “toda aquisição nova consiste em assimilar um objeto ou uma situação a um esquema anterior aumentando assim esse esquema” (PIAGET, 1983, p. 245), acredita-se que alguns estudantes necessitem de mais tempo para aprender, e ter um novo conteúdo a cada aula pode dificultar esse processo. Isso pode ser agravado nos casos dos estudantes trabalhadores, os quais dispõem de pouco tempo extraclasse para realização de atividades.

Neste sentido, caso o estudante disponha de mais tempo, poderá aumentar seu nível nos níveis da TOEDC, logo, quanto maior o nível, mais apropriação o estudante terá sobre o conteúdo. Assim, será necessário que o mesmo empregue tempo suficiente manipulando o objeto de aprendizagem, pois somente assim ele poderá assimilar novos esquemas de ações. Portanto, ao ter o tempo limitado para o aprendizado, o estudante é confrontado com uma situação crítica da área da programação: aplicação de um conceito integrado aos conceitos anteriores. Por exemplo, ao aplicar o conceito de estruturas de repetição, é necessário aplicar conjuntamente o conceito de operadores relacionais.

Ao analisar estas situações, pode-se dizer que para o estudante conseguir acompanhar a disciplina de LPI é necessário que o mesmo tenha conseguido atingir pelo menos o nível aplicação da TOEDC, uma vez que esse tipo de disciplina não é

apenas teórica, e na construção de um programa muitos conceitos são utilizados. Vale ressaltar que o nível aplicação é o primeiro a requerer um nível de abstração maior, segundo Bloom et al.

> o aluno ao demonstrar ‘compreensão’, pode usar a abstração quando seu uso está especificado. Na ‘aplicação’, o aluno deve usar corretamente a abstração em uma situação na qual ela não estiver de modo algum especificada (1972, p. 103).

Ao analisar as atividades práticas, pode-se constatar que os estudantes aprovados na disciplina realizaram em torno de 73% das atividades. Além disso, foi possível observar que até a metade da disciplina esses estudantes entregaram uma média de 95% das atividades. Acredita-se que esses dados podem apontar que os CIP (variáveis, tipo de dados, operadores aritméticos, operadores relacionais, operadores lógicos, estruturas de decisão e estruturas de repetição) estudados na primeira metade da disciplina possam influenciar o desenvolvimento cognitivo dos estudantes a respeito de programação.

Nesse sentido, é suposto que o professor deva investir mais tempo nesses conceitos, assim mais estudantes conseguiriam acompanhar a disciplina. É possível fazer essa constatação visto que ao cruzar os dados entre as entregas de atividades, avaliações e desistência de estudantes após o simulado e primeira avaliação, verificou-se que estes entregaram em torno de 37,5% das atividades. Assim sendo, os conceitos iniciais se mostram de fato importantes para que auxilie o desenvolvimento cognitivo dos estudantes para o aprendizado de programação. No tópico a seguir é apresentado uma análise tendo como foco o desenvolvimento dos estudantes na disciplina de LPI.

6.2.1.1. Os estudantes e a disciplina de LPI

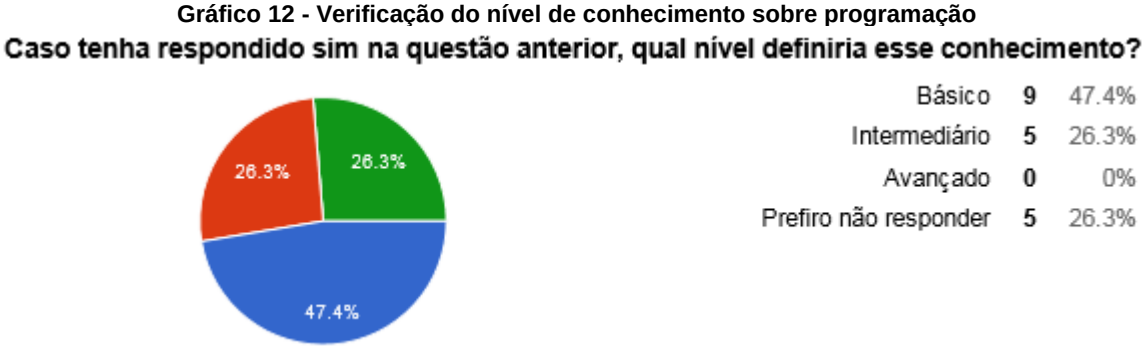
Com o intuito de conhecer a opinião sobre a disciplina de programação, bem como o perfil dos sujeitos participantes, foi realizado um questionário fechado (Apêndice A). Os estudantes que responderam este questionário, 95% foram aprovados na disciplina, sendo que 78,9% deles responderam que já tinham conhecimentos em programação antes de iniciar o curso STSI (Gráfico 11).

Gráfico 11 - Verificação do conhecimento de programação anterior ao curso

Fonte: A autora

Além disso, foi questionado como os estudantes avaliariam o nível de conhecimento que tinham em programação. Observando o Gráfico 12 nota-se que 47,4% dos estudantes responderam ter conhecimento básico em programação, equivalente ao que eles estudam na disciplina, sendo esse um facilitador para os mesmos.

Gráfico 12 - Verificação do nível de conhecimento sobre programação



Fonte: A autora

Analisando esses dados em conjunto com o levantamento de estudantes aprovados na disciplina, surge um questionamento: “A disciplina foi pensada para estudantes que já programam ou para àqueles que estão iniciando na área? ”. Essa pergunta surgiu uma vez que ao se somar a quantidade de estudantes que possuíam conhecimentos “Básico” e “Intermediário” em programação, tem-se um total de 14 (conforme Gráfico 12). Já a quantidade de estudante aprovados na disciplina corresponde a 18, sendo que desses, 14 eram os que possuíam conhecimento prévio.

Os estudantes que apresentaram maior dificuldade no semestre fizeram alguns relatos sobre as barreiras enfrentadas durante o aprendizado dos CIP. Um desses relatos transparece a opinião daqueles que não possuíam conhecimento prévio: “Nas

aulas de C eu não conseguia entender nada de nada, por mais que tivesse slides e as respostas dos exercícios eu não sabia como começar os programas. ” Outro comentário que chama atenção foi dos estudantes que não tiravam suas dúvidas com o professor: “[...] não fazia ideia de como iniciar os programas, e ao longo do semestre pedi ajuda para colegas e mesmo assim: nada! ”

Em ambos comentários dos estudantes fica claro que as dificuldades eram dos CIP, uma vez que não sabiam nem como começar um programa. Além disso, muitos falaram da falta de tempo para se dedicarem à disciplina, outros alegavam que nem buscando vídeos explicativos conseguiam compreender. Questionando esses estudantes sobre o motivo pelo qual não pediam auxílio ao professor, muitos alegaram que não conseguiam nem formular suas perguntas para “tentar aprender”.

Uma vez que a atividade avaliativa é permitida a consulta nos materiais, alguns discentes comentaram que tentavam fazer “Ctrl + C” e “Ctrl + V” de programas prontos, mas que também não sabiam ao certo o que deveria ser copiado. E quando achavam que haviam conseguido desenvolver o programa, ao compilar o mesmo dava erro e os estudantes não compreendiam o que estava errado. Através desses relatos, acredita-se que os CIP devam ser melhor desenvolvidos em sala, sem que haja uma preocupação em “vencer os conteúdos”.

Com o intuito de ilustrar o que os estudantes com dificuldades relataram, fez-se uma análise quantitativa em cima dos dados do caderno de chamada do professor da disciplina, sobre frequência e conceito. Essa análise dos estudantes foi dividida em três momentos: frequência e conceito na primeira metade do semestre, frequência e conceito na segunda metade do semestre levando em conta os dados levantados no momento anterior, e situação final do estudante levando em conta todo o semestre.

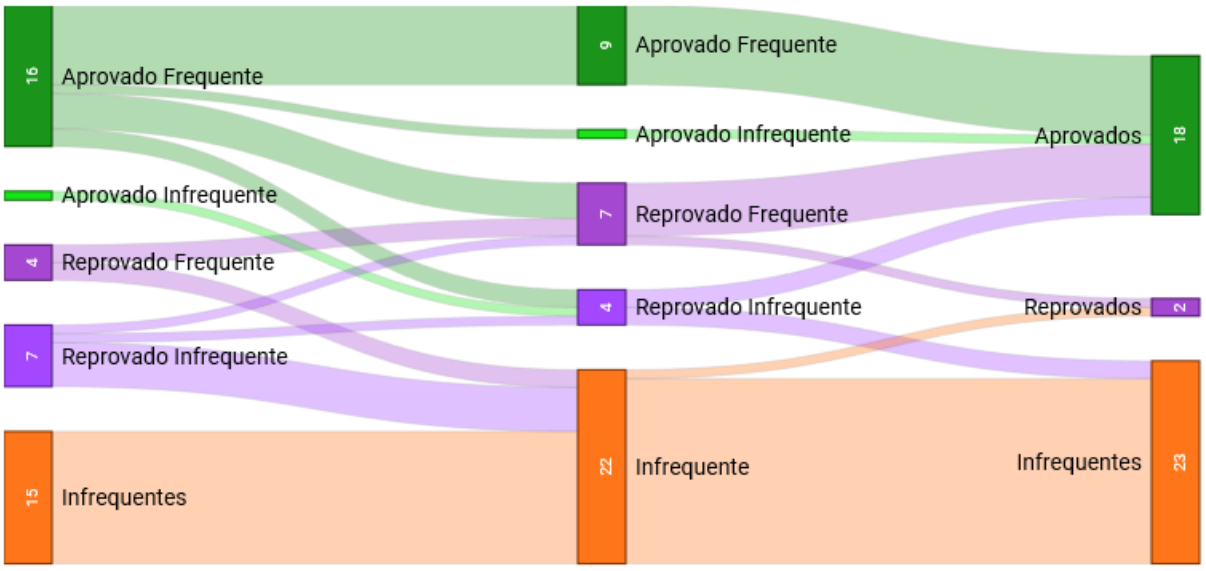
Para chegar nesses resultados foram definidas algumas situações levando em conta as informações sobre aprovação e reprovação contidas no PPC do curso. Logo, delineou-se as seguintes situações:

- Aprovado frequente: possuir conceito A, B ou C, e frequência superior ou igual a 75%;
- Reprovado frequente: possuir conceito D, e frequência inferior a 75%;
- Reprovado infrequente: possuir conceito D, e frequência inferior a 75%;
- Infrequente: possuir conceito E, e frequência inferior a 75%;

- Reprovado (utilizado apenas para o final do semestre): possuir conceito D e frequência superior ou igual a 75%.

Para um melhor acompanhamento da situação dos estudantes, fez-se o uso do gráfico Sankey[21] através dos dados levantados no caderno de chamada do semestre 2016/1. Assim, no Gráfico 13 é possível verificar que na primeira metade do semestre já haviam 23 casos de estudantes infrequentes, sendo que desses, 15 não realizaram a primeira avaliação. Dos oito que fizeram a primeira atividade avaliativa, apenas um foi considerado aprovado por atingir o conceito mínimo para tal. Porém, este mesmo estudante que tinha conceito para aprovação na primeira metade do semestre, evadiu a disciplina.

Gráfico 13 - Acompanhamento de frequência e conceito dos estudantes (2016/1)



Fonte: A autora

Os outros sete estudantes (reprovado infrequente), cinco evadiram a disciplina, e dois tentaram realizar a segunda atividade avaliativa. Um dos que realizou a avaliação continuou infrequente na segunda metade do semestre e não teve conceito mínimo para aprovação, e o outro apesar de ter tido frequência, não foi aprovado, por também não ter atingido conceito mínimo.

Já ao averiguar os estudantes que tinham frequência (aprovado frequente e reprovado frequente), tem-se o quantitativo de 20. Mesmo tendo frequência, quatro

[21] Diagrama que representa fluxos de entrada e saída, em que as espessuras das linhas são proporcionais aos dados.

não atingiram o conceito mínimo, e desses, dois evadiram a disciplina, um foi reprovado e apenas um aprovado. Já os outros 16 que estavam aprovados na primeira metade, apenas nove aprovaram atingiram conceito mínimo na segunda metade, mas através da análise qualitativa feita professor da disciplina a respeito do desempenho dos estudantes, os 16 foram aprovados.

Com isso pode-se afirmar que os estudantes que apresentam dificuldades no aprendizado dos CIP não conseguem ter bom rendimento na segunda metade. Isso pode ocorrer uma vez que

> a assimilação nunca pode ser pura, visto que, ao incorporar os novos elementos nos esquemas anteriores, a inteligência modifica incessantemente os últimos para ajustá-los aos novos dados. Mas inversamente, as coisas nunca são conhecidas em si mesmas, porquanto esse trabalho de acomodação só é possível em função do processo inverso de assimilação (PIAGET, 1975, p. 18).

Logo, deve-se pensar em uma readequação da ementa, uma vez que a mesma é densa, havendo em torno de 10 conteúdos[22] para 20 aulas, sendo que muitos deles se desdobram em mais conceitos. Portanto, através da ConsProg tentar-se-á constatar em que momentos durante o semestre os estudantes apresentam maiores dificuldades. Na subseção a seguir é feita a análise da segunda etapa do estudo de caso.

6.2.2 Segunda etapa do estudo de caso

Realizada no segundo semestre de 2016, observou-se a aplicação da proposta pedagógica feita pelo mesmo docente do semestre anterior. Os materiais de aula foram reformulados: slides, questionários teóricos, dicas e erros comuns em programação, todos desenvolvidos de acordo com a ConsProg.

Para a adequação dos materiais, a pesquisadora e o docente reuniram-se para verificação dos materiais “antigos” do professor, e quando necessário fez-se adaptações que atendessem a proposta pedagógica, ou seja, uso de materiais concretos, dinâmicas, exemplos do cotidiano dos estudantes. Os questionários

[22] Como apresentado na seção 2.3, os conteúdos trabalhados na disciplina de Linguagem de Programação I são: variáveis, tipos de dados, operadores (aritméticos, relacionais e lógicos), estruturas de controle (`if, switch`), estruturas de repetição (`for, while, do`), vetores, matrizes, definições de funções, ponteiros e estruturas (`structs`) para definição de tipos abstratos de dados

teóricos foram reformulados, uma vez que os desenvolvidos pelo professor não faziam o uso dos níveis da TOEDC, não proporcionando uma análise facilitada do desenvolvimento de aprendizagem do estudante. Estes questionários foram disponibilizados no Google através dos formulários, uma vez que faz a tabulação dos dados e geração de gráficos de forma automática. Apesar do AVA Moodle também apresentar essas ferramentas de tabulação de dados e geração de gráficos, os apresentados pelo Google ficam mais claros e proporcionam mais opções de configuração.

Acrescentou-se aos materiais de apoio dos estudantes conteúdos relacionados a erros comuns e dicas de programação retirados do livro da bibliografia básica da disciplina. Manteve-se os três momentos da aula (correção das atividades, conteúdo novo, atividades relacionadas ao novo conteúdo), porém tendo o cuidado de iniciar as explicações com exemplos do cotidiano dos estudantes para que os mesmos “acessassem” suas estruturas cognitivas semelhantes a esse novo conhecimento. Através desse “acesso” foi possível estimular o estudante a assimilar o conteúdo e posteriormente acomodá-lo, visto que “todo esquema de assimilação tende a se alimentar, ou seja, incorporar os elementos que são exteriores e compatíveis com sua natureza [...]” (PIAGET, 1976, p. 52).

Durante o decorrer das aulas observou-se, assim como no semestre anterior, que os estudantes pouco participavam da correção das atividades práticas, não tirando as dúvidas com o professor. Porém, durante as explicações com exemplos práticos e dinâmicas, os discentes motivavam-se a participar e a responder os questionamentos do professor, visto que “dominavam” o assunto. A partir do momento em que as explicações passavam a ser mais técnicas, a interação reduzia e aumentava a distração dos estudantes.

Logo, acredita-se que quanto menos técnico for as explicações iniciais, maior a chance do desenvolvimento de aprendizado, visto que quando se tem familiaridade com o novo, há afetividade. Na TC de Piaget, a afetividade é a “[...] mola das ações das quais resulta, a cada nova etapa, a ascensão progressiva [...], pois é a afetividade que atribui valor às atividades e lhes regula a energia. ” (1974, p.70). Sendo assim, é necessário que o estudante se reconheça naquilo que irá aprender, já que

> é indiscutível que o afeto tem um papel essencial no funcionamento da inteligência. Sem o afeto não haveria nem interesses, nem necessidades, nem motivação; em consequência, as interrogações ou problemas não

> poderiam ser formulados e não haveria inteligência. O afeto é uma condição necessária para a constituição da inteligência. No entanto, em minha opinião, não é uma condição suficiente. ” (PIAGET, 1974 p.129).

Agregado a afetividade, as explicações do professor devem ser aprofundadas aos poucos, para que haja tempo de ocorrer assimilação, acomodação e equilibração dos conteúdos trabalhados. Apesar da tentativa do docente em fazer isso, muitas vezes ocorriam “cortes” entre os exemplos concretos e os conceitos técnicos, não permitindo, talvez, que os estudantes com maior dificuldade acompanhassem de forma exponencial o conteúdo.

Assim sendo, durante todo o semestre foram necessárias reuniões entre o discente e a pesquisadora com o intuito de adequar alguns procedimentos da aula com a ConsProg. É importante ressaltar neste momento, que essas adequações não estão relacionadas com a forma que o docente estava realizando suas aulas, mas sim com a interpretação que ele havia tido sobre a ConsProg. Logo, vale pensar se a criação de metodologias, técnicas e propostas pedagógicas por si só mudará o ensino, visto que cada profissional fará uma leitura sobre as mesmas, e tentará aplicar da forma que as interpretou.

Acredita-se assim, que o investimento na formação didático/pedagógica dos profissionais da área de educação é imprescindível para o aprimoramento do ensino. Porém esta deve ser feita não apenas através de cursos, palestrar e “importação” de modelos, presume-se que um especialista que acompanhe esse processo e direcione o docente àquilo que de fato é esperado seja de grande valia. Neste momento é possível fazer essa constatação, uma vez que além de criar a proposta pedagógica, a pesquisadora acompanhou aula a aula a prática da mesma sendo executada por alguém que interpretou a ConsProg.

Através desse acompanhamento ficou evidente que nem sempre o que se idealiza é interpretado da mesma forma por outra pessoa. Sendo assim, afirmar-se novamente sobre a importância do acompanhamento de um especialista da área da educação, para que este auxilie o professor a compreender o que é esperado daquele processo de ensino-aprendizagem. Logo, no decorrer das aulas a ConsProg foi sendo aprimorada, tanto por parte do professor em sua prática, quanto pela pesquisadora que buscou tornar as explicações da proposta mais claras.

Á vista disso, com a utilização da ConsProg e com as reuniões com o professor da disciplina, foi possível identificar que para aprender a programar não basta atingir

o nível aplicação da TOEDC. Visto que no nível aplicação o estudante consegue aplicar apenas o conceito aprendido naquele momento, não fazendo interações com os demais. Neste sentido, para que o estudante aprenda a programar, é esperado que o mesmo atinja o nível síntese, no qual exigirá a apropriação dos outros conteúdos para que seja possível a criação de um programa completo e não apenas trechos de códigos que sozinhos nada fazem. Para Bloom et al. (1972) o nível síntese requer habilidades de unir partes, tendo como finalidade a criação de um todo novo, podendo ser este um conjunto de relações abstratas combinadas.

Por meio da ConsProg foi possível constatar mais uma vez que a ementa da disciplina deve ser repensada, visto que há excesso de conteúdos para uma DIP no qual muitos estudantes ingressam sem conhecimento prévio de programação. Além disso, acredita-se que uma reformulação no cronograma poderá auxiliar os estudantes, deixando alguns conceitos mais abstrato (manipulação de Strings, por exemplo) para o final do semestre, pois assim o desenvolvimento cognitivo dos estudantes poderá estar melhor preparado para o entendimento destes. Assim, foram sugeridas algumas alterações para o professor da disciplina:

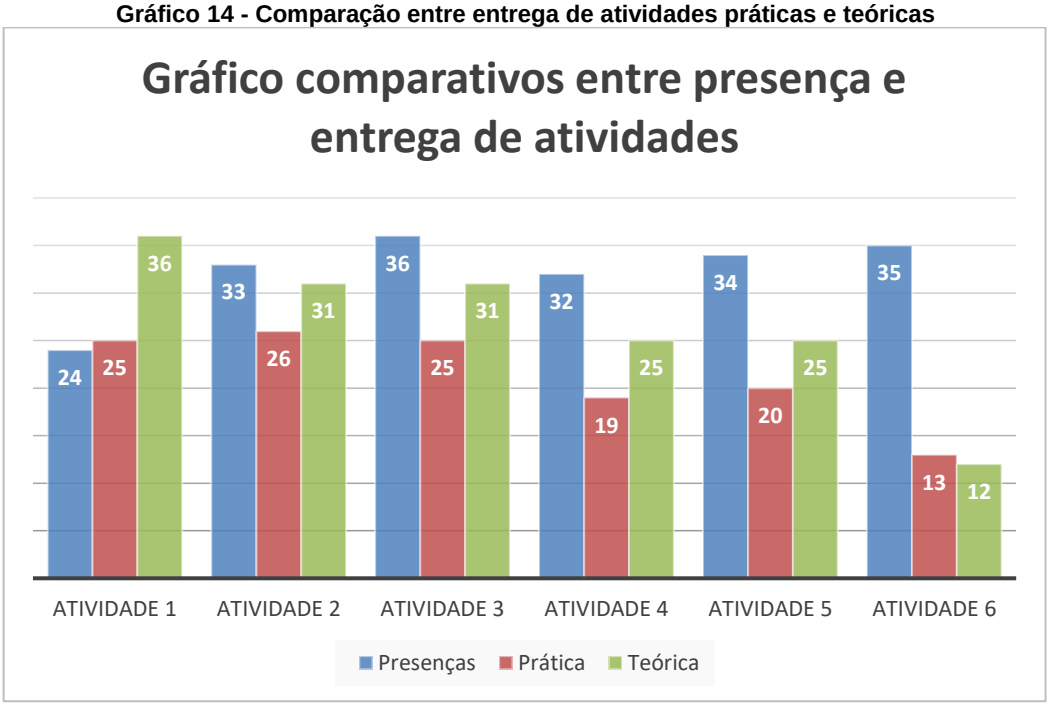
- Reduzir a quantidade de conteúdos e conceitos dados em uma aula;
- Organizar mais aulas para os CIP;
- Trabalhar variáveis e tipos de dados em uma aula separada de entrada e saída de dados;
- Dar ênfase, ao ensinar estrutura de controle, nos operadores lógicos e relacionais,
- Ensinar manipulação de `String` depois da primeira avaliação e não nas aulas iniciais como era realizado;
- Separar, em pelo menos duas aulas, o ensino de estrutura de repetição;
- Deixar para o semestre seguinte o ensino de registros e ponteiros;

Desenvolveu-se essas sugestões levando em conta a observação dos dois semestres (2016/1 e 2016/2), em conjunto com os dados coletados no AVA Moodle referente à participação, avaliação e frequência dos estudantes. Após apresentar estas propostas ao docente, o cronograma de 2017/1 foi construído de forma conjunta (docente e pesquisadora), atendendo as observações apontadas pela pesquisadora. Assim sendo, os CIP ganharam mais tempo no cronograma: conteúdos como

ponteiros e registros foram suprimidos do cronograma; manipulação de `String` saiu da terceira aula para ser ministrada em conjunto com vetores (lecionado na segunda metade do semestre); separou-se também o ensino de variáveis do ensino de entrada e saída de dados, deixando assim uma aula para cada conceito; estrutura de repetição separou-se em duas aulas, uma para `for` e outra para `while` e `do.. while`.

Ao analisar o decorrer do semestre a respeito da participação dos estudantes sobre a entrega das atividades, pode-se constatar que, assim como em 2016/1, a realização das atividades teóricas ainda apresenta um índice maior em relação às atividades práticas. O Gráfico 14 ilustra esses índices, apresentando também a quantidade de discentes presentes. Cabe ressaltar ainda que os exercícios práticos se tratam de uma lista de programas a serem desenvolvidos e, em muitos casos, os estudantes entregam apenas partes dessas tarefas.

Gráfico 14 - Comparação entre entrega de atividades práticas e teóricas



Fonte: A autora

Salienta, assim, a dificuldade que os estudantes apresentam no nível síntese da TOEDC, no qual é necessário conseguir fazer o uso de diversos CIP a fim de solucionar um problema. Ou seja, o estudante deve conseguir estruturar programas cada vez mais complexo, fazendo o uso dos conteúdos ensinados. Já a média geral da turma, daqueles que fizeram os questionários, foi de 72% levando em conta os

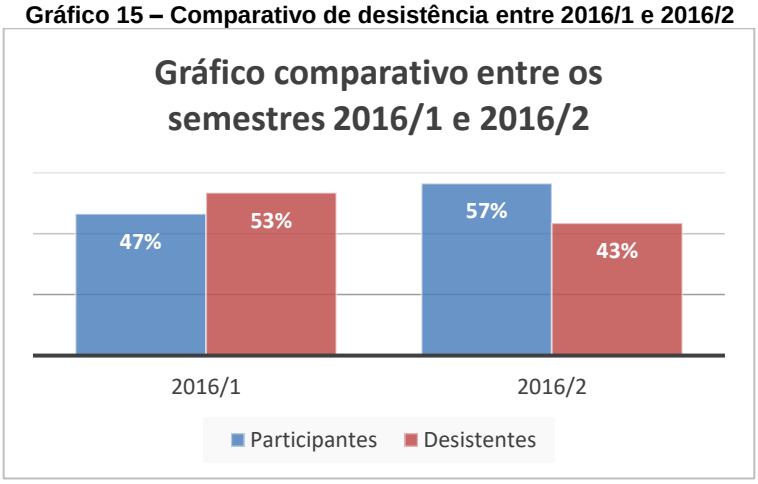
níveis conhecimento e compreensão. A seguir apresentar-se-á a análise referente aos estudantes em relação às aulas da disciplina LPI.

6.2.2.1. Os estudantes e a disciplina de acordo com a ConsProg

Assim como na primeira etapa do estudo de caso, foi aplicado no último dia de aula um questionário fechado, para ter informações sobre os estudantes, bem como a opinião deles sobre o semestre 2016/2. Uma vez que a disciplina LPI no segundo semestre de cada ano é ofertada no turno da manhã, o perfil dos discentes quanto ao trabalho se difere dos que estudam no turno da noite. Enquanto que no semestre de 2016/1 tem-se 79% de estudantes trabalhadores, em 2016/2 apenas 56,5% trabalhavam.

Consequentemente, de acordo com o levantamento, os discentes possuíam mais tempo livre para se dedicar à disciplina. Assim sendo, dos 46 estudantes matriculados, 19 aprovaram, 7 reprovaram e 20 tiveram frequência abaixo dos 75%. Apesar de 20 estudantes terem “desistido” da disciplina, ao comparar o semestre 2016/1 com o 2016/2 teve-se uma queda de 10% no índice de desistência da disciplina que pode ser visualizado no Gráfico 15.

Gráfico 15 – Comparativo de desistência entre 2016/1 e 2016/2



Fonte: A autora

O Gráfico 15 apresenta duas taxas para cada semestre: desistência - estudantes que possuem menos de 75% de frequência; participação, - estudantes que possuem frequência mínima. Neste sentido, analisando o Gráfico 15, pode-se inferir que a ConsProg auxiliou no aumento dos participantes, mesmo que o índice de

repetência tenha aumentado. Esse dado mostra que os estudantes conseguiram acompanhar melhor a disciplina, visto que muitos deles poderiam ter desistido logo após a primeira atividade avaliativa por não terem ido muito bem.

Alguns estudantes que possuíam dificuldades relataram que os conteúdos mais complicados e que a maioria deles não compreender foram: vetor, strings e ponteiros. Segundo eles, mesmo que o professor explicasse passo a passo esses conceitos, não conseguiam acompanhar. Um destes estudantes comentou: “ponteiros não entendi nada de nada, muito difícil de entrar na minha cabeça, por mais que o professor colocava exemplos no quadro”.

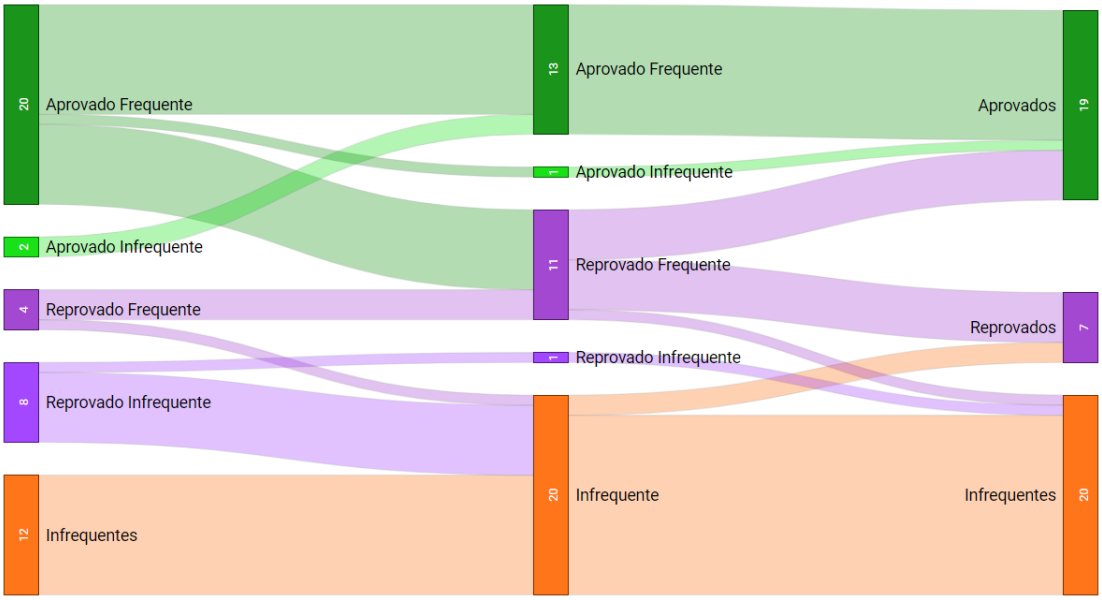
Relataram ainda que achavam as avaliações extensas e se desestimulavam, pois, em virtude de suas barreiras, levavam mais tempo para conseguir realizar uma questão ou parte dela. Logo, na maioria das vezes entregaram suas atividades avaliativas com erros ou sem fazer por não conseguir realizarem no tempo estipulado. Porém, comentaram que o professor corrigia as avaliações com cada um de seus estudantes, e que muitas vezes se saiam um pouco melhor, já que o docente os questionava sobre o erro e se soubessem responder, a questão não era zerada.

A realização da correção das provas com os estudantes não foi algo elaborado pela ConsProg, é uma metodologia utilizada pelo professor. Apesar de tomar bastante tempo, o docente fazia isso com todas as avaliações, dando prioridade aos discentes que apresentassem menor rendimento. Assim, o professor poderia verificar a dificuldade do estudante e auxiliá-lo na construção do conhecimento sobre os CIP.

Ao analisar a situação dos estudantes no decorrer do semestre através do Gráfico 16[23], é possível observar que 22 estudantes tinham frequência abaixo de 75% na primeira metade do semestre. Destes 22 infrequentes, dez fizeram a avaliação um e apenas dois atingiram conceito mínimo. Esse dado reforça a teoria desta pesquisa no que refere a importância dos CIP, visto que aqueles estudantes que não acompanharam as aulas introdutórias, não conseguiram apresentar bom rendimento na avaliação.

[23] Para o desenvolvimento deste gráfico, levou-se em conta a frequência dos estudantes, bem como suas atividades avaliativas. A primeira coluna refere-se às dez primeiras aulas e a avaliação um, a segunda coluna refere-se às dez últimas aulas e a segunda avaliação, a terceira coluna leva em conta as 20 aulas do semestre e todas as atividades avaliativas. Neste sentido, aprovado representa os estudantes com conceitos A, B ou C, reprovado trata-se dos que possuem conceito D, os infrequentes possuem menos de 75% de presença e os frequentes tem 75% ou mais de frequência.

Gráfico 16 – Acompanhamento de frequência e conceito dos estudantes (2016/2)



Fonte: A autora

Seguindo essa mesma análise (Gráfico 16), a primeira metade do semestre tem 24 estudantes frequentes, destes, 17 tiveram aproveitamento superior ao mínimo na primeira avaliação e foram aprovados na disciplina. Os outros dois aprovados na disciplina apresentaram baixa frequência na primeira metade do semestre, porém tiveram conceito acima do mínimo exigido para aprovação na atividade avaliativa 1. Neste sentido, os CIP mostram-se como uma base de extrema importância para o desenvolvimento da aprendizagem de programação. No tópico a seguir é descrito como foi feito o mapeamento dos CIP através dos dados gerados e coletados durante a pesquisa.

6.2.2.2. Mapeamento das relações e dependências entre os conteúdos

Para o mapeamento das relações e dependências entre os CIP foi utilizado o levantamento bibliográfico, as observações e análise dos dados do AVA Moodle dos semestres 2016/1 e 2016/2, bem como a RB criada por Vier et al.[24] (2015) sobre os conteúdos de programação orientada a objetos. Fez-se também consultas aos

[24] Pode ser verificado na Figura 4 da seção 5.5 Redes Bayesianas

especialistas da área de educação e informática, a fim de verificar se o mapeamento estava condizendo com as experiências e conhecimentos destes.

Neste contexto, foi observado que variáveis é a base inicial do conhecimento em programação, uma vez que ela é necessária para o desenvolvimento de qualquer programa que trabalhe com dados. Além disso foi possível inferir durante os semestres que estudantes que apresentam dificuldades em operadores relacionais, não aplicavam as comparações corretas na resolução de atividades sobre estrutura de controle, também necessários no aprendizado de estrutura de repetição.

Os operadores lógicos são utilizados em programas com estrutura de controle ou de repetição, que necessitam de mais de uma comparação. O Quadro 16 traz um exemplo de enunciado que pode ser solucionado fazendo o uso de operadores lógicos, uma vez que o estudante necessita verificar o mês e o dia do aniversário do usuário para que consiga dizer o signo do mesmo. Neste exemplo, seria necessário a utilização de uma estrutura de controle `if`, conforme pode-se observar na possível solução do problema no Código 5.

Quadro 16 - Exemplo de tarefa sobre estrutura de controle

Desenvolva um programa que mostre ao usuário seu signo levando em conta o dia e o mês de nascimento. Para isso, utilize a tabela abaixo:

Signo	Início	Fim
Áries	21/03	20/04
Touro	21/04	20/05
Gêmeos	21/05	20/06
Câncer	21/06	21/07
Leão	22/07	22/08
Virgem	23/08	22/09
Libra	23/09	22/10
Escorpião	23/10	21/11
Sagitário	22/11	21/12
Capricórnio	22/12	21/01
Aquário	22/01	19/02
Peixes	20/02	20/03

Fonte: a autora

Ao analisar o Código 5, tem-se o uso de diversos conteúdos tais como: variáveis, entrada e saída de dados, operadores relacionais, operadores lógicos e estrutura de controle. Há um destaque no Código 5, no bloco de comandos referente à estrutura de controle, para que se observe a utilização dos operadores lógicos e relacionais que estão em negrito. Percebe-se que para conseguir estruturar um

comando `if`, é de extrema importância que o estudante saiba aplicar os conceitos de operadores lógicos e relacionais.

Código 5 - Possível solução para a tarefa do Quadro 16

```
#include <stdio.h>
int main(void){
   int mes, dia;
   printf("Digite o mês de seu aniversário:");
   scanf("%d", &mes);
   printf("Digite o dia de seu aniversário:");
   scanf("%d", &dia);
   if(((mes==3)&&((dia>=21)&&(dia<=31)))||((mes==4)&&((dia>=1)&&(dia<=20))))
     printf("\nÁrie");
   else if(((mes==4)&&((dia>=21)&&(dia<=30)))||((mes==5)&&((dia>=1)&&(dia<=20))))
     printf("\nTouro");
   else if(((mes==5)&&((dia>=21)&&(dia<=31)))||((mes==6)&&((dia>=1)&&(dia<=20))))
     printf("\nGemeos");
   else if(((mes==6)&&((dia>=21)&&(dia<=30)))||((mes==7)&&((dia>=1)&&(dia<=21))))
     printf("\nCancer.");
   else if(((mes==7)&&((dia>=22)&&(dia<=31)))||((mes==8)&&((dia>=1)&&(dia<=22))))
   printf("\nLeao");
   else if(((mes==8)&&((dia>=23)&&(dia<=31)))||((mes==9)&&((dia>=1)&&(dia<=22))))
     printf("\nVirgem");
   else if(((mes==9)&&((dia>=23)&&(dia<=30)))||((mes==10)&&((dia>=1)&&(dia<=22))))
    printf("\nLibra");
   else if(((mes==10)&&((dia>=23)&&(dia<=31)))||((mes==11)&&((dia>=1)&&(dia<=21))))
     printf("\nEscorpiao");
   else if(((mes==11)&&((dia>=22)&&(dia<=30)))||((mes==12)&&((dia>=1)&&(dia<=21))))
     printf("\nSagitario");
   else if(((mes==12)&&((dia>=22)&&(dia<=31)))||((mes==1)&&((dia>=1)&&(dia<=20))))
     printf("\nCapricornio");
   else if(((mes==1)&&((dia>=21)&&(dia<=31)))||((mes==2)&&((dia>=1)&&(dia<=19))))
     printf("\nAquario");
   else if(((mes==2)&&((dia>=20)&&(dia<=29)))||((mes==3)&&((dia>=1)&&(dia<=20))))
     printf("\nPeixes");
   else
     printf("\nDia ou mês inválido");
   return 0;
}
```

Fonte: A autora

Essa mesma importância encontra-se no aprendizado de estrutura de repetição, visto que o bloco de comandos desta repete até que uma dada comparação seja verdadeira. Para ilustrar essa necessidade, no Quadro 17 é exposto um exemplo de atividade no qual o estudante precisa utilizar estrutura de repetição.

Quadro 17 - Exemplo de enunciado sobre estrutura de repetição

Desenvolva um programa que peça um valor inteiro para o usuário e repita esse processo até que seja digitado zero ou um valor maior que 100.

Fonte: A autora

Uma possível resposta para a tarefa do Quadro 17 seria o Código 6, que fez a aplicação dos conteúdos: variáveis, entrada e saída de dados, operadores lógicos e operadores relacionais. Em destaque encontra-se o bloco de códigos referente a estrutura de repetição, e em negrito os operadores lógicos e relacionais. É possível observar que, assim como na estrutura de controle, para que se possa aplicar o conteúdo referente à estrutura de repetição, é necessário que o estudante saiba utilizar os operadores lógicos e relacionais.

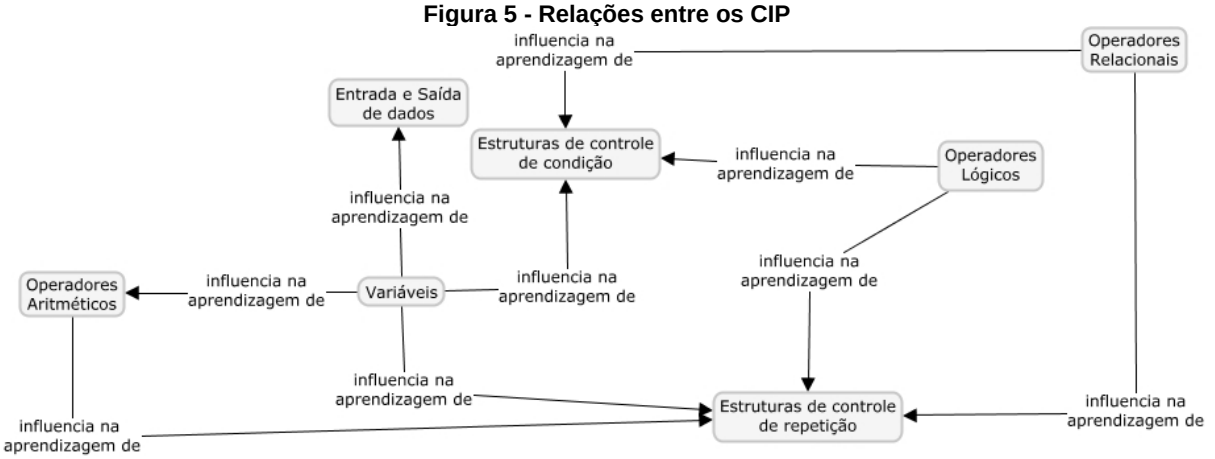
Código 6 - Possível solução para a atividade do Quadro 17

```
#include <stdio.h>
int main(void){
    int valor;
    do{
        printf("\n Digite um valor inteiro:");
        scanf("%d", &valor);
    } while((valor!=0)&&(valor<100));
    return 0;
}
```

Fonte: A autora

Assim sendo, desenvolveu-se o mapeamento das relações entre os CIP, que pode ser observado na Figura 5. Este mapeamento serviu de base para estrutura da RB desenvolvida para a ConsProg.

Figura 5 - Relações entre os CIP



Fonte: A autora

Por meio desta Figura 5, é possível identificar as relações exemplificadas no presente tópico. Na subseção a seguir tem-se a análise da terceira etapa do estudo de caso.

6.2.3 Terceira etapa do estudo de caso

Última etapa do estudo de caso, nela coletou-se dados em uma nova turma na disciplina LPI da mesma Instituição, no primeiro semestre de 2017. O intuito dessa era conhecer a opinião do docente a respeito da ConsProg. Logo, foi solicitado ao docente da disciplina um relatório a respeito da experiência do uso da proposta pedagógica. Para atender as necessidades da pesquisa, algumas questões (Quadro 18) foram entregues ao professor para que pontos importantes para a pesquisa fossem respondidos no relatório.

Quadro 18 - Questões entregues ao docente

1) Como foi criar materiais de acordo com a ConsProg?
2) A adequação dos materiais foi propícia ao aprendizado dos estudantes?
3) Trazer exemplos do cotidiano dos estudantes facilitou o entendimento do conteúdo por parte deles?
4) Dar aula construtivista (iniciando do concreto indo para o mais abstrato) faz com que os estudantes interajam mais?
5) É possível acompanhar o desenvolvimento de aprendizagem dos estudantes fazendo o uso da ConsProg?
6) Quais dificuldades encontrou ao utilizar a ConsProg?
7) Quais foram os pontos positivos na ConsProg?
8) Notou mudanças no rendimento e/ou participação dos estudantes?

Fonte: A autora

Uma vez que a presente subseção tratará de dois aspectos, a mesma está dividida em duas partes: uma que analisa o semestre 2017 através dos dados coletados no AVA Moodle sobre a participação dos estudantes na disciplina, a resolução das atividades e dos questionários; e outra que trará a análise do relato do professor.

6.2.3.1. ConsProg e os Estudantes

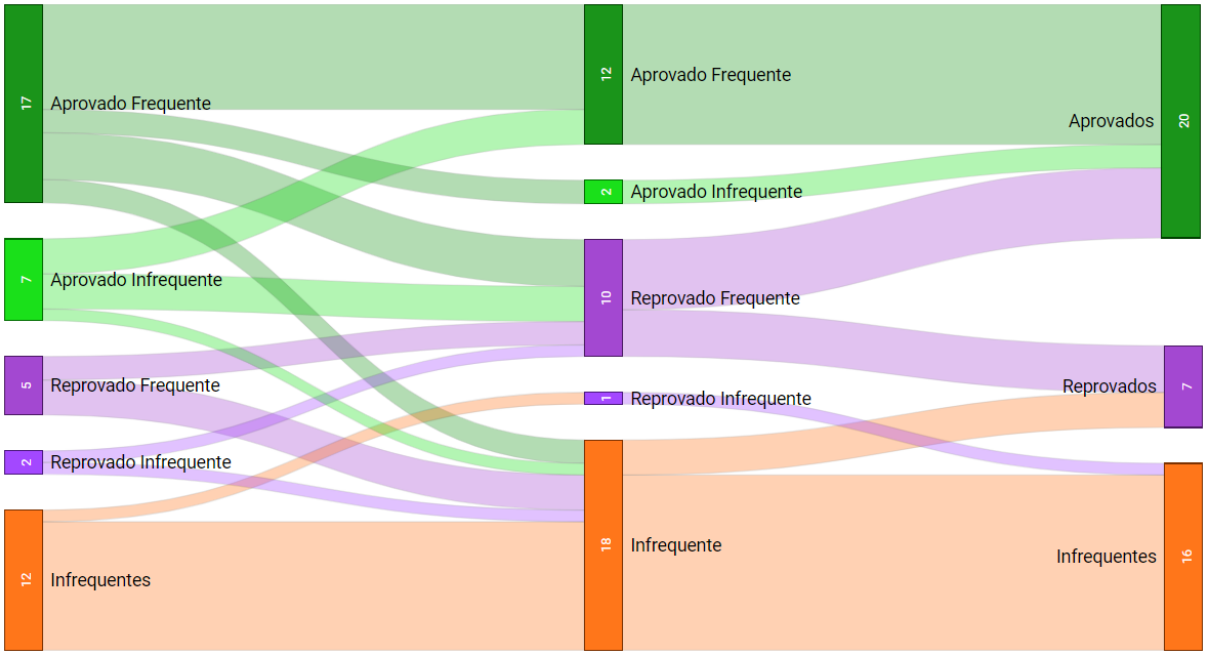
O questionário fechado realizado nos semestres anteriores também foi efetuado nesta etapa do estudo de caso, porém o professor da disciplina que aplicou o mesmo. De acordo com os dados levantados, trata-se de uma turma de estudantes trabalhadores, de faixas etárias diversas, que apresentam mais de 50% de novatos em programação. A maioria dos discentes possuíam menos de duas horas por semana para dedicar-se à disciplina.

Muitos estudantes deixaram comentários no questionário, no qual a maioria relatou ter gostado de não terem aprendido ponteiros durante o semestre, pois tiveram

mais tempo para aprender os CIP. Alguns descreveram que o conteúdo que tiveram maior dificuldade em aprender foi funções. Outros sentiram falta de ponteiros, porém disseram compreender que se trata de um conceito complexo e que em uma aula não é possível assimilar esse conteúdo. Percebe-se através das narrativas dos estudantes, que para eles o tempo para aprender é importante, o que vai ao encontro à teoria construtivista, no qual deve-se respeitar o tempo do desenvolvimento cognitivo do estudante para que haja aprendizado (PIAGET, 1983).

Ao analisar os discentes quanto sua participação e desempenho na disciplina, foi possível constatar que um estudante que não apresentou bom rendimento na segunda metade do semestre foi aprovado. Este caso ocorreu por decisão do professor da disciplina, que por meio de uma análise qualitativa do desempenho do estudante ao longo do semestre demonstrou bom rendimento. Além disso, o estudante não apresentou problema de frequência na primeira parte da disciplina, estando assim acima dos 75% previstos na Lei 9.394 (BRASIL, 1996).

Gráfico 17 - Acompanhamento frequência e conceito dos estudantes (2017/1)



Fonte: A autora

O Gráfico 17 refere-se a análise da situação dos estudantes, levando em conta a frequência e as atividades avaliativas em três momentos: dez primeiras aulas e avaliação 1 (primeira coluna); dez aulas finais e avaliação 2 (segunda coluna); todas as 20 aulas e as duas avaliações (terceira coluna). No Quadro 19 é possível conferir o que cada situação levou em conta.

Quadro 19 - Dados considerados para cada situação

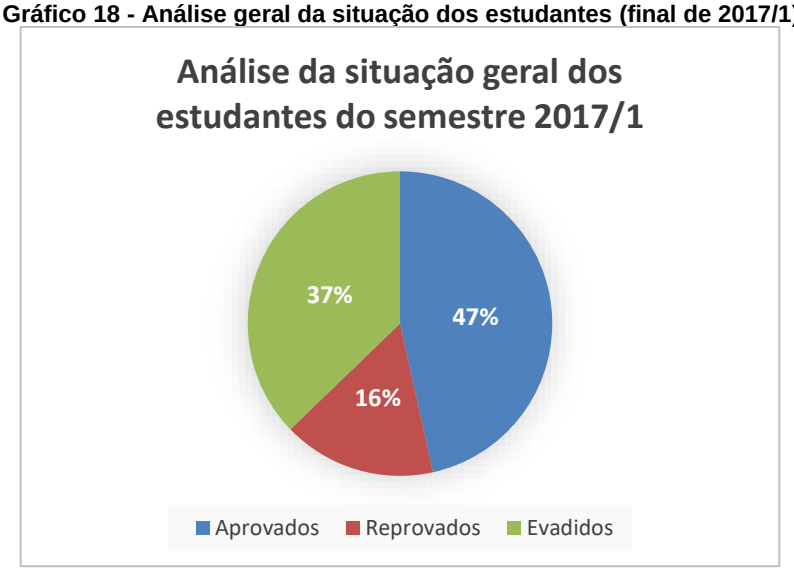
Situação	O que representa
Aprovado frequente	Frequência igual ou acima de 75% e conceito A, B ou C
Aprovado infrequente	Frequência abaixo de 75% e conceito A, B ou C
Reprovado frequente	Frequência igual ou acima de 75% e conceito D
Reprovado infrequente	Frequência abaixo de 75% e conceito D
Infrequente	Frequência abaixo de 75% e não realizou a avaliação
Aprovado	Frequência igual ou acima de 75% e conceito A, B ou C
Reprovado	Frequência igual ou acima de 75% e conceito D

Fonte: A autora

Assim sendo, no Gráfico 17 pode-se verificar que 20 estudantes foram aprovados na disciplina, sendo que desses, 18 obtiveram conceitos acima do mínimo na primeira metade do semestre. Neste sentido, o que inicialmente tratava-se de uma teoria e foi constatado na segunda etapa do estudo de caso, através desses dados levantados vem reforçar a importância dos CIP para o bom desenvolvimento do estudante no aprendizado de programação. Logo, acredita-se que seja de extrema importância registrar aqui que os docentes que ministram DIP devem dar uma atenção especial nos CIP, podendo assim reduzir as dificuldades dos estudantes ao longo da disciplina, bem como nas demais relacionadas com programação.

Nessa etapa do estudo de caso os estudantes tiveram mais tempo para aprender os CIP, uma vez que foram feitas algumas alterações no cronograma da disciplina, o que auxiliou na redução do índice de evasão na disciplina. No Gráfico 18 pode-se observar que apenas 37% dos estudantes evadiram a disciplina, acredita-se que esse resultado pode ser alcançado em virtude do uso da ConsProg, em conjunto com o novo cronograma.

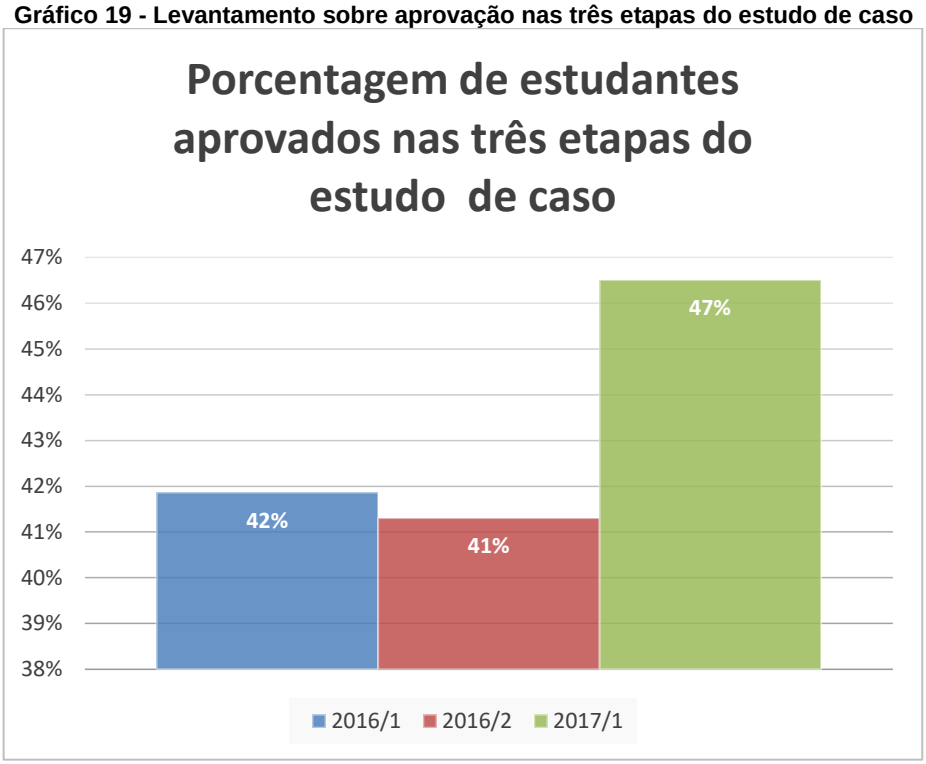
Gráfico 18 - Análise geral da situação dos estudantes (final de 2017/1)



Fonte: A autora

Das três etapas do estudo de caso, esta foi a que apresentou melhor resultado no quesito aprovação. No Gráfico 19, a porcentagem de estudantes aprovados em cada semestre analisado (2016/1, 2016/2, 2017/1) é apresentada, deixando visível a melhora do índice de aprovação em 2017/1, no qual fez-se o uso da ConsProg.

Gráfico 19 - Levantamento sobre aprovação nas três etapas do estudo de caso



Fonte: A autora

Após as análises dos dados dessa etapa, acredita-se que a ConsProg contribuiu para processo educacional da turma, uma vez que auxiliou o docente no acompanhamento do desenvolvimento de aprendizagem dos estudantes. A seguir, apresenta-se a análise do relato do professor da disciplina quanto ao uso da ConsProg em suas aulas.

6.2.3.2. ConsProg e o Docente

O semestre 2017/1 foi ministrado pelo mesmo professor dos semestres anteriores, sendo assim, o mesmo já possuía experiência com o uso da proposta pedagógica desenvolvida. Para o docente as dificuldades dos estudantes apontadas pela ConsProg foram importantes, visto que os discentes pouco interagiam durante a

correção das atividades práticas. Neste sentido, segundo o docente, o acompanhamento do desenvolvimento dos estudantes através dos questionários elaborados de acordo com a ConsProg indicou a ele “quais conteúdos/aspecto deveriam ser enfatizados na resolução dos exercícios”. Logo, a proposta apontou ao docente o que seria necessário trabalhar com a turma para que esta seguisse o aprendizado de forma construtiva, visto que os CIP são dependentes entre si. Caso o professor não obtivesse essas informações, esses conteúdos poderiam não ser revistos e os estudantes seguiriam com as dúvidas, dificultando assim o aprendizado dos conteúdos posteriores.

Vale ressaltar que a ConsProg faz proposições sobre o desenvolvimento do aprendizado, assim, é necessário também que o professor investigue sobre a dificuldade junto a seus estudantes. Um caso omisso ocorreu no semestre em questão, de acordo com o professor, a ConsProg sugeria que um discente tinha dificuldades com operadores relacionais, e em sala o estudante não apresentava bloqueios com o mesmo. Assim sendo, o professor foi averiguar o que ocorria, e de acordo com o estudante, ele cometia esses equívocos por falta de atenção, e não por dificuldades.

No relato do professor destaca-se um caso sobre o uso da ConsProg:

> [...] a confirmação obtida através do acompanhamento das atividades e o resultado final do aluno garantiu maior segurança para eu poder avaliar quais os alunos que necessitavam realizar a avaliação de recuperação e quais poderiam ser dispensados da mesma e quais tinham não tinham apropriado conteúdos suficientes para seguir para o próximo semestre.

Pode-se dizer que a ConsProg auxiliou o professor não apenas durante o semestre, mas também trouxe para o mesmo, evidências a respeito do nível de aprendizado dos CIP, dando-lhe mais segurança na aprovação ou não dos estudantes. Observa-se então que a ConsProg teve para ele uma utilização na qual não havia sido pensada na proposta, e que contribuiu na tomada de algumas decisões.

Ademais, o docente relatou que “o uso da abordagem possibilitou reavaliar o material e realizar adequações no mesmo de forma a implementar a abordagem”. Para ele esse processo de adequação foi simples, uma vez que já possuía materiais prontos da disciplina, podendo então reorganizá-los de forma a atender as especificações da ConsProg. Neste sentido, destaca-se que o docente não precisou abrir mão dos materiais que já possuía, visto que a ConsProg apenas propõe formas

que podem facilitar o processo de ensino-aprendizagem embasadas em teorias consistentes.

Através da adaptação dos materiais, o professor acredita que “com inclusão de exemplos concretos do cotidiano tornou a apresentação dos conceitos mais ‘leve’, tornando o ambiente da sala de aula menos ‘tenso’”. Ainda observou “maior interação na apresentação dos exemplos concretos, porém ao adentrar nos conceitos mais específicos, alunos ficam menos participativos”. Com esse relato identifica-se que quanto mais próximo à explicação for daquilo que o estudante já domina/conhece, mais à vontade se sente em participar. Isso ocorre, pois, as estruturas cognitivas fazem relações com o que já se conhece, deixando menos abstrato e facilitando o processo de desenvolvimento da aprendizagem.

Apenas uma limitação foi apontada pelo docente, este já fazia o uso de questionários em suas aulas, porém as questões não eram elaboradas de acordo com os níveis da TOEDC e os estudantes podiam fazer ilimitadas vezes. Já os questionários desenvolvidos de acordo com a ConsProg servem para verificar o desenvolvimento de aprendizagem do estudante, assim, permitindo-o responder a mesma questão mais vezes, fará com que o mesmo tente até acertar, não significando necessariamente aprendizado. Com isso, os questionários da ConsProg são limitados a uma única tentativa, para que se possa inferir sobre o real aprendizado do estudante.

Logo, para o docente essa limitação mudou a perspectiva do questionário, que segundo ele a permissão de tentar ilimitadas vezes era para que o estudante dedicasse um tempo maior estudando programação. Assim sendo, o professor acredita que essa limitação poderá ser sanada posteriormente, e segundo ele “a perspectiva do ConsProg possibilita a obtenção de informações mais ricas e interessantes”.

Acredita-se que essa questão de permitir que o estudante realize o questionário sobre um determinado conteúdo mais vezes poderá ser solucionada com a criação de uma base de questões maior. Com isso, ao invés de serem as mesmas questões oferecidas, pode ser sorteado um novo conjunto de questões de modo a estimular o estudante a aprimorar-se. Esse sorteio poderia levar em conta as hipóteses levantadas na RB, já que esta verifica o desenvolvimento da aprendizagem dos estudantes.

Assim, as hipóteses calculadas pela rede RB poderiam servir de base de controle para o sorteio das questões, selecionando aquelas que sejam relacionadas

às dificuldades dos discentes. Agregado a essa base de questões, poderiam ser incorporadas técnicas de gamificação para tentar envolver mais o discente na busca do saber. Na sequência encontra-se a seção que explica como foi criada a RB utilizada na ConsProg.

6.3 PROCESSO DE CRIAÇÃO DA REDE BAYESIANA

A estrutura da RB foi elabora por dois especialistas, sendo um da área da educação e um da área de informática, que se reuniram diversas vezes para discutir sobre as necessidades de relações entre os CIP, levando em conta a TOEDC. Neste sentido, buscaram-se identificar quais seriam os dados relevantes sobre o ensino-aprendizagem de programação, para que estas fossem representadas na RB. Esses dados foram relacionados com os coletados durante os estudos de caso, a fim de analisar se as variáveis atenderiam as necessidades do professor e se auxiliaria no acompanhamento do processo de aprendizagem dos estudantes. Na sequência da presente seção será apresentado como foi realizado o desenvolvimento da RB, bem como alguns experimentos realizados.

6.3.1 Métodos para a estruturação da Rede Bayesiana

O modelo estrutural final da RB foi desenvolvido fazendo o uso dos conhecimentos dos especialistas, fontes bibliográficas, dados coletados no estudo de caso, bem como algumas relações sugeridas pelo algoritmo GTT. Cabe lembrar que o GTT se trata de um algoritmo que auxilia na estruturação da RB, levando em conta a entrada de dados (CHENG, 1998), que neste caso tratam-se das respostas dos questionários coletadas no AVA Moodle. Esses dados são utilizados pelo GTT para a aprendizagem dos nós através das variáveis da base de dados, bem como as possíveis relações existentes entre os nós.

Dessa forma, no Quadro 20 apresentam-se as variáveis utilizadas na base de dados, assim como o nome utilizado na RB. Essas variáveis referem-se aos CIP, juntamente com os níveis da TOEDC para cada um dos conteúdos, resultando assim em 42 nós (seis níveis da TOEDC para cada um dos sete conteúdos – variáveis,

entrada e saída de dados, operadores aritméticos, operadores relacionais, operadores lógicos, estrutura de controle e estrutura de repetição).

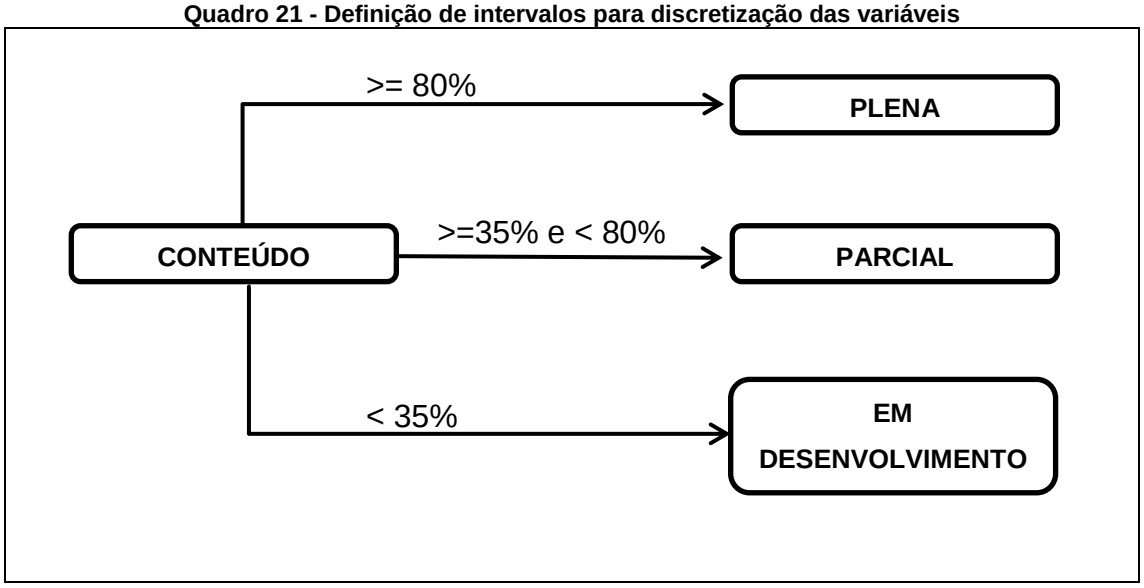
Quadro 20 - Nome dado aos nós utilizados na RB

Nó (variável) CIP + TOEDC	Nome utilizado na RB
Variáveis Conhecimento	VAR_CON
Variáveis Compreensão	VAR_COM
Variáveis Aplicação	VAR_APLI
Variáveis Análise	VAR_ANA
Variáveis Síntese	VAR_SIN
Variáveis Avaliação	VAR_AVA
Entrada e saída de dados Conhecimento	ES_CON
Entrada e saída de dados Compreensão	ES_COM
Entrada e saída de dados Aplicação	ES_APLI
Entrada e saída de dados Análise	ES_ANA
Entrada e saída de dados Síntese	ES_SIN
Entrada e saída de dados Avaliação	ES_AVA
Operadores aritméticos Conhecimento	OP_ARI_CON
Operadores aritméticos Compreensão	OP_ARI_COM
Operadores aritméticos Aplicação	OP_ARI_APLI
Operadores aritméticos Análise	OP_ARI_ANA
Operadores aritméticos Síntese	OP_ARI_SIN
Operadores aritméticos Avaliação	OP_ARI_AVA
Operadores relacionais Conhecimento	OP_REL_CON
Operadores relacionais Compreensão	OP_REL_COM
Operadores relacionais Aplicação	OP_REL_APLI
Operadores relacionais Análise	OP_REL_ANA
Operadores relacionais Síntese	OP_REL_SIN
Operadores relacionais Avaliação	OP_REL_AVA
Operadores lógicos Conhecimento	OP_LOG_CON
Operadores lógicos Compreensão	OP_LOG_COM
Operadores lógicos Aplicação	OP_LOG_APLI
Operadores lógicos Análise	OP_LOG_ANA
Operadores lógicos Síntese	OP_LOG_SIN
Operadores lógicos Avaliação	OP_LOG_AVA
Estrutura de controle Conhecimento	EST_CON_CON
Estrutura de controle Compreensão	EST_CON_COM
Estrutura de controle Aplicação	EST_CON_APLI
Estrutura de controle Análise	EST_CON_ANA
Estrutura de controle Síntese	EST_CON_SIN
Estrutura de controle Avaliação	EST_CON_AVA
Estrutura de repetição Conhecimento	EST_REP_CON
Estrutura de repetição Compreensão	EST_REP_COM
Estrutura de repetição Aplicação	EST_REP_APLI
Estrutura de repetição Análise	EST_REP_ANA
Estrutura de repetição Síntese	EST_REP_SIN
Estrutura de repetição Avaliação	EST_REP_AVA

Fonte: A autora

Neste sentido, inicialmente foi necessária a seleção das variáveis (listadas no Quadro 20) que auxiliassem a responder o seguinte caso: “Quais possíveis dificuldades serão apresentadas pelo estudante ‘x’, a partir das respostas indicadas nos questionários? ”. Logo, para melhor identificação do processo de aprendizagem, as variáveis (nós) foram discretizadas e classificadas em “pleno” (se o estudante respondeu corretamente mais de 80%), “parcial” (se o estudante respondeu corretamente entre 35% e 79%) e “em desenvolvimento” (se o estudante respondeu corretamente menos de 34%). No Quadro 21 é possível visualizar essa classificação, que foi proposta com base nas atividades, questionários e avaliações realizadas pelos estudantes.

Quadro 21 - Definição de intervalos para discretização das variáveis



Fonte: A autora

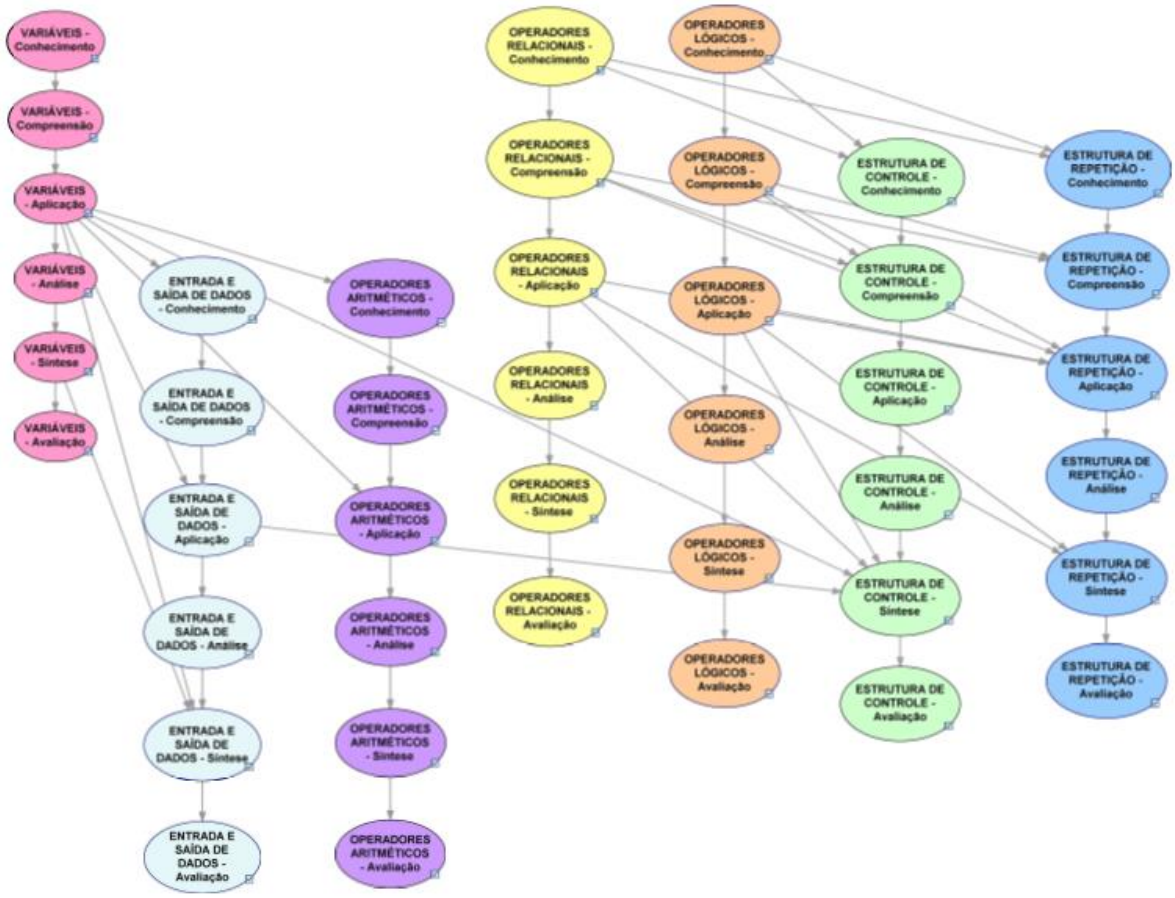
Depois da classificação determinada, o conjunto de dados coletados do AVA Moodle (referente às resoluções dos questionários criados de acordo com a ConsProg) foi reorganizado de forma a servir de entrada para a criação da RB. Assim, os dados foram categorizados em “plena”, “parcial” e “em desenvolvimento” e armazenados em um arquivo do tipo csv[25] para posterior utilização. Para tal, foram considerados os dados daqueles estudantes que responderam mais do que 50% dos questionários, e que não fossem repetentes, totalizando 20 estudantes. Tendo em vista que alguns destes não haviam realizados todos os questionários, os dados

[25] *Comma Separated Values*, trata-se de um arquivo texto com dados oriundos de uma planilha eletrônica a ser usado em aplicativos.

faltantes foram estimados pelos especialistas, levando em conta as atividades práticas, atividades avaliativas e os questionários respondidos pelo estudante.

Os dados foram tabulados em uma planilha eletrônica, e utilizados como entrada para a estimativa da estrutura da RB. Em um primeiro momento esses dados foram analisados de forma manual a fim de construir uma versão estrutural da RB pelos especialistas. Na Figura 6, é possível observar essa versão que foi utilizada posteriormente para estabelecer as restrições na RB.

Figura 6 - Primeira versão da Rede Bayesiana



Fonte: A autora

Para desenvolver a estrutura da Figura 6 utilizou-se o aplicativo GeNIe Modeler versão 2.1[26]. Esse programa foi escolhido por ser gratuito para estudantes, permitindo "construir modelos de qualquer tamanho e complexidade, limitados, apenas, pela capacidade da memória operacional do seu computador" (BayesFusion, 2017, p. 31).

[26] "Ambiente de desenvolvimento para a construção de modelos gráficos de decisão-decisão. " (BayesFusion, 2017, p. 30).

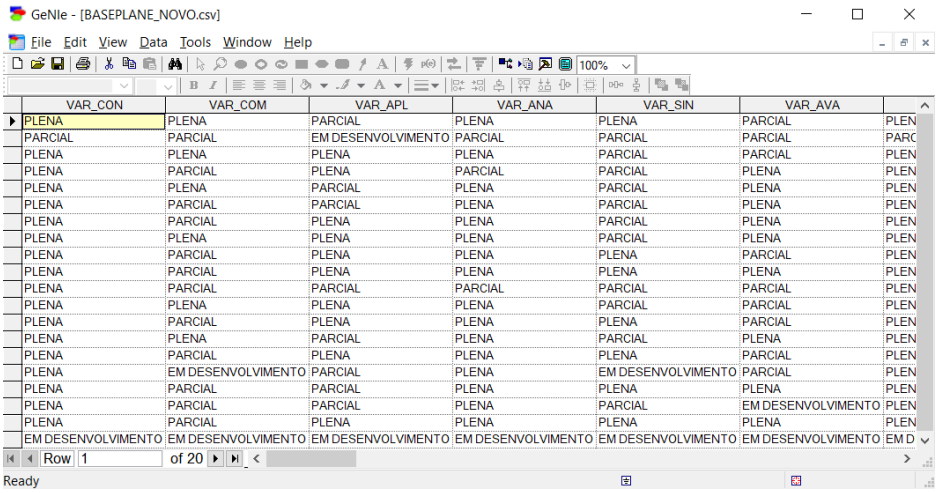
Outros aplicativos também foram testados: Bayes Server versão 7.18, Hugin Lite versão 8.5, Netica versão 6.03 e MBNX versão 1.4; assim como o GeNIe esses também apresentam uma versão gratuita, porém limitam a quantidade de nós que se pode utilizar.

Com o intuito de verificar possíveis relações não observadas manualmente, fez-se a utilização de um algoritmo no GeNIe para a construção de uma RB automática, levando em conta as restrições dadas pelos especialistas. Os algoritmos para aprendizado de estrutura podem ser classificados nos que utilizam um sistema de busca com heurística e os que constroem a rede analisando a relação de dependência entre os nós. Como exemplo do primeiro tipo tem-se Bayesian Search (COOPER, 1992), enquanto do segundo tipo tem-se Greedy Thick Thinning (CHENG, 1998). Em geral, os algoritmos que fazem uso de heurística têm uma complexidade de tempo razoável no pior caso, mas não conseguem encontrar a melhor solução devido à natureza da heurística. Contudo, os algoritmos da segunda classificação necessitam que o conjunto de dados seja grande de forma que os testes realizados na análise de dependência sejam confiáveis, mas possuem como vantagem estarem assintoticamente corretos e terem complexidade $O(N^2)$.

Neste sentido, optou-se pelo algoritmo *Greedy Thick Thinning* (GTT), por construírem as relações levando em conta as dependências dos nós. De acordo com Cheng (1998), o algoritmo GTT é composto por três etapas: esboço, engrossamento e afinamento. Na primeira etapa, um esboço da rede é criado por meio da medição da proximidade de cada par de nós. Na segunda, arcos são adicionados entre pares que possuem dependência entre eles. Na última etapa, testes de independência condicional são realizados de forma que os arcos possam ser removidos à medida que os nós sejam independentes um do outro.

Fazendo o uso do software GeNIe, fez-se a abertura do arquivo “csv” já com os dados discretizados, este arquivo é necessário para que o algoritmo possa aprender sobre os dados e assim gerar uma estrutura. Com o arquivo aberto no aplicativo, conforme ilustra a Figura 7, iniciou-se o processo de aprendizado da RB por intermédio do algoritmo GTT. No aplicativo GeNIe, o primeiro passo do aprendizado da RB é a escolha do algoritmo, e de acordo com o escolhido pode-se definir algumas configurações que são chamados de atributos.

Figura 7 - Conjunto de dados aberto no aplicativo GeNIe

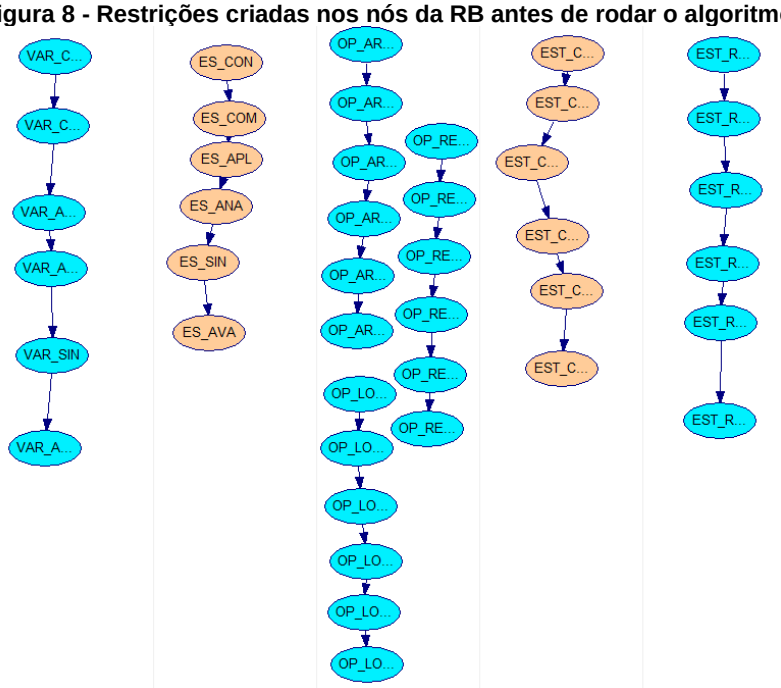


VAR_CON	VAR_COM	VAR_APL	VAR_ANA	VAR_SIN	VAR_AVA	
PLENA	PLENA	PARCIAL	PLENA	PLENA	PARCIAL	PLEN
PARCIAL	PARCIAL	EM DESENVOLVIMENTO	PARCIAL	PARCIAL	PARCIAL	PARC
PLENA	PLENA	PLENA	PLENA	PARCIAL	PARCIAL	PLEN
PLENA	PARCIAL	PLENA	PARCIAL	PARCIAL	PLENA	PLEN
PLENA	PLENA	PARCIAL	PLENA	PARCIAL	PLENA	PLEN
PLENA	PARCIAL	PARCIAL	PLENA	PARCIAL	PLENA	PLEN
PLENA	PARCIAL	PLENA	PLENA	PARCIAL	PLENA	PLEN
PLENA	PLENA	PLENA	PLENA	PARCIAL	PLENA	PLEN
PLENA	PARCIAL	PLENA	PLENA	PLENA	PARCIAL	PLEN
PLENA	PARCIAL	PLENA	PLENA	PLENA	PLENA	PLEN
PLENA	PARCIAL	PARCIAL	PARCIAL	PARCIAL	PARCIAL	PLEN
PLENA	PLENA	PLENA	PLENA	PARCIAL	PARCIAL	PLEN
PLENA	PARCIAL	PLENA	PLENA	PLENA	PARCIAL	PLEN
PLENA	PLENA	PARCIAL	PLENA	PARCIAL	PLENA	PLEN
PLENA	PARCIAL	PLENA	PLENA	PLENA	PARCIAL	PLEN
PLENA	EM DESENVOLVIMENTO	PARCIAL	PLENA	EM DESENVOLVIMENTO	PARCIAL	PLEN
PLENA	PARCIAL	PARCIAL	PLENA	PLENA	PLENA	PLEN
PLENA	PARCIAL	PARCIAL	PLENA	PARCIAL	EM DESENVOLVIMENTO	PLEN
PLENA	PARCIAL	PLENA	PLENA	PLENA	PLENA	PLEN
EM DESENVOLVIMENTO	EM DESENVOLVIMENTO	EM DESENVOLVIMENTO	EM DESENVOLVIMENTO	EM DESENVOLVIMENTO	EM DESENVOLVIMENTO	EM D

Fonte: A autora

Nesse processo da configuração de atributos é possível incluir algumas restrições entre as relações que serão criadas nos nós através da opção *Background Knowledge*. Na Figura 8 é possível verificar todas as restrições criadas, que teve o intuito de respeitar a hierárquica do desenvolvimento cognitivo criada por Bloom et al. (1972). Por exemplo, a primeira lista de nós representa o conteúdo variáveis em conjunto com TOEDC, no qual criou-se as seguintes restrições: VAR_CON (variáveis conhecimento) influenciando o nó VAR_COM (variáveis compreensão), que influenciará o nó VAR_APL (variáveis aplicação), e assim por diante.

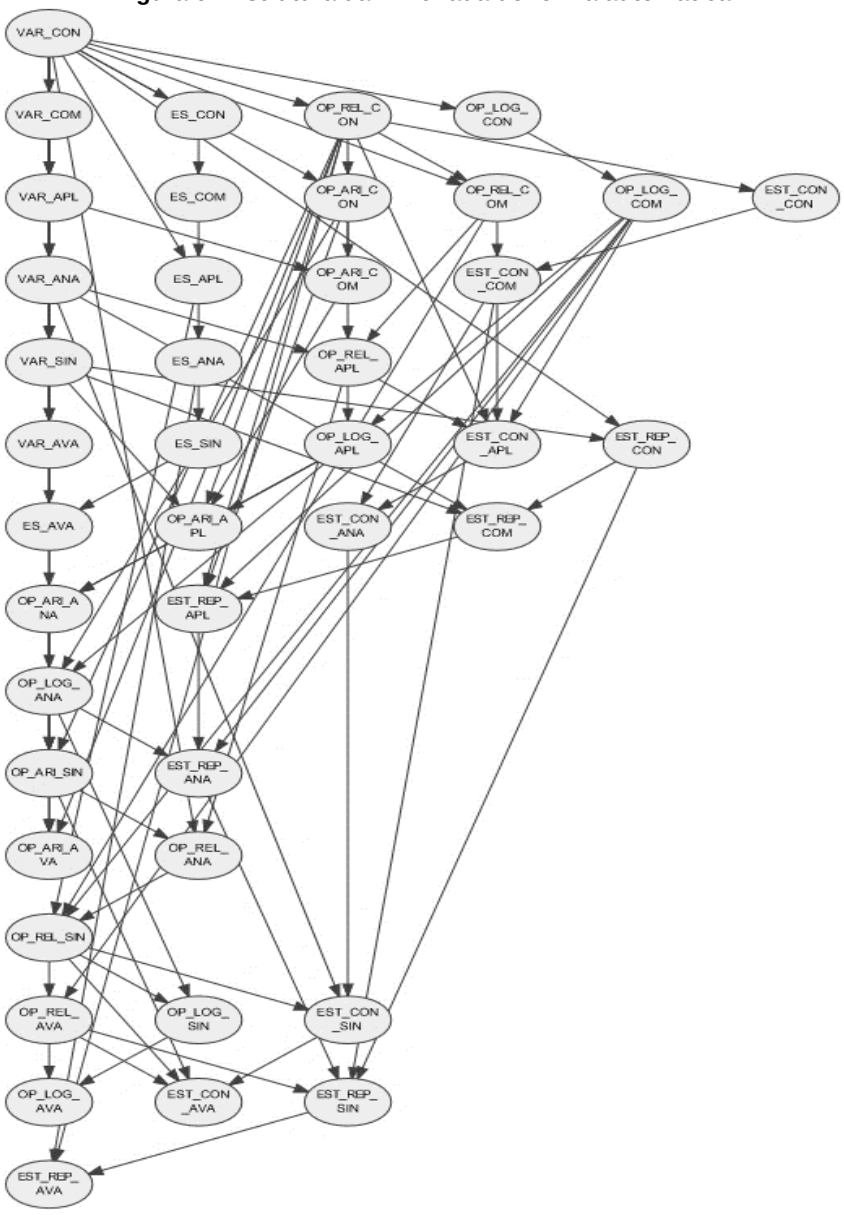
Figura 8 - Restrições criadas nos nós da RB antes de rodar o algoritmo



Fonte: A autora

Além disso, é possível escolher o número máximo de nós pais, ou seja, o máximo de relação que um nó pode ter. Neste sentido, determinou-se o máximo de quatro nós pais, visto que em outros testes feitos esse valor foi o que melhor atendeu as necessidades dos especialistas (relações entre os CIP e TOEDC). É necessário também escolher a prioridade métrica de pontuação da RB, que se refere a qualidade da RB em representar o conjunto dos dados. Logo, foi escolhida a prioridade *K2* para a métrica de pontuação da que de acordo com Souza (2010, p. 63), "é considerado um dos mais importantes dentre todos os algoritmos que se referenciam a busca de pontuação para estimação de estrutura".

Figura 9 - Estrutura da RB criada de forma automática

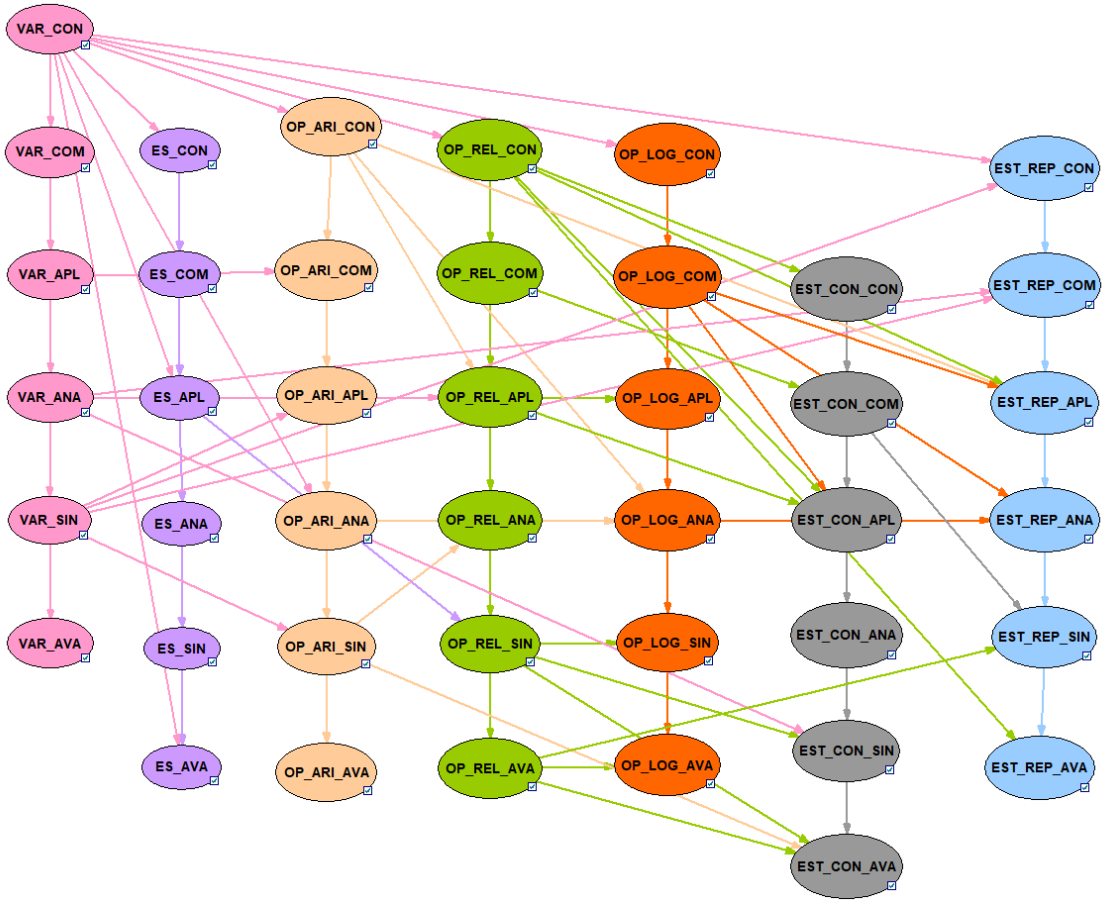


Fonte: A autora

Assim sendo, a prioridade *K2* inicia ordenando os nós para criar uma estrutura acíclica, pesquisando entre $2^{n(n-1)/2}$ possibilidades de estrutura da RB analisando qual maximiza a função *score* (SOUZA, 2010). Após esses procedimentos, utilizando o software GeNIe com o algoritmo de prioridade K2 em conjunto com o algoritmo GTT foi produzida a estrutura que pode ser observada na Figura 9.

Portanto, a estrutura resultante da prioridade K2 com o algoritmo GTT e as restrições iniciais (Figura 9) foi analisada pelos especialistas, que julgaram algumas relações desnecessárias, uma vez que não geravam informações coerentes ou compreensíveis. Sendo assim, alguns arcos foram removidos para que a RB atendesse a ConsProg, bem como a expectativa dos especialistas, resultando na RB representada na Figura 10. Vale ressaltar que o software GeNIe gera a RB no padrão de cores e formato apresentado na Figura 9, para facilitar a leitura da mesma, optou-se em reorganizá-la utilizando-se cores (Figura 10) com o intuito de tornar mais clara a identificação dos conteúdos, nós e suas relações.

Figura 10 - Estrutura da RB da ConsProg



Fonte: A autora

Assim que a estrutura da RB ficou definida, alguns experimentos foram realizados através de inferências. Na subseção a seguir é possível conhecer alguns desses experimentos realizados.

6.3.2 Experimentos realizados

Os experimentos foram realizados com os dados coletados em 2017/1, e teve o intuito de verificar se a RB apresenta os resultados compatíveis com o desenvolvimento real dos estudantes. Um dos experimentos realizados foi com o Estudante 1 (E1) que apresentou os dados do Quadro 22, que tenta traduzir o desenvolvimento de aprendizagem em variáveis e entrada e saída de dados. Utilizou-se os dados das variáveis como evidências na RB desenvolvida no sistema GeNIe, e após, fez-se o *update* para que fosse calculado a probabilidade de aprendizagem de entrada e saída de dados.

Quadro 22 – Evidências parciais do E1

ESTUDANTE 1					
VAR_CON	**VAR_COM**	**VAR_APL**	**VAR_ANA**	**VAR_SIN**	**VAR_AVA**
PLENA	PLENA	PLENA	PLENA	PARCIAL	PARCIAL
ES_CON	**ES_COM**	**ES_APL**	**ES_ANA**	**ES_SIN**	**ES_AVA**
PLENA	PLENA	PLENA	PLENA	PARCIAL	PLENA

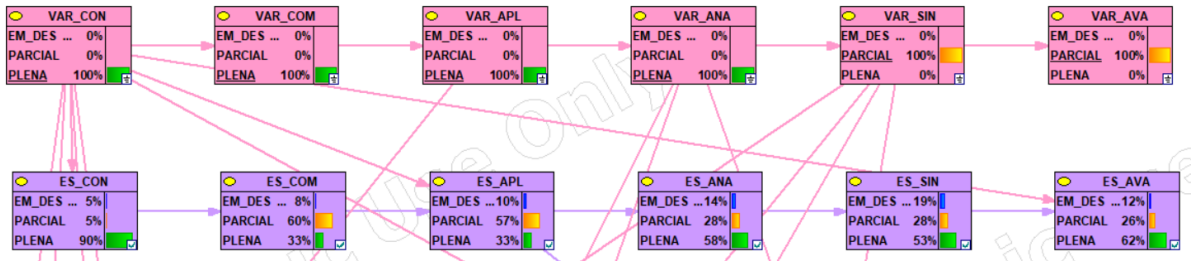
Fonte: A autora

Na Figura 11, pode-se observar que nos nós do conteúdo de variáveis (rosas) foram incluídas as evidências[27] referentes ao aprendizado do E1, que constam no Quadro 22. Já os nós correspondentes ao conteúdo entrada e saída de dados apresenta a probabilidade de o estudante aprender o conteúdo, dadas as evidências inseridas nos nós das variáveis. Ao compararmos essas probabilidades, pode-se observar pequenas divergências como: na linha 5 da coluna 2 do Quadro 22 mostra que o E1 atingiu o nível de aprendizado pleno, enquanto que na linha 2 da coluna 2 da Figura 11 a RB inferiu que o E1 atingiria o nível parcial, visto que apresenta a hipótese de 60%; na linha 5 coluna 3 do Quadro 22 o E1 apresenta nível de aprendizado pleno, já na linha 2 coluna 2 da Figura 11 a RB inferiu uma hipótese de

[27] A evidência inserida fica sublinhada e marca 100%.

57% para o aprendizado parcial; por último tem-se na linha 5 coluna 5 do Quadro 22 a evidência parcial, porém a RB calculou a hipótese de 53% para o aprendizado pleno dadas as hipóteses de aprendizado do conteúdo entrada e saída de dados no nível análise (ES_ANA).

Figura 11 - Parte da RB com as evidências e probabilidades do E1



Fonte: A autora

Neste sentido, é valido que o professor confirme essas hipóteses, visto que a entrada de dados utilizada foi pequena, podendo assim haver alguns cálculos que não sejam tão precisos. Para a estimativa de hipóteses mais precisas, é necessário continuar alimentando a RB.

No Quadro 23 é apresentado o nível de aprendizado do E1 utilizado como evidências para o cálculo das hipóteses de aprendizagem do conteúdo operadores lógicos. Consequentemente, foram incluídas 24 evidências correspondendo aos conteúdos: variáveis, entrada e saída de dados, operadores aritméticos e operadores relacionais. Na Figura 12 tem-se a representação da RB com as evidências inferidas (linhas 1 a 4) e as hipóteses calculadas na linha 5.

Quadro 23 - Evidências do E1

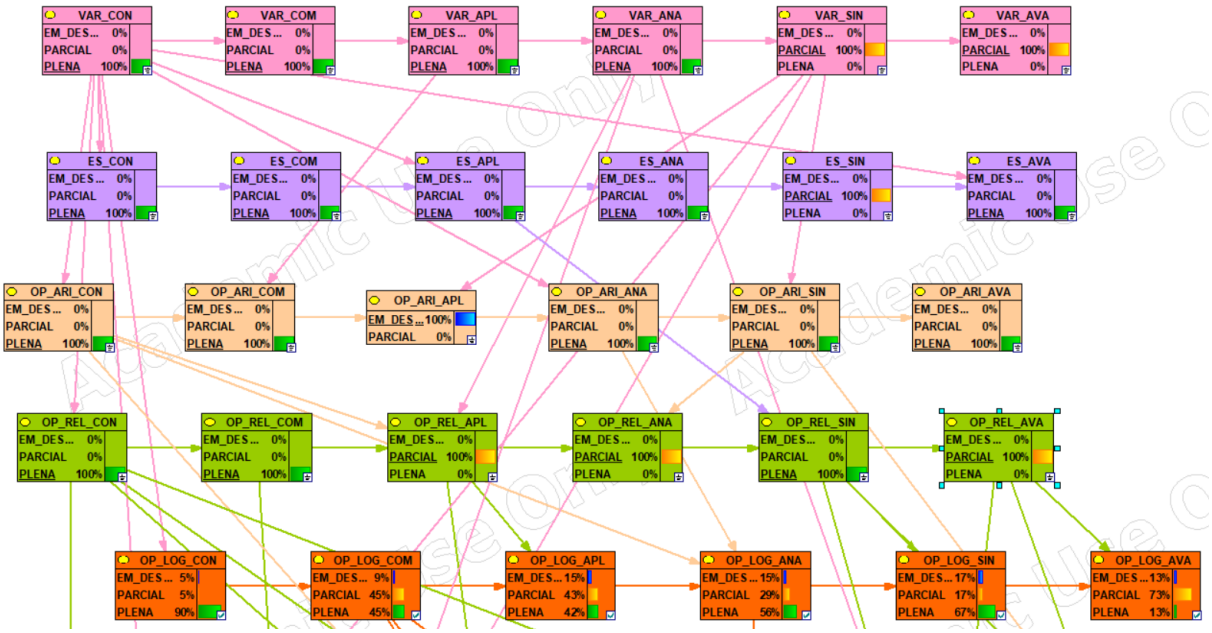
ESTUDANTE 1					
VAR_CON	**VAR_COM**	**VAR_APL**	**VAR_ANA**	**VAR_SIN**	**VAR_AVA**
PLENA	PLENA	PLENA	PLENA	PARCIAL	PARCIAL
ES_CON	**ES_COM**	**ES_APL**	**ES_ANA**	**ES_SIN**	**ES_AVA**
PLENA	PLENA	PLENA	PLENA	PARCIAL	PLENA
OP_ARI_CON	**OP_ARI_COM**	**OP_ARI_APL**	**OP_ARI_ANA**	**OP_ARI_SIN**	**OP_ARI_AVA**
PLENA	PLENA	EM DESENVOLVIMENTO	PLENA	PLENA	PLENA
OP_REL_CON	**OP_REL_COM**	**OP_REL_APL**	**OP_REL_ANA**	**OP_REL_SIN**	**OP_REL_AVA**
PLENA	PLENA	PARCIAL	PARCIAL	PLENA	PARCIAL
OP_LOG_CON	**OP_LOG_COM**	**OP_LOG_APL**	**OP_LOG_ANA**	**OP_LOG_SIN**	**OP_LOG_AVA**
PLENA	PLENA	PARCIAL	PLENA	PLENA	PARCIAL

Fonte: A autora

Pode-se dizer, ao analisar as hipóteses apresentadas para o conteúdo operadores lógicos, que neste experimento a RB estimou corretamente os níveis de

aprendizado do E1. Analisando os resultados tem-se: para o nível conhecimento o E1 atingiu o aprendizado pleno, como pode ser verificado no Quadro 23 (linha 11, coluna 1), o mesmo foi apontado na RB que estimou a hipótese de 90% para o aprendizado pleno neste nível (Figura 12, linha 5, coluna 1); no nível compreensão, no qual o E1 apresentou aprendizado pleno na resolução das atividades (Quadro 23, linha 11, coluna 2) e a RB calculou a hipótese de 45% para aprendizagem plena (Figura 12, linha 5, coluna 2); no nível aplicação a RB apresentou a hipótese de 43% para o aprendizado parcial (Figura 12, linha 5, coluna 3) que está de acordo com o que o E1 atingiu nas atividades (Quadro 23, linha 11, coluna 3); o mesmo pode ser observado nos demais níveis (análise, síntese e avaliação), no qual a RB estimou corretamente o aprendizado do conteúdo de operadores lógicos apresentados pelo E1 no Quadro 23.

Figura 12 – Parte 2 da RB com as evidências e probabilidades do E1



Fonte: A autora

Analisando de modo geral a RB representada na Figura 12 sobre o E1, pode-se dizer que o estudante não apresentou dificuldades de aprendizado. Porém, o E1 não conseguiu apresentar uma abstração refletida em suas atividades, visto que no nível Avaliação atingiu parcialmente o aprendizado dos conteúdos. O último nível da TOEDC abrange os aspectos de todos os outros níveis, sendo acrescido critérios que abrangem valores (BLOOM et al., 1972) e por isso é esperado uma abstração refletida,

para que se possa avaliar qual melhor prática de programação deve ser utilizada na resolução de um dado problema.

Novos experimentos foram feitos, agora com o Estudante 2 (E2) que apresenta as evidências do Quadro 24 para os CIP. Ao inserindo as 30 primeiras evidências, obteve-se a RB apresentada na Figura 13, que estimou as hipóteses de aprendizado para o conteúdo estrutura de repetição.

Quadro 24 - Evidências do E2

ESTUDANTE 2					
VAR_CON	**VAR_COM**	**VAR_APL**	**VAR_ANA**	**VAR_SIN**	**VAR_AVA**
PLENA	PARCIAL	PLENA	PARCIAL	PARCIAL	PLENA
ES_CON	**ES_COM**	**ES_APL**	**ES_ANA**	**ES_SIN**	**ES_AVA**
PLENA	PLENA	EM DESENVOLVIMENTO	PLENA	PARCIAL	PLENA
OP_ARI_CON	**OP_ARI_COM**	**OP_ARI_APL**	**OP_ARI_ANA**	**OP_ARI_SIN**	**OP_ARI_AVA**
PLENA	PLENA	EM DESENVOLVIMENTO	PLENA	EM DESENVOLVIMENTO	PLENA
OP_REL_CON	**OP_REL_COM**	**OP_REL_APL**	**OP_REL_ANA**	**OP_REL_SIN**	**OP_REL_AVA**
PLENA	PLENA	PARCIAL	PARCIAL	EM DESENVOLVIMENTO	EM DESENVOLVIMENTO
OP_LOG_CON	**OP_LOG_COM**	**OP_LOG_APL**	**OP_LOG_ANA**	**OP_LOG_SIN**	**OP_LOG_AVA**
PLENA	PLENA	PARCIAL	PARCIAL	EM DESENVOLVIMENTO	EM DESENVOLVIMENTO
EST_CON_CON	**EST_CON_COM**	**EST_CON_APL**	**EST_CON_ANA**	**EST_CON_SIN**	**EST_CON_AVA**
PLENA	PLENA	PARCIAL	PLENA	PARCIAL	EM DESENVOLVIMENTO
EST_REP_CON	**EST_REP_COM**	**EST_REP_APL**	**EST_REP_ANA**	**EST_REP_SIN**	**EST_REP_AVA**
PLENA	PARCIAL	PLENA	PLENA	PLENA	PARCIAL

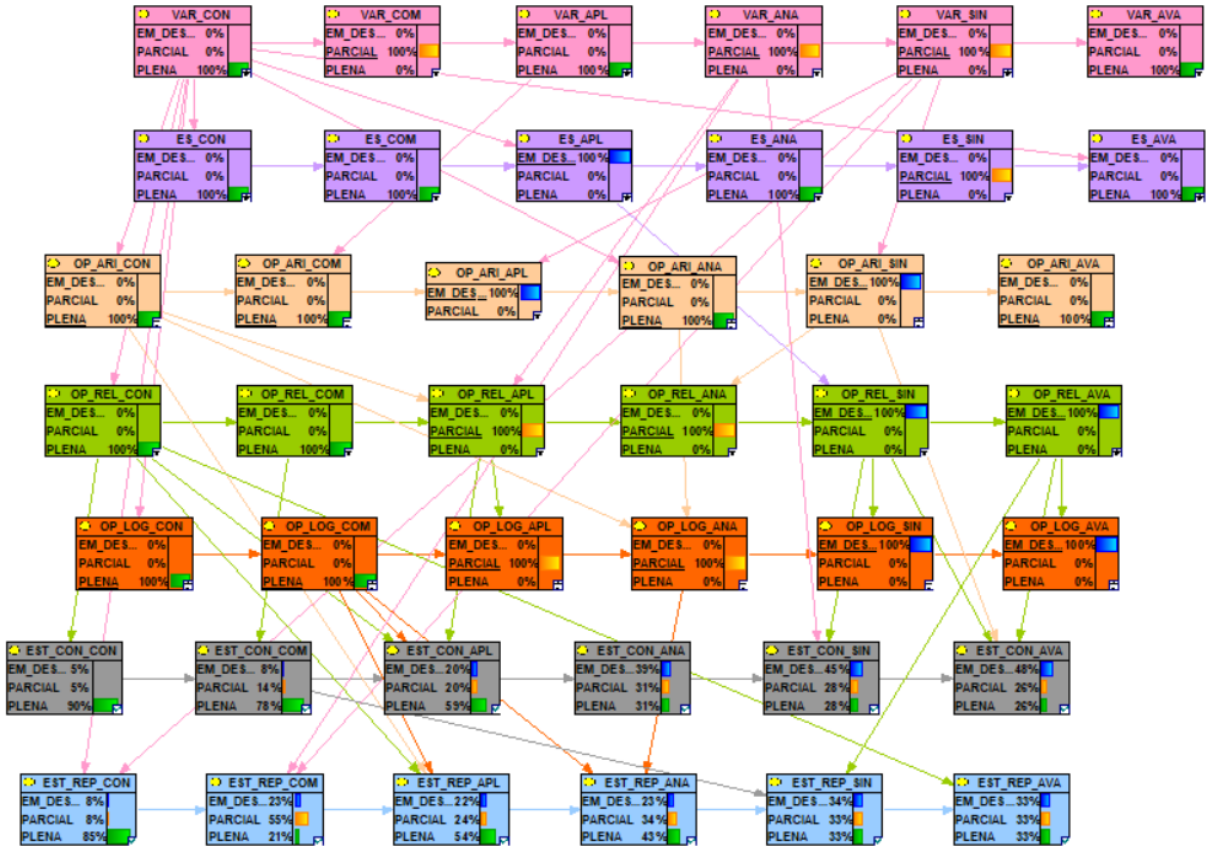
Fonte: A autora

Refletindo sobre a RB resultante do E2 (Figura 13), pode-se dizer que se trata de um estudante que possuía dificuldades na parte prática dos CIP, visto que apresentou na maioria dos conteúdos nos níveis Aplicação e Síntese aprendizado parcial ou em desenvolvimento. Verificando o resultado final deste estudante na DIP pode-se confirmar essa dificuldade, já que foi aprovado com conceito C.

No experimento realizado com os dados do E2, incluiu-se evidências dos conteúdos: variáveis, entrada e saída de dados, operadores aritméticos, operadores relacionais e operadores lógicos; resultando nas hipóteses dos conteúdos sobre estrutura de controle e estrutura de repetição. Essas hipóteses podem ser verificadas na Figura 13 linhas 6 e 7, e ao compará-las com o que o estudante apresentou sobre o aprendizado desses conteúdos durante o semestre verifica-se que a RB se

equivocou em quatro dos 12 nós calculados. Assim sendo, a RB mostrou-se 67% de acordo com o que de fato ocorreu com o aprendizado do E2.

Figura 13 – RB com evidências e hipóteses do E2



Fonte: A autora

Outro experimento realizado foi com os dados do Estudante 3, o qual apresenta as evidências do Quadro 25 sobre seu aprendizado dos CIP.

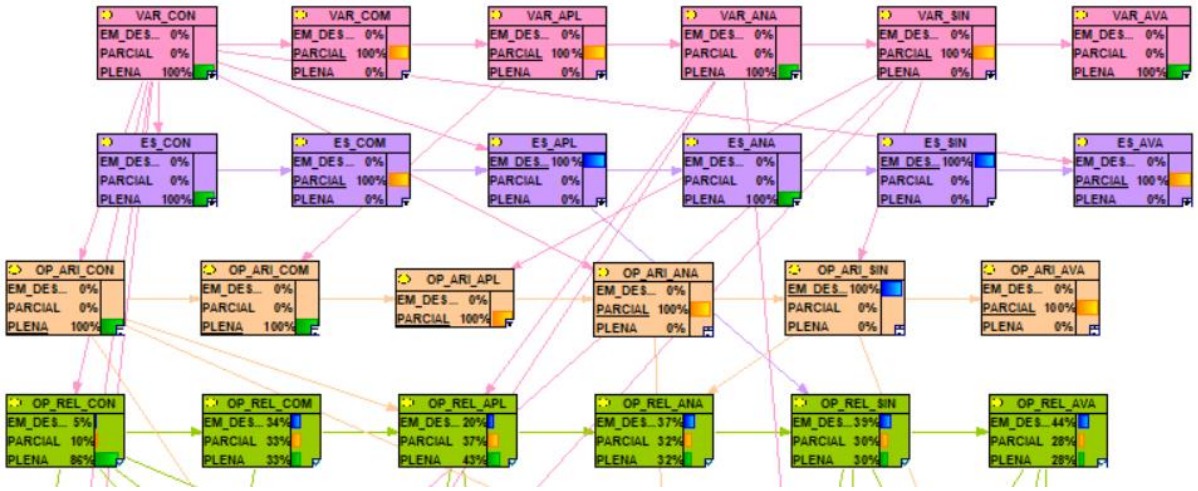
Quadro 25 - Evidências do E3

ESTUDANTE 3					
VAR_CON	**VAR_COM**	**VAR_APL**	**VAR_ANA**	**VAR_SIN**	**VAR_AVA**
PLENA	PARCIAL	PARCIAL	PLENA	PARCIAL	PLENA
ES_CON	**ES_COM**	**ES_APL**	**ES_ANA**	**ES_SIN**	**ES_AVA**
PLENA	PARCIAL	EM DESENVOLVIMENTO	PLENA	EM DESENVOLVIMENTO	PLENA
OP_ARI_CON	**OP_ARI_COM**	**OP_ARI_APL**	**OP_ARI_ANA**	**OP_ARI_SIN**	**OP_ARI_AVA**
PLENA	PLENA	PARCIAL	PARCIAL	EM DESENVOLVIMENTO	PARCIAL
OP_REL_CON	**OP_REL_COM**	**OP_REL_APL**	**OP_REL_ANA**	**OP_REL_SIN**	**OP_REL_AVA**
PLENA	PLENA	PLENA	EM DESENVOLVIMENTO	PLENA	PARCIAL

Fonte: A autora

As evidências de aprendizagem incluídos foram dos conteúdos: variáveis, entrada e saída de dados e operadores lógicos, fazendo com que a RB estime sobre o aprendizado de operadores relacionais. Na Figura 14 é apresentada a RB do E3, no qual é possível identificar que o desenvolvimento prático e teórico do estudante é satisfatório, porém o mesmo apresenta baixo rendimento no que se refere a análise e avaliação do código. Esses dois níveis (Análise e Avaliação) são semelhantes, visto que em ambos é esperado que o estudante utilize critérios para julgar a melhor prática de programação. Enquanto uma delas o estudante precisa reconhecer possíveis erros de código, sintaxe, lógica, etc.; a outra depende de um conhecimento mais avançado de julgamento, para que seja possível desenvolver uma boa prática de programação.

Figura 14 - RB parcial com evidências e hipóteses do E3



Fonte: A autora

Apesar da entrada de dados para a aprendizagem da RB ser restrita, a rede desenvolvida mostrou-se promissora, uma vez que o intuito da mesma é auxiliar o professor no acompanhamento do desenvolvimento de aprendizagem de seus estudantes. No próximo capítulo é apresentado como se faz a utilização da ConsProg.

7 DESCRIÇÃO DA PROPOSTA PEDAGÓGICA

Quando se pensa em ensino-aprendizagem de algum conteúdo ou conceito, o docente deve planejar[28] como isso ocorrerá. No presente capítulo será apresentado desde como o professor organiza e planeja suas aulas, até como acompanhar o desenvolvimento de aprendizagem dos estudantes. Logo, ter-se-á a explicação detalhada do uso da ConsProg, e para facilitar a compreensão da mesma, o capítulo está organizado em seções: Planejando a disciplina e as aulas; Desenvolvendo as aulas; Como criar, acompanhar e avaliar as atividades; Análise do desenvolvimento da aprendizagem com o uso da Rede Bayesiana.

7.1 PLANEJANDO A DISCIPLINA E AS AULAS

Todo curso, indiferente de sua modalidade e nível, tem seu PPC. Nele o professor encontra informações muito importantes para o planejamento[29] de sua disciplina como: objetivo do curso, perfil do profissional e do curso, organização curricular, ementas das disciplinas, além de outras informações sobre o curso e sobre a Instituição de Ensino. Neste sentido, é importante que o professor acesse esse material para que possa desenvolver o planejamento de sua disciplina de acordo com o que a Instituição e o Curso necessita.

No objetivo do curso, o professor encontrará o que a Instituição espera proporcionar na formação de seus estudantes. Logo, o planejamento da disciplina deve proporcionar que esses objetivos sejam atendidos em sua disciplina, de forma que o estudante consiga, ao longo do curso, desenvolver essas habilidades. Vale ressaltar, que apesar de existirem disciplinas semelhantes numa Instituição, cada uma delas tem objetivos distintos, pois normalmente encontram-se em cursos diferentes, assim, o professor deve planejar cada disciplina de acordo com o PPC do curso no qual ela pertence.

---

[28] “[...] antecipar mentalmente uma ação ou um conjunto de ações a ser realizadas e agir de acordo com o previsto.” (Vasconcellos, 2000, p. 79).

[29] “O planejamento enquanto construção-transformação de representações é uma mediação teórica metodológica para ação, que em função de tal mediação passa a ser consciente e intencional. Tem por finalidade procurar fazer algo vir à tona, fazer acontecer, concretizar, e para isto é necessário estabelecer as condições objetivas e subjetivas prevendo o desenvolvimento da ação no tempo. ” (Vasconcellos, 2000, p. 79).

Os objetivos do curso e o perfil do profissional acabam se complementando, uma vez que o perfil do profissional é o resultado dos objetivos atingidos dentro das disciplinas. Nesse caso, é apresentado no perfil do profissional o que se espera que o estudante consiga desenvolver após se formar, ou seja, ao se tornar profissional da área. Adjeto aos objetivos do curso e ao perfil do profissional, o PPC traz o perfil do curso, neste tópico o professor pode se apropriar da ideologia do curso e das disciplinas, conhecendo como o curso foi pensado e elaborado.

Através da organização curricular é possível visualizar e compreender como foi pensada a organização do curso (anual, semestral ou trimestral, dependendo do curso), bem como o que se espera ao final de cada uma das etapas. Logo na sequência é apresentada, normalmente, a matriz curricular. Nela o professor pode tomar conhecimento sobre a carga horária das disciplinas, o semestre em que elas se encontram e também os pré-requisitos, se existirem.

Após tomar conhecimento de toda estrutura do curso e de como ele funciona, o professor pode verificar as ementas das disciplinas, e está é uma das partes mais importantes para o planejamento das aulas. Na ementa de cada disciplina, o professor tem acesso aos objetivos da disciplina, os conteúdos que devem ser trabalhados, os critérios de avaliação, a bibliografia básica e a bibliografia complementar da disciplina. Assim, com essas informações, o professor poderá ter o conhecimento de quais assuntos deverão ser abordados em sua disciplina.

O planejamento normalmente é feito antes de iniciar a mesma a disciplina, e neste sentido, precisa pensar e idealizar o que vai ocorrer em cada aula, desde o conteúdo a ser abordado e como até a forma que se verificará o aprendizado do estudante. Porém o professor deve ter em mente que planejar não deve ser apenas um ato de cumprir protocolo e passar todo o conteúdo requisitado na ementa, deve garantir que o desenvolvimento cognitivo do estudante acompanhe esse planejamento. Logo, pode acontecer que o planejamento necessite adaptações ao longo da disciplina, e isso ocorrerá uma vez que o planejamento deve ser pensado para a turma. De acordo com Vasconcellos (2000, p.159), é conveniente tomar o cuidado para que não se caia em dois problemas: “de um lado, o planejamento se tornar o tirano da ação, ou de outro, se tornar um simples registro, um jogo de palavras desligado da prática efetiva do professor”.

O ideal é que o professor seja flexível, uma vez que adaptações são necessárias quando o mesmo deixa de ser conteudistas, preocupando-se muito mais

com a qualidade do ensino-aprendizado do que com a quantidade de informação que consegue passar ao estudante. Nesse pensamento tem-se Piaget (1998, p.15) afirmando “[...] que toda verdade a ser adquirida seja reinventada pelo aluno, ou pelo menos reconstruída e não simplesmente transmitida. [...]”, e ainda que “[...] o que se deseja é que o professor deixe de ser apenas um conferencista e que estimule a pesquisa e o esforço, ao invés de se contentar com a transmissão de soluções já prontas”. Sendo assim, muitas vezes pode-se ter que ir além daquilo que foi planejado, ou ainda ter que ir mais devagar para que os estudantes de fato se apropriem do conhecimento desejado.

O professor deve ter em mente também que em seu planejamento deve incluir as avaliações, ou seja, como ele vai verificar se o estudante atingiu os objetivos da disciplina. É importante ressaltar que uma avaliação não precisa ser, necessariamente, uma prova descritiva sem consulta. O professor tem a liberdade em escolher a forma que vai avaliar seu estudante, logo pode ser através de prova, de trabalho, ou até mesmo do acompanhamento de seu desenvolvimento durante a disciplina.

Em um modelo tradicional de ensino, a avaliação por meio do acompanhamento do desenvolvimento do estudante pode parecer muito subjetivo. Da mesma forma que avaliar o estudante através de uma prova de perguntas e respostas não garante que de fato ele tenha se apropriado do conhecimento. Sendo assim, cabe ao professor determinar qual a melhor forma de verificar se seu estudante atingiu ou não os objetivos da disciplina.

O planejamento geralmente é entregue na instituição antes de iniciar a disciplina, ou nas primeiras semanas de aula, e cada instituição tem seu modelo. Apesar disso, normalmente o planejamento é composto pelas datas das aulas, o conteúdo programático, os procedimentos didáticos e os instrumentos de avaliação. Sendo assim, em cada aula, o professor deve preencher com a data que ela irá ocorrer, o conteúdo programático (aquilo que ele pretende ensinar) e os procedimentos didáticos (como ele pretende ensinar). Na próxima seção é apresentada a forma no qual o docente pode desenvolver uma aula de acordo com a ConsProg.

7.2 DESENVOLVENDO A AULA

É um equívoco pensar que o estudante não sabe nada, ou mesmo que ele deva iniciar um curso tendo domínio da área que ele escolheu. Em primeiro lugar o professor deve conhecer sua turma e aula a aula fazer adequações em seu planejamento, a fim de que seus materiais sejam favoráveis para aquela turma. Nesse sentido, pegar o material que está pronto e replicar diversas vezes não seria uma boa prática pedagógica.

Outro ponto a ser considerado é com a qualidade do ensino-aprendizagem do conteúdo, e não com a quantidade de informações que estão sendo passadas. Ou seja, não ser conteudistas, se preocupando apenas em finalizar todos os conteúdos, sem levar em conta se os estudantes estão conseguindo aprender. Sendo assim, os materiais devem ser pensados e criados partindo do conhecimento a priori dos estudantes.

Esse conhecimento a priori podem ser exemplos de sua vida cotidiana, de forma a ser feito uma analogia entre o concreto (aquilo que ele viveu) e o abstrato (aquilo que ele precisa aprender). Logo, ao iniciar a criar o material que será utilizado na aula pode-se responder as seguintes perguntas:

1. O que vou ensinar?
2. O que os estudantes precisam saber para aprender esse conceito?
3. O que é mais importante que eles aprendam?
4. De que forma, na vida do estudante, ele já vivenciou algo semelhante ao que vai ser ensinado?
5. Que perguntas posso fazer ao estudante sobre o que ele já vivenciou a fim de conduzi-lo a pensar de forma semelhante ao que vai ser ensinado?

Todas essas perguntas devem ser respondidas para que o professor crie um ambiente propício para o ensino-aprendizagem, uma vez que o estudante estará se conectando ao conteúdo. Apesar de normalmente ser a primeira etapa de uma aula, o discente pode ter dificuldades no aprendizado, uma vez que programar se trata de informações abstratas e complexas. Neste momento pode ser que o discente tenha a necessidade de exemplos de seu cotidiano, ou dinâmicas que facilitem o entendimento do conteúdo que está sendo iniciado. De acordo com a teoria de Piaget (1995), pode-se dizer que este nível representaria a abstração empírica, no qual o

sujeito consegue solucionar um determinado problema através de suas experiências vividas.

Com isso, o docente pode iniciar fazendo relações entre o cotidiano dos estudantes e o conteúdo a ser ministrado. Logo, as conexões “tornam” o que é abstrato em algo concreto (através dos exemplos), facilitando o enriquecimento das propriedades coordenadas entre o concreto e o abstrato, desenvolvendo assim seu cognitivo. Quando o sujeito faz essa coordenação, significa que o objeto (conteúdo) está sendo modificado pelas ações do sujeito e este atinge um patamar superior, desenvolvendo estruturas para o próximo nível. Importante lembrar que “[...] o conhecimento procede a partir, não do sujeito, nem do objeto, mas da interação entre os dois [...]” (PIAGET, 1977, p. 198).

Assim sendo, para ensinar um novo conteúdo é aconselhável criar estratégias que façam o estudante pensar da mesma forma que pensaria se já conhecesse os conceitos a serem apresentados. Tratando-se de CIP, na aula em que o professor vai apresentar como é executado um programa, ele pode fazer a seguinte dinâmica: Pedir para a turma se organizar em duplas, entregar a elas uma folha em branco e pedir para que escrevam nela como uma pessoa que possui deficiência visual sairia da porta do laboratório em que se encontrar e consiga chegar até a biblioteca (escolher um lugar não muito distante do laboratório para que a dinâmica não fique muito extensa). Cada dupla pensará em uma estratégia, após a conclusão desta etapa, devem-se recolher as folhas e entregá-las a duplas diferentes. Pedir que cada dupla se organize da seguinte forma: um deles deve ser o “computador” que fará a leitura sequencial das informações escritas na folha recebida de outra dupla, e o outro executará elas de olhos vendados (ou fechados) a partir da porta do laboratório.

Durante a dinâmica eles podem perceber que faltam dados, ou que ao executar o que está escrito, eles não chegam ao destino, ou ainda que existe equívocos no que está escrito (manda ir para direita, mas deveria mandar ir para esquerda), etc.

Observe que para uma primeira aula, essa dinâmica também servirá para integração da turma, desinibição, descontração, e por meio deste o professor estará “preparando” as estruturas cognitivas dos estudantes para conhecerem o novo conteúdo. É como se o docente pudesse “escolher” o que o discente precisaria pensar quando estiver no processo de aprendizagem, nas palavras de Piaget (1975, p. 56) “um esquema resume em si o passado e consiste sempre, portanto, em uma

organização ativa da experiência vivida". Na subseção seguinte apresentar-se-á como o professor poderá seguir sua aula fazendo o uso da ConsProg.

Após a dinâmica ou exemplos do cotidiano, o docente não deve simplesmente fingir que nada aconteceu e partir para suas explicações sem relacionar com a preparação inicial, se não é como se não tivesse ocorrido. Assim, o professor pode pedir para que relatem o que aconteceu, e posterior a isso começar a apresentar o novo conteúdo e comparar este com os relatos dos estudantes. Neste sentido, ao iniciar os conceitos relacionados à estrutura básica da Linguagem C ANSI, explicando o que é uma linguagem de programação estruturada[30] e passos lógicos, o professor pode questionar se a ordem dos passos escritos na dinâmica, se colocados de forma diferente, funcionaria.

Vale ressaltar que deve ser ensinado aquilo que realmente é útil para o momento, evitando excesso de informação que poderá confundir os estudantes. Assim sendo, o ideal é intercalar conceitos com a prática e exemplos concretos, aumentando o nível de dificuldade gradativamente a medida em que os estudantes foram demonstrando compreensão por meio da interação sujeito x objeto. Para Piaget (1975, p. 19) "todo e qualquer ato de inteligência supõe um sistema de implicações mutuas e de significações solidárias", e a atividade do sujeito está em "uma interdependência irredutível entre a experiência e a razão" (idem, p. 26). Logo, os exemplos concretos e a prática servirão para dar significado aos conceitos, facilitando assim a compreensão do novo conteúdo.

Na sequência é apresentado como a ConsProg sugere criar, acompanhar e avaliar as atividades durante a disciplina.

7.3 COMO CRIAR, ACOMPANHAR E AVALIAR AS ATIVIDADES

Com o objetivo de apresentar uma parte da proposta pedagógica ConsProg, referente à criação, desenvolvimento, aplicação, acompanhamento e análise de atividades, esta seção ilustra como se cria um conjunto de atividades. Para estar de acordo com a TOEDC, essas atividades precisam aumentar gradativamente o nível de dificuldade. Ademais, será explicado como a análise destas atividades podem

[30] Segundo Deitel (2011), programação estruturada é a forma de programar baseada em três estruturas: sequencia, decisão e repetição, favorecendo a simplicidade do programa.

auxiliar na identificação das barreiras encontradas pelos estudantes. No apêndice n é possível verificar as questões utilizadas no levantamento de dados sobre o desenvolvimento da aprendizagem dos estudantes na disciplina LPI.

Para cada conceito ou conteúdo é necessário a criação de um conjunto de atividades que contemplem os seis níveis da TOEDC (conhecimento, compreensão, aplicação, análise, síntese e avaliação). Com o intuito de sistematizar o processo de coleta das respostas dos estudantes, e assim agilizar a análise das mesmas, adotou-se a utilização dos questionários para sistematizar a coleta das resoluções das atividades. Durante as aplicações realizadas na pesquisa, a ferramenta de criação de questionário do AVA Moodle mostrou-se o mais adequado para este fim, uma vez que permite a análise dos dados de forma individual, ou coletiva, por meio de gráficos, facilitando assim o acompanhamento do processo de ensino-aprendizagem.

Antes de utilizar os níveis da TOEDC, que norteiam a elaboração dos exercícios e a análise dos mesmos, o docente precisa levar em conta algumas questões: “[...] a porção de conhecimentos que se deve exigir do aluno [...] o nível de precisão a ser exigido [...] organização do conhecimento para facilitar a aprendizagem” (Bloom et al., 1972, p. 32 – 33). Posteriormente, inicia-se o passo da construção dos exercícios para cada conteúdo, tendo seu nível de dificuldade organizado de acordo com a TOEDC. Estes tem o propósito de identificar possíveis barreiras no aprendizado, e devem ser criados de maneira que abranjam o máximo dos conceitos que se fazem importante para o processo de ensino-aprendizagem do conteúdo como um todo. Porém, deve ser evitado questionamentos sobre informações que não serão utilizadas na sequência da disciplina.

Dessa maneira ter-se-á dados relevantes para a análise do processo de ensino-aprendizagem, que após coletados devem ser observados pelos docentes a fim de identificar as prováveis dificuldades apresentadas pelos estudantes. Caso essa análise aponte para déficits na aprendizagem, é indicado que o professor retome o conteúdo, uma vez que eles se interligam e servem de base para outros.

As primeiras atividades são relacionadas ao **nível conhecimento**, que de acordo com Bloom et al. (1972, p. 171) “envolve a evocação de específicos e universas, de métodos e processos, ou de um padrão, estrutura ou composição”. Espera-se, portanto, que o estudante encontre no problema elementos que o leve a indícios apropriados para “acessar” as informações daquilo que foi adquirido e armazenado sobre o conteúdo (Bloom et al., 1972). Em outras

palavras, o discente precisa mostrar que consegue: apresentar, reconhecer, ordenar, relacionar, listar, etc., informações do que foi aprendido. Um exemplo prático seria uma atividade de completar lacunas conforme apresentado no Quadro 26.

Quadro 26 - Exemplos de atividades nível conhecimento

“A instrução `if` é utilizada na tomada de decisões. ”
“A variável de controle é utilizada em laços de repetição”.

Fonte: A autora

Além disso, este nível também é responsável por estruturar o conhecimento de forma que os estudantes consigam, posteriormente, compreender para que servem, construir trecho de códigos relacionados ao conteúdo aprendido, analisar um programa pronto e reconhecer possível erros, desenvolver a síntese dos conteúdos, e ainda avaliar a melhor prática de programação.

Na sequência, conta-se com o **nível compreensão** que é considerada como sendo a mais inferior em questão de entendimento. Na TOEDC “compreensão refere-se àqueles objetivos, comportamentos ou respostas que representam um entendimento da mensagem literal contida em uma comunicação” (Bloom et al., 1972, p. 77). Com esse nível é importante que estudante entenda o que está sendo pedido, coordenando suas ideias para que possa ir além do que o exercício apresenta. Logo, espera-se que ele possa compreender, interpretar, alterar, selecionar, explicar, etc. sobre o que lhe foi ensinado. Em programação seriam questões relacionados a problemas no qual o discente precisa interpretá-lo para responder, como o exemplo apresentado no Quadro 27.

Quadro 27 – Exemplos de atividades nível compreensão

“Um programa precisa mostrar a seu usuário o Índice de Massa Corpórea que foi calculado na variável ‘imc’, que armazena números reais. Logo, faz-se o uso da instrução `printf`, diga qual o código de formatação é utilizado para apresentar a variável ‘imc’ ao usuário: ”
“De acordo com as regras de precedência, responda verdadeiro ou falso: Os operadores aritméticos ‘`+`’, ‘`*`’ e ‘`/`’ possuem o mesmo nível de precedência.”.

Fonte: A autora

Através das respostas dos estudantes neste nível é possível prever se os mesmos conseguirão aplicar os conceitos aprendidos. Sendo assim, caso os estudantes apresentem déficit na resolução dos exercícios do nível compreensão, possivelmente terão dificuldades no nível seguinte.

No terceiro nível, a **aplicação**, requer “o uso de abstrações em situações particulares e concretas” (Bloom et al., 1972, p. 176). A partir desta camada, o nível de abstração exigido do estudante é bastante elevado, uma vez “deverá aplicar as abstrações apropriadas sem que lhe tenha sido sugerido quais são estas abstrações ou sem que lhe seja ensinado com usá-las naquela situação” (Bloom et al., 1972, p.103). Tratando-se de programação, é solicitado que o discente resolva um determinado problema por meio de códigos de programação que necessita de uma sequência lógica para que funcione. Para aqueles que tiveram dificuldades na camada anterior, a resolução dessas atividades pode se tornar confuso em virtude do nível de abstração que é requerido para tal. Porém nesse nível a aplicação se dá em “partes”, ou seja, não se “misturam” conceitos. Um exemplo de atividade para essa camada seria pedir ao estudante que desenvolva um trecho de código relacionado ao que está sendo estudado (Quadro 28).

Quadro 28 - Exemplos de atividades nível aplicação

“A área de um triângulo equilátero é calculado através da fórmula: (base*altura)/2, o cálculo dessa fórmula é armazenado na variável ‘area’. Escreva um trecho de código para declarar as variáveis reais ‘base’, ‘altura’ e ‘area’. ”
“O programa ‘Mostrando números’ mostra na tela os números de 0 (zero) a 100 (cem) um abaixo do outro. Desenvolva o trecho de código que apresente a estrutura `for` para a solução deste problema. ”

Fonte: A autora

Verificando a resolução das atividades deste nível, o professor deve observar se seus estudantes conseguem estruturar de forma correta os códigos, e se os mesmos compreendem o que está sendo pedido. Após a construção de trechos de códigos, é importante que o discente consiga identificar nos códigos ou trecho de códigos possíveis erros, objetivos este do **nível análise**. De acordo com Bloom et al. (1972), neste nível se tem a percepção sobre a organização do material, suas interligações, partes constitutivas e técnicas, a fim de alcançar uma maior compreensão sobre o conteúdo. Ao analisar as relações entre os elementos de um código, por exemplo, necessita-se julgar se o mesmo está coerente. Vale salientar que esse julgamento não deve ser a expressão de opiniões, pois trata-se de técnicas e regras a serem seguidas. Consequentemente, ao desenvolver exercícios para este nível, o professor pode apresentar um código e pedir para que o estudante identifique possíveis erros, ou expor um problema com opções de soluções e solicitar que seja

marcada aquela que soluciona a atividade e que não apresenta erros, ou ainda pedir para que responda sobre a execução de um código. Exemplo possível: “Qual o valor da variável acumulador ao final da execução do código apresentado no Código 7:

Código 7 - Exemplo de atividade do nível analise

```
#include <stdio.h>
int main(){
     int acumulador = 0;
     for(int x=0; x<10; x++)
          acumulador = acumulador + x;
     printf(“%d”, acumulador);
     return 0;
}
```

Fonte: A Autora

No **nível síntese**, o quinto na TOEDC, passa-se a incorporar todos os conteúdos estudados, no caso da programação, deixa de ser avaliado apenas trechos de código e passa para a solução completa dos problemas. Bloom et al. (1972, p. 137) explicam que síntese “é um processo de trabalhar com elementos, partes, etc. e combiná-los para que constituam uma configuração ou estrutura não claramente percebida antes”. Sendo assim, é esperado que todo discente consiga chegar pelo menos até este nível hierárquico, e assim estar apto em relação aos CIP. Neste nível o estudante deve recombinar conceitos já conhecidos com os novos, obtendo assim um todo novo e bem integrado (Bloom et al., 1972).

Um exemplo prático para este nível seria pedir para que o estudante resolva um determinado problema, de forma que envolva outros conceitos já estudados. Supondo que o conteúdo da aula “x” é estrutura de controle `if`, verifique o exemplo no Quadro 29.

Quadro 29 - Exemplo de atividade nível síntese

“Um mini mercado gostaria de controlar a idade media de seus fregueses e também conhecer qual faixa etária frequenta mais o mesmo: Adolescente (de 16 a 24), Adulto (25 a 59), Senil (acima de 60). ”

Fonte: A autora

É esperado que o estudante apresente toda a estrutura básica do programa, juntamente com operadores relacionais e matemáticos, e ainda o novo conceito referente a estrutura de controle. Assim sendo, se o aprendizado dos conteúdos anteriores foi efetivo, o estudante poderá apresentar dificuldades na construção do `if`, caso contrário o mesmo não conseguirá desenvolver o programa.

O nível mais elevado é a **avaliação** requerendo um nível de abstração bastante elevada no qual é denominada por Piaget (1995) de abstração refletida, pois o sujeito se apropriou do que foi aprendido. Quando adquire um novo conhecimento, necessita-se de exemplos para tornar a compreensão mais fácil, quando o estudante se apropriou do que foi ensinado, isso se torna parte dele, não sendo mais necessário dos exemplos ou de consultas para a utilização. Pode-se dizer que é como se fosse algo “decorado”, porém consciente do que é, para que serve e como utilizar de diversas formas e situações não vivenciadas.

Bloom et al. (1972, p.157) definem “avaliação como o processo de julgamento acerca do valor de ideias, [...] soluções, métodos, [...] realizados com um determinado propósito”, podendo ser em alguns casos “[...] o preludio da aquisição de um novo conhecimento, de um novo esforço de compreensão, de aplicação, ou de uma nova síntese”. No caso dos CIP, um conteúdo normalmente será a base para o novo, e por isso se torna tão importante o acompanhamento do aprendizado dos estudantes. Atividades para esse nível podem estar relacionada à capacidade de apontar códigos que apresentem resoluções mais adequadas para um determinado problema como: “Marque a alternativa no Código 8que apresenta a melhor solução para o seguinte problema: Faça um programa que peça ao usuário 4 valores inteiros e retorne a média deles: ”

Código 8 - Alternativas da atividade exemplo do nível avaliação

a)
```
#include <stdio.h>
int main(){
  int val1, val2, val3, val4;
  int soma;
  float media;
  prinf(“Digite um valor:”);
  scanf(“%d”, &val1);
  prinf(“Digite um valor:”);
  scanf(“%d”, &val2);
  prinf(“Digite um valor:”);
  scanf(“%d”, &val3);
  prinf(“Digite um valor:”);
  scanf(“%d”, &val4);
  soma = val1+val2+val3+val4;
  media = soma/4;
  printf(“Media: %f”, media);
}
```

b)
```
#include <stdio.h>
int main(){
  int val, i;
  int soma;
  float media;
  for(i=0;i<4;i++){
    prinf(“Digite um valor:”);
    scanf(“%d”, &val1);
    soma += val1;
  }
  media = soma/4;
  printf(“Media: %f”, media);
}
```

Fonte: A autora

Através desses seis níveis espera-se que o estudante desenvolva a capacidade intelectual de programar e não meramente o conhecimento e informação de como

programar. Segundo Bloom et al. (1972, p. 34 - 35) a capacidade intelectual representa “combinações de conhecimento e habilidades” no qual se espera que o estudante “organize ou reorganize um problema, reconheça o material necessário, evoque este material e o utilize na situação problemática”. Sendo assim, caso haja alguma dificuldade durante o processo de aprendizagem, o docente poderá identificar em que momento o estudante apresentou tal situação e se for necessário retomar o conteúdo a fim de suprir qualquer carência. Na seção seguinte apresenta-se a forma que a ConsProg propõe de analisar o desenvolvimento da aprendizagem.

7.4 ANÁLISE DO DESENVOLVIMENTO DA APRENDIZAGEM

Uma vez criada as atividades e disponibilizada aos estudantes, cabe ao professor acompanhar as respostas e participação deles, a fim de identificar as possíveis dificuldades apresentadas. Através dessas informações o professor poderá retomar os conceitos necessários para o melhor desenvolvimento da aprendizagem dos discentes. Neste sentido, é necessário que o docente crie um conjunto de atividades para cada aula, seguindo as orientações descritas na seção 7.3 - Como criar, acompanhar e avaliar as atividades.

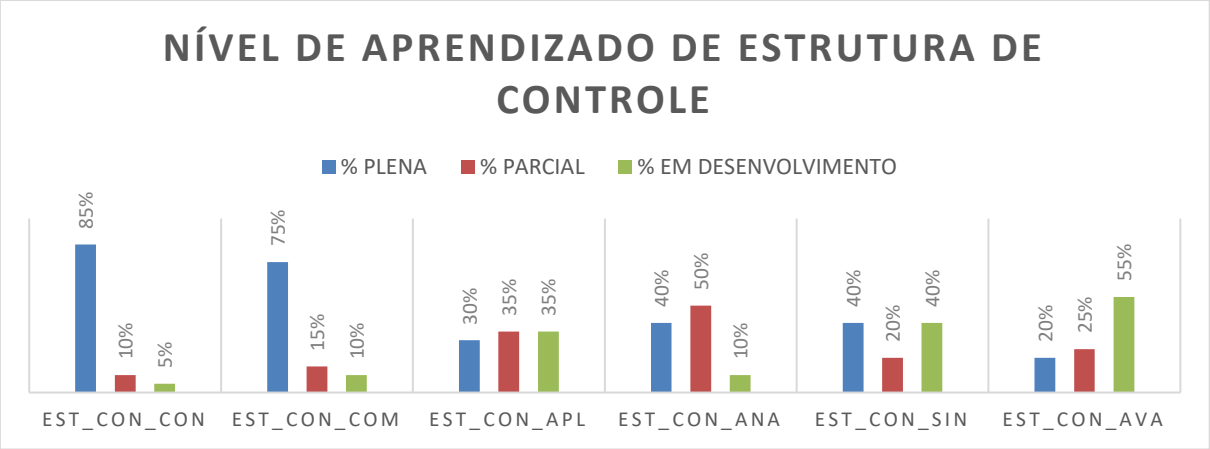
No final da correção das atividades, o professor terá a porcentagem de acertos de cada nível, podendo incluir essas informações na RB, conforme apresentado na subseção 6.3.2 - Experimentos realizados. É esperado que os estudantes atinjam o aprendizado pleno em cada um dos níveis da TOEDC para cada conteúdo, pois irá demonstrar que o mesmo está conseguindo desenvolver a aprendizagem de programação. Logo, o docente pode verificar o desenvolvimento da aprendizagem de duas formas: uma analisando a turma como um todo, e a outra analisando os estudantes individualmente.

No Gráfico 20 é possível observar um exemplo do levantamento de aprendizado dos estudantes no conteúdo estrutura de controle. Em primeiro lugar o professor deve verificar os níveis da TOEDC que a turma apresenta um índice de aprendizado “em desenvolvimento” maior. No exemplo apresentado (Gráfico 20) o nível avaliação é o único que se enquadra nessa análise.

O nível de avaliação requer que o estudante consiga avaliar os códigos, a fim de encontrar ou desenvolver aquele que está mais de acordo com as boas práticas

de programação. Neste caso, não se trata e um nível que possa comprometer o aprendizado dos demais conceitos de programação, logo, o professor não precisa se preocupar em retomar o conteúdo, já que esse nível vai sendo aprimorada à medida em que se programa.

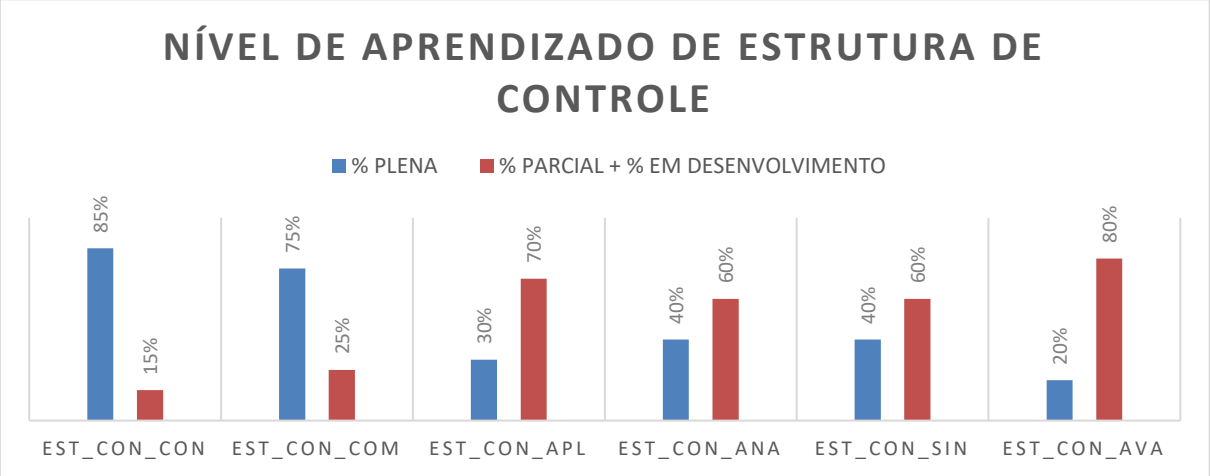
Gráfico 20 – Desenvolvimento de aprendizado em estrutura de controle



Fonte: A autora

Uma segunda análise seria somar a porcentagem de aprendizado “parcial” com a “em desenvolvimento” para que se possa verificar se há índices preocupantes. No Gráfico 21 é apresentado o nível de aprendizado de estrutura de controle, ao observá-lo é possível identificar que os níveis aplicação (EST_CON_APL), análise (EST_CON_ANA) e síntese (EST_CON_SIN) apresentam uma taxa superior no requisito dificuldade (parcial + em desenvolvimento). Esses três níveis devem ser revistos pelo professor, pois são bases fundamentais para o aprendizado dos demais conteúdos.

Gráfico 21 – Desenvolvimento de aprendizado - estrutura de controle (análise 2)



Fonte: A autora

O nível de aplicação exige que o estudante consiga colocar em prática o que foi aprendido na aula em questão (estrutura de controle), a análise mostra se o estudante consegue identificar erros no programa, e na síntese espera-se que o estudante consiga desenvolver um programa completo, fazendo o uso de todos os conhecimentos adquiridos até então. Logo, seria indicado que o docente retomasse o conteúdo de estruturas de controle, já que boa parte da turma não apresentou ter assimilado o mesmo. O professor pode investigar mais a fundo, analisando em cada um dos níveis as atividades correspondentes, e assim retomar o que de fato os estudantes necessitam.

Os gráficos apresentados anteriormente não levaram em conta o aprendizado dos demais conteúdos, visto que não foi calculado pela RB. Assim sendo, caso o professor queira identificar as possíveis causas para as dificuldades apresentadas, é importante que os dados coletados da turma sejam inseridos na RB, e assim ter-se-á as possíveis causas das dificuldades. Suponha que a turma x possua as evidências apresentadas no Quadro 30 para o conteúdo entrada e saída de dados, ao inserir essas evidências na RB ela calcula hipóteses de aprendizagem do conteúdo anterior e do posterior, conforme pode ser observado na Figura 15.

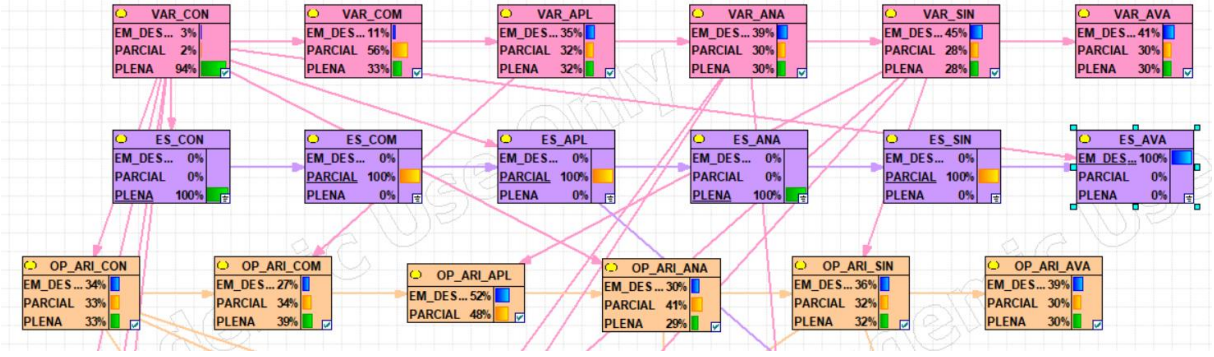
Quadro 30 - Evidências fictícias de uma turma X

EVIDÊNCIAS DO CONTEÚDO ENTRADA E SAÍDA DE DADOS					
ES_CON	**ES_COM**	**ES_APL**	**ES_ANA**	**ES_SIN**	**ES_AVA**
PLENA	PARCIAL	PARCIAL	PLENA	PARCIAL	EM DESENVOLVIMENTO

Fonte: A autora

Vale ressaltar que a RB apresentada nesse trabalho não foi implementada, por isso as evidências estão sendo incluídas de forma manual, caso contrário, essas informações são geradas de forma automática para que o professor faça a consulta.

Figura 15 - RB das evidências fictícias da turma x



Fonte: A autora

Dadas as evidências do Quadro 30, a RB (Figura 15) estimou que o possível motivo para dificuldade no nível avaliação no conteúdo de entrada e saída de dados, foram as dificuldades apresentadas no conteúdo variáveis nos níveis aplicação, análise, sínteze e avaliação. Caso o professor não retome o conteúdo de variáveis, as dificuldades acumularão, resultando na dificuldade nos níveis aplicação, sínteze e avaliação do conteúdo operadores aritméticos. Assim sendo, a RB proposta sistematiza o acompanhamento do desenvolvimento de aprendizagem dos estudantes, facilitando assim o levantamento das hipóteses de dificuldades para que o docente possa pensar em estratégias pedagógicas para auxiliar os estudantes no aprendizado. No próximo capítulo são apresentadas as considerações finais do presente trabalho.

8 CONSIDERAÇÕES FINAIS

As DIP são consideradas por muitos estudantes uma das disciplinas mais difíceis nos cursos de TI, segundo Lathinen (2005) e Sevella et al. (2013) isso é devido ao alto nível de abstração que é exigido para se aprender a programar. Neste sentido, estudos sobre o ensino-aprendizagem de programação vem crescendo nos últimos anos, tendo como principais enfoques a compreensão das dificuldades encontradas pelos estudantes, e a proposição de estratégias para minimizar essas barreiras. Neste sentido, através de levantamentos bibliográficos, pesquisas documentais e estudo de caso, elaborou-se uma proposta pedagógica a fim de auxiliar o processo educativo, visto que a maioria dos trabalhos pesquisados tratam da construção ou uso de ferramentas sem embasamento de teorias educacionais.

Assim sendo, a presente pesquisa desenvolveu uma proposta pedagógica baseada na TC, na TOEDC e a partir de observações realizadas na disciplina LPI durante o ano de 2016. Estas observações possibilitaram na compreenção das dificuldades dos estudantes, bem como as necessidades de estratégias pedagógicas que resultaram na proposta pedagógica ConsProg.

Essa proposta teve o intuito de facilitar a elaboração de atividades que possibilitam inferências sobre o processo de aprendizagem dos estudantes. Estas inferências tem o intuito de auxiliar o docente a identificar as possíveis dificuldades encontradas pelos estudantes, podendo assim realizar intervenções individuais ou coletivas, de acordo com as necessidades apresentadas. Durante a pesquisa foi possível testar a ConsProg em dois semestres letivos, com turmas de LPI do curso STSI do IFRS Campus Porto Alegre.

Nesses dois momentos identificou-se que a construção de uma proposta pedagógica por si só não é suficiente, já que será utilizada por um terceiro, e este pode ter um entendimento diferente daquilo que se idealizou. Logo, acredita-se que haja a necessidade de cursos de extensão, oficinas, acompanhamentos das aulas, para que não ocorra interpretações erronias do que a ConsProg propõe. Além disso, ao observar e analisar alguém aplicando a proposta pedagógica, permitiu a reflexão sobre a mesma, resultando em aprimoramentos que não seriam perceptíveis caso fosse aplicado por quem a desenvolveu. Logo, foi bastante válida a experiência de criar e observar a proposta sendo utilizada por alguém que não tem formação

pedagógica, pois permitiu o crescimento profissional e a melhoria da prática docente de ambos (professor da disciplina e pesquisador).

A ConsProg se propõe a auxiliar o professor desde o desenvolvimento de materiais para as aulas até o acompanhamento do desenvolvimento de aprendizagem dos estudantes. Para que seja possível esse acompanhamento, o docente deve criar atividades de acordo com a ConsProg. As atividades podem ser disponibilizadas tanto em papel quanto em meio digital, essa segunda forma foi a escolhida na pesquisa.

Logo, para a criação dos questionários utilizados na pesquisa foram utilizadas duas ferramentas TICs: Google Forms e questionários do AVA Moodle. As duas ferramentas permitem a criação de questionários com apresentação da “nota” de acordo com os erros e acertos, bem como feedback final ou por questão. Além disso, ambas apresentam um gráfico sobre a turma para o professor, porém no Google Forms os gráficos são mais fáceis de compreender e há a opção de visualização individual, ou seja, por estudante.

Tanto o Google Forms quanto os questionários do AVA Moodle permitem o download das informações em formato de planilha eletrônica, contudo, as informações exportadas pelo AVA Moodle não são tão completas quanto do Google. Após a utilização de ambos, para as Instituições que já utilizam o AVA Moodle, recomenda-se o uso do mesmo para a criação dos questionários, visto que o controle de acesso dos estudantes fica mais fácil, não sendo necessário a utilização de mais uma ferramenta. Ademais, a reutilização das questões, bem como a manipulação de um banco de questões é mais fácil no AVA Moodle.

Uma vez que o AVA Moodle foi desenvolvido para um público geral, não há suporte a códigos de programação, o que acaba dificultado quando se trata de questões do nível de aplicação e síntese. Nesses dois níveis é necessário que o estudante coloque em prática aquilo que aprendeu, não sendo útil apenas questões de marcar. Logo, opta-se pelo modelo de respostas curtas, entretanto, se faz necessário a correção manual pelo professor, pois cada estudante tem uma forma de programar, sendo inviável a tentativa de “controlar” as respostas com algumas sugestões do docente.

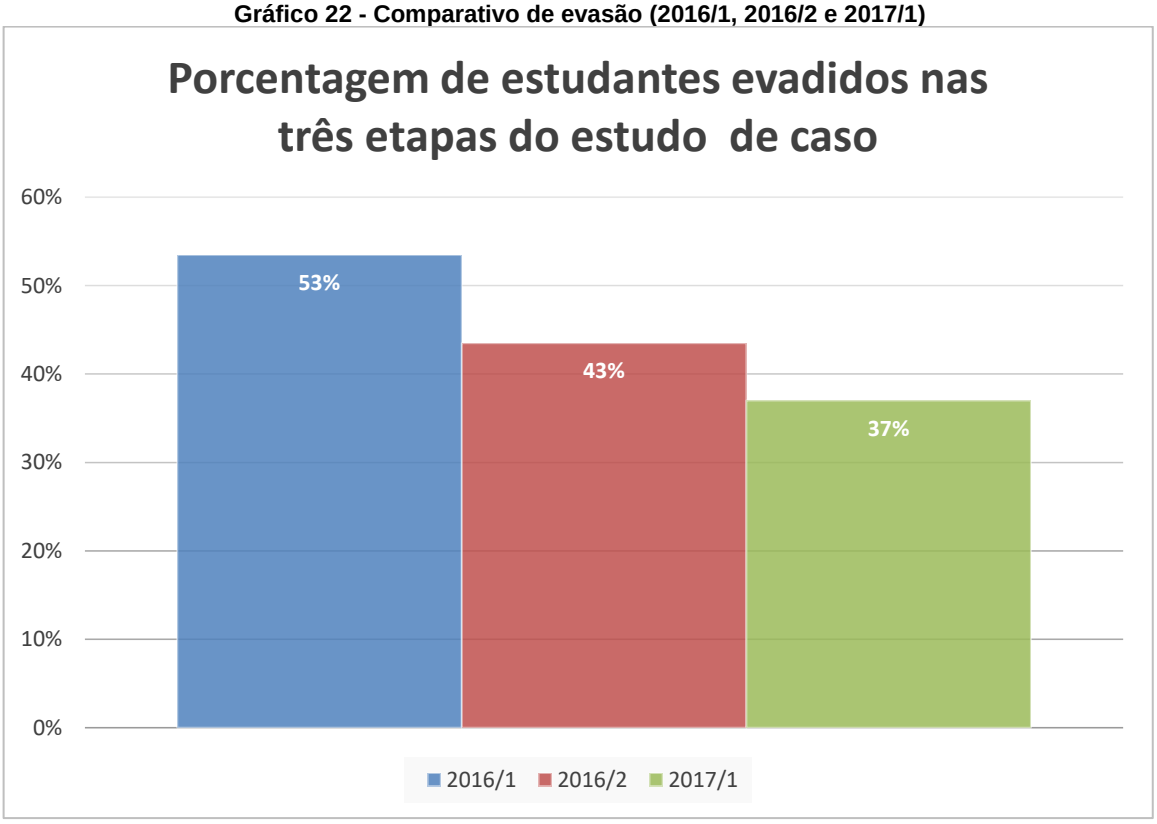
Assim sendo, acredita-se que a construção de uma ferramenta que atenda essas necessidades faria com que o acompanhamento do estudante ocorresse de forma mais rápida. Esse sistema poderia ser utilizado durante a aula e/ou posterior a ela, compreendendo atividades desenvolvidas de acordo com a ConsProg, para que

o professor possa receber notificações e alertas sobre o desenvolvimento de aprendizagem de seus estudantes. Já para os discentes, o sistema poderia contar com materiais de apoio, indicados pelo sistema de acordo com as dificuldades por eles apresentadas. Porém, vale ressaltar que a ConsProg não está diretamente ligada a uma ferramenta digital, podendo o professor fazer o uso dela até mesmo com atividades no papel, para correção e acompanhamento manual do estudante.

Através do uso da ConsProg foi possível identificar que o cronograma da disciplina LPI do curso STSI do IFRS Campus Porto Alegre era muito apertado para que houvesse um aprendizado mais efetivo. Com isso, optou-se em reestruturá-lo, retirando os conteúdos registros e ponteiros, o que segundo os estudantes ficou melhor, uma vez que tiveram mais tempo para se dedicar aos CIP.

Traçando um comparativo entre os três semestres analisados, pode-se dizer que o uso da ConsProg se mostrou promissora, visto que os índices de evasão reduziram de 53% em 2016/1 para 37% em 2017/1 conforme pode ser observado no Gráfico 22. Vale ressaltar que é considerado evadido da disciplina aquele estudante que tem frequência menor que 75%.

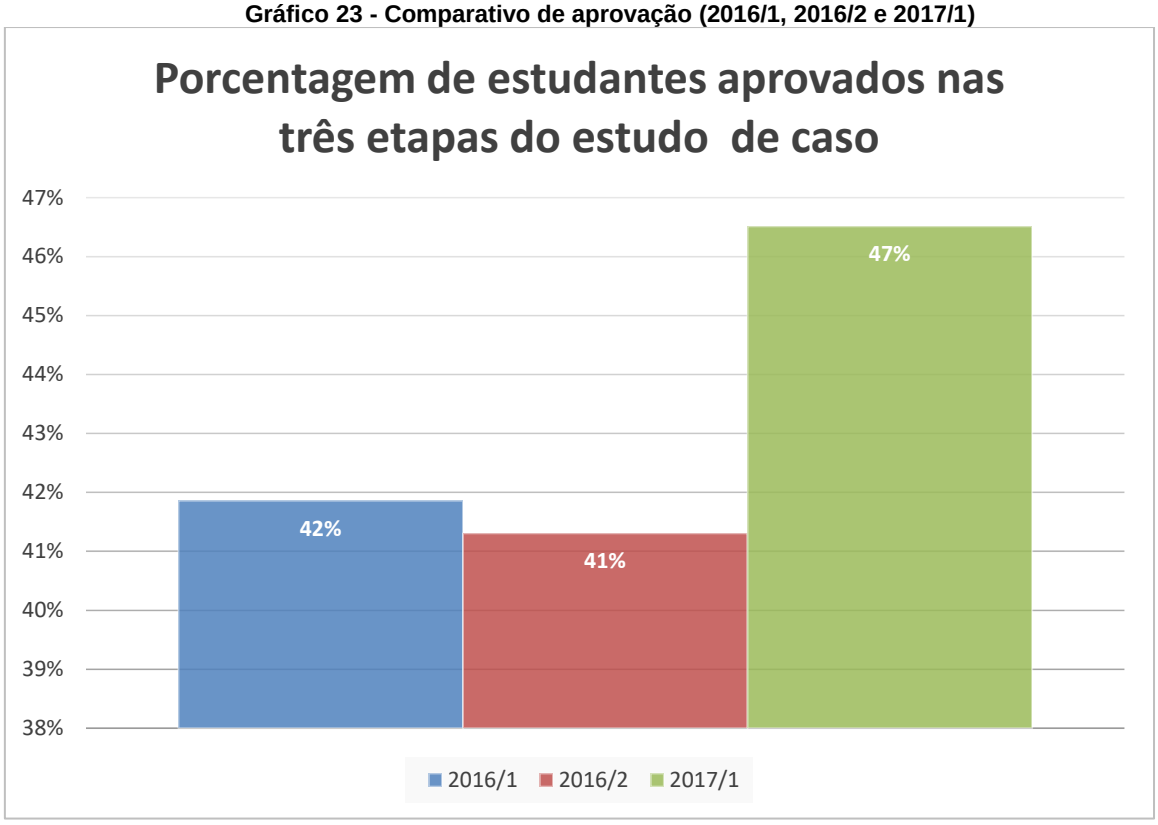
Gráfico 22 - Comparativo de evasão (2016/1, 2016/2 e 2017/1)



Fonte: A autora

Além disso, o índice de aprovação subiu 5% em 2017/1 comparando com 2016/1 em que não havia a proposta pedagógica sendo utilizada (Gráfico 23). De acordo com os estudantes do semestre 2017/1, que reprovaram em 2016/1, as aulas foram mais fáceis de acompanhar e os exemplos que o professor utilizava ficava mais fácil de compreender o que tinha que ser feito. Porém, esses estudantes relataram que ainda possuem dificuldades para saber o que precisa ser feito no programa. Nota-se ainda no Gráfico 23 que houve uma baixa de 1% no índice de aprovação comparando 2016/1 com 2016/2, acredita-se que essa diferença é irrelevante, visto que a taxa de estudantes evadidos caiu em 10% (Gráfico 22).

Gráfico 23 - Comparativo de aprovação (2016/1, 2016/2 e 2017/1)

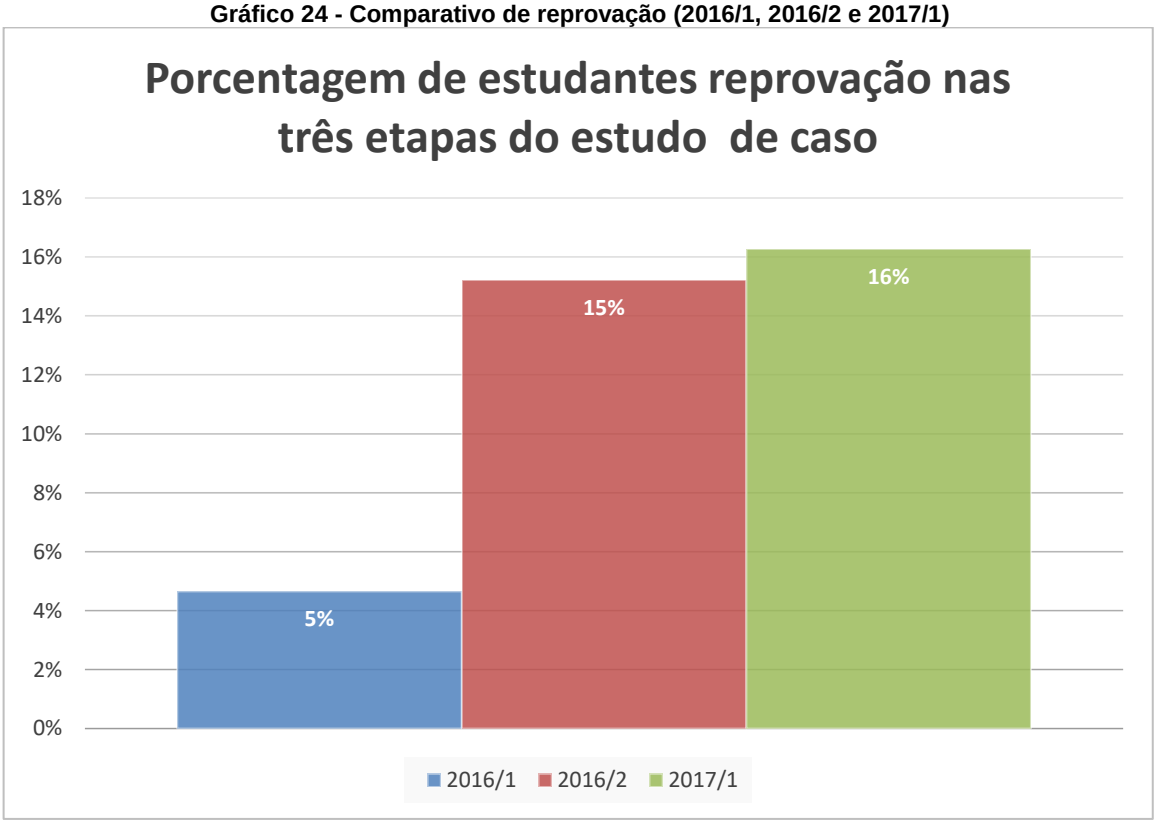


Fonte: A autora

Neste sentido, pensa-se que uma DIP, tendo como público alvo estudantes trabalhadores, não deva ter apenas 80 horas aula como ocorre na disciplina de LPI. Acredita-se que se a disciplina de LPI tivesse mais horas, a qualidade do atendimento do docente para com seus estudantes aumentaria. Além disso, os discentes teriam mais tempo para resolução das atividades em sala, contando com o auxílio do professor, visto que trabalham e dificilmente conseguem se dedicar à disciplina fora do horário de aula.

Em contrapartida, o índice de reprovação aumentou, conforme mostra o Gráfico 24. Isso ocorreu devido ao fato de ter diminuído o índice de evasão, sendo assim, alguns estudantes mesmo não indo tão bem, acompanharam a disciplina até o final. Logo, apesar desta taxa ter aumentado, o dado mostra-se positivo, visto que a dificuldade encontrada pelos reprovados não foi determinante para evadirem a disciplina.

Gráfico 24 - Comparativo de reprovação (2016/1, 2016/2 e 2017/1)



Fonte: A autora

Além de auxiliar o docente na construção de materiais de aula, a ConsProg também apresenta um RB que teve como o intuito de sistematizar o acompanhamento do desenvolvimento da aprendizagem dos estudantes. Com esse acompanhamento, o professor também pode identificar possíveis dificuldades, bem como a identificação de quais inferir sobre os conteúdos que podem ser causadores dessas barreiras. Devido ao curto tempo, contou-se apenas com dados de 20 estudantes para o processo de aprendizagem sobre as hipóteses de aprendizado dos CIP.

Essa pequena quantidade faz com que a RB apresente uma margem de erro muito grande. Porém, a intenção é de continuar alimentando a RB, para que as taxas de erros diminuam gradativamente e assim contar com hipóteses mais reais.

Pretende-se ainda implementar essa RB em um sistema on-line, conforme descrito anteriormente.

Um aspecto observado durante a aplicação da ConsProg, é de que ela pode auxiliar no apontamento de estudantes que necessitam de estudos de recuperação, já que os docentes são incumbidos de “estabelecer estratégias de recuperação para os alunos de menor rendimento” (BRASIL, 1996, art. 13). E de acordo com a Lei nº9.394 de 20 de dezembro de 1996 sobre a verificação do rendimento escolar, define que os estudos de recuperação devem ser oferecidos, obrigatoriamente, pelas instituições, e de preferência paralelamente ao período letivo (BRASIL, 1996).

Por fim, acredita-se que, apesar de ter sido desenvolvida para uma disciplina específica (Linguagem de Programação I), a proposta pedagógica aqui apresentada pode ser utilizada em outras áreas, visto que se trata de recomendações para fácil acompanhamento do desenvolvimento de aprendizagem dos estudantes. Tal acompanhamento depende também da sensibilidade e experiência do docente que, no acompanhamento de sua turma, poderá realizar refinamentos e adequações para melhor atender seu conjunto de estudantes.

REFERÊNCIAS

ARANHA, Maria Lúcia de Arruda. **História da educação e da pedagogia**: geral e Brasil. 3. ed. rev. e ampl., São Paulo: Moderna 2006.

BAYESFUSION, LCC. **GeNIe Modeler –** User manual. 2017. Disponível em: < http://support.bayesfusion.com/docs/genie.pdf> Acesso em: 08 fev. 2017.

BARBOSA, Alexandre de A.; et al. **Influência da linguagem no ensino introdutório de programação**. p. 612-621. *In*: Anais do III Congresso Brasileiro de Informática na Educação (CBIE 2014), XXV Simpósio Brasileiro de Informática na Educação (SBIE, 2014), 2014.

BLOOM, Benjamin Seymour; et al. **Taxonomia de objetivos educacionais**: 1 domínio cognitivo. Trad. Flávia Maria Sant´Anna. Porto Alegre: Editora Globo, 1972.

BOULAY, DU. **Some difficulties of learning to program.** *In* Soloway & Spohrer: Studying the Novice Programmer. p. 283-300. 1989.

BRASIL. **LEI Nº 11.892, DE 29 DE DEZEMBRO DE 2008.** Disponível em: < http://www.planalto.gov.br/ccivil_03/_ato2007-2010/2008/lei/l11892.htm> Acesso em: 10 ago. 2016.

BRASIL. **LEI Nº 9.394, DE 20 DE DEZEMBRO DE 1996.** Disponível em: < http://www.planalto.gov.br/ccivil_03/leis/L9394.htm> Acesso em: 20 dez. 2017.

BRASIL. **Decreto Nº 7.566 DE 23 DE SETEMBRO DE 1909**. Disponível em < http://www2.camara.leg.br/legin/fed/decret/1900-1909/decreto-7566-23-setembro-1909-525411-publicacaooriginal-1-pe.html> Acesso em 10 fev. 2016.

BRASSCOMM - Associação Brasileira das Empresas de Tecnologia da Informação e Comunicação. **O Mercado de Profissionais de TI no Brasil**. Disponível em: <http://www.brasscom.org.br/brasscom/portugues>.Acesso em: 12 nov. 2015.

CARVALHO, Leandro S. G.; GADELHA, Bruno F.; NAKAMURA, Fabíola G.; et al.. **Ensino de programação para futuros não-programadores:** contextualizando os exercícios com as demais disciplinas de mesmo período letivo. p.2116-2125. In: XXXVI Congresso da Sociedade Brasileira de Computação (CSBC 2016), XXIV Workshop sobre Educação em Computação (WEI 2016), 2016.

CHENG, Jie; BELL, David A.; LIU, Weiru. **An Algorithm for Bayesian Belief Network Construction from Data**. In: Proceedings of AI & STAT'97, 1998.

COOPER, Gregory F.; HERSKOVITS, Edward. **A Bayesian method for the induction of probabilistic networks from data**. In: Machine learning, v. 9, n. 4, p. 309-347, 1992.

CRESWELL, John W. **Projeto de pesquisa:** métodos qualitativo, quantitativo e misto. 3.ed. Porto Alegre: Artmed, 2010.

DEITEL, Paul; DEITEL, Harvey. **C**: como programar. 6ª ed. Pearson Education – Br, 2011.

DELGADO, Carla; XEXÉO, José Antônio Moreira; SOUZA, Isabel Fernandes de; et al.. **Uma abordagem pedagógica para iniciação ao estudo de algoritmos.** In: Anais do IX WEI - Workshop de Educação em Informática, 2004.

FONSECA, João José Saraiva da. **Metodologia da pesquisa científica.** Fortaleza: UEC, 2002. Apostila. Disponível em <http://www.ia.ufrrj.br/ppgea/conteudo/conteudo-2012-1/1SF/Sandra/apostilaMetodologia.pdf> Acesso em 10 jul. 16.

FRIEDMAN, N.; GEIGER, D.; GOLDSZMIDT, M. **Bayesian network classifiers**. Machine Learning, 29(2-3):131–163, 1997.

GARCÍA, P., AMANDI, A., SCHIAFFINO, S., CAMPO, M. **Evaluating Bayesian Networks precision for detecting students learning styles**. In: Computers & Education. Volume 49, no. 3, pp. 794–808. Elsevier. Buenos Aires.

GOMES, Marina; BECKER, Liliane; GESTARO, Lucas; et al. **Um estudo sobre erros em programação** - Reconhecendo as dificuldades de programadores iniciantes. p.1398-1407. Anais dos Workshops do IV Congresso Brasileiro de Informática na Educação (CBIE 2015), 2015.

GONZÁLEZ, Sahudy Montenegro; TAMARIZ, Annabell del Real. **Integração de uma metodologia de ensino presencial de programação com um sistema tutor inteligente.** In: Revista Brasileira de Informática na Educação, Volume 22, Número 2, 2014.

IFRS, Campus Porto Alegre. **Projeto Pedagógico do Curso Superior de Tecnologia em Sistemas para Internet**. Coord. Alex Martins de Oliveira; Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Porto Alegre: 2016. Disponível em: <http://www.poa.ifrs.edu.br/wp-content/uploads/2014/01/ppc_sistemas_internet_curriculo_2014.pdf> Acesso em 24 fev. 2017.

IFRS; **RESOLUÇÃO Nº 046 DE 08 DE MAIO DE 2015**. Organização Didática do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul. 2015

IFRS, Campus Porto Alegre. **Projeto Pedagógico do Curso Superior de Tecnologia em Sistemas para Internet**. Coord. Tanisi Pereira de Carvalho; Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Porto Alegre: 2013. Disponível em: <http://www.poa.ifrs.edu.br/wp-content/uploads/2014/01/ppc_sistemas_internet_curriculo_2014.pdf> Acesso em 14 jun. 2016.

IFRS, Campus Porto Alegre; **100 anos de história:** Da Escola Técnica da UFRGS ao Campus Porto Alegre do IFRS – 1909 a 2009; Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – campus Porto Alegre: 2011. Disponível em: <http://www.poa.ifrs.edu.br/institucional/a-instituicao/do-campus-porto-alegre/pequeno-historico> Acesso em 12 dez. 2016

INEP; **Censo da educação superior 2013:** resumo técnico. – Brasília: Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira, 2015.

KORB, K. B.; NICHOLSON, A. E.. **Bayesian artificial intelligence**. London: Chapman & Hall/CRC Press UK, 2004.

KUENZER, Acacia Zeneida. **Ensino médio:** construindo uma proposta para os que vivem do trabalho. 2. ed. São Paulo: Cortez, 2001

LAHTINEN,E.; ALA-MUTKA,K.; JÄRVINEN,H.-M. **A study of the difficulties of novice programmers**. Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education - ITiCSE '05, 2005.

LUGER, George F. **Inteligência Artificial**. Trad. Daniel Vieira; Revisão técnica Andréia Iabrudi Tavares – 6ª ed. – São Paulo: Pearson Education do Brasil, 2013.

MEC. **Catálogo Nacional de Cursos Superiores de Tecnologia.** 3ª ed. Ministério da Educação: Brasília/DF, 2016. Disponível em <http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=44501-cncst-2016-3edc-pdf&category_slug=junho-2016-pdf&Itemid=30192> Acesso em: 25 nov. 16.

MEC. **Diretrizes Curriculares Nacionais para os cursos de graduação em Computação.** Ministério da Educação: Brasília/DF, 2012. Disponível em <http://portal.mec.gov.br/index.php?option=com_docman&view=download&alias=11205-pces136-11-pdf&category_slug=julho-2012-pdf&Itemid=30192> Acesso em: 03 dez. 2016.

MENEZES, Crediné Silva de; NOBRE, Isaura Alcina Martins. **Um ambiente cooperativo para apoio a cursos de introdução a programação**. In: Congresso da Sociedade Brasileira de Computação, X Workshop sobre Educação em Computação, 2002.

MOLL, Jaqueline; e colaboradores. **Educação profissional e tecnológica no Brasil contemporâneo**: desafios, tensões e possibilidades. Porto Alegre: Artmed, 2010.

NEAPOLITAN, R. E. **Learning Bayesian Networks**. Upper Saddle River: Pearson, 2004

OKUYAMA, Fabio Yoshimitsu; MILETTO, Evandro Manara; NICOLAO, Mariano. **Desenvolvimento de software I:** conceitos básicos. Porto Alegre: Bookman, 2014.

PACHECO, Eliezer; **Institutos Federais** – Uma revolução na educação profissional e tecnológica. São Paulo: Moderna, 2011.

PAPERT, Seymour. **LOGO:** Computadores e Educação. São Paulo: Brasiliense, 1985.

PIAGET, Jean. **Sobre a Pedagogia**: textos inéditos. Tradução de Claudia Berliner. São Paulo: Casa do Psicólogo, 1998.

_____. **Abstração reflexionante:** relações logico-aritméticas e ordem das relações espaciais. Porto Alegre: Artmed, 1995.

_____. **A epistemologia genética/ Sabedoria e ilusões da filosofia; Problemas de psicologia genética**. In: Piaget. Traduções de Nathanael C. Caixeiro, Zilda Abujamra Daeir, Celia E. A. Di Pier. 2ªed. SãoPaulo: Abril Cultural, 1983. (Os pensadores)

_____. **A tomada de consciência**. São Paulo: EDUSP/Melhoramentos, 1977.

_____. **A equilibração das estruturas cognitivas:** problema central do desenvolvimento. Tradução de Marion M. dos S. Penna. Rio de Janeiro: Zahar, 1976.

_____. **O nascimento da inteligência na criança**. Tradução de A. Cabral. Rio de Janeiro: Zahar, 1975.

_____. **A psicologia da criança.** São Paulo: Difel, 1974.

RAABE, André Luís Alice; ZANINI, Adriana Salvador; SANTANA, André Luiz Maciel; et al.. **Influência dos enunciados na resolução de problemas de programação introdutória**. p.66-82. In: Revista Brasileira de Informática na Educação (RBIE), v. 24, nº 1, 2016.

REIS FILHO, Casemiro. **A educação e a ilusão liberal**. São Paulo, Cortez/Autores Associados, 1981.

RENUMOL, V. G.; JANAKIRAM, Dharanipragada; JAYAPRAKASH, S. **Identification of Cognitive Processes of Effective and Ineffective Students During Computer Programming.** In: ACM Transactions on Computing Education, Vol. 10, No. 3, Article 10, Pub. date: August 2010.

SEFFRIN, Henrique M.; JAQUES, Patricia. **Avaliando o conhecimento algébrico dos estudantes através de Redes Bayesianas Dinâmicas**. In: Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE), 2015, p.987 – 996,

SEVELLA, Pranay Kumar; LEE, Young; YANG, Jeong. **Determining The Barriers Faced By Novice Programmers**. INTERNATIONAL JOURNAL OF SOFTWARE ENGINEERING (IJSE), p. 10, 2013.

SOUZA, Draylson Micael; BATISTA, Marisa Helena da Silva; BARBOSA, Ellen Francine. **Problemas e Dificuldades no Ensino e na Aprendizagem de Programação**: Um Mapeamento Sistemático. In: Revista Brasileira de Informática na Educação, Volume 24, Número 1, 2016, p. 39 – 52.

SOUZA, Anderson Luiz Ara. **Redes Bayesianas:** uma introdução aplicada a credit scoring. In: 19º Simpósio Nacional de Probabilidade e Estatística (SINAPE), 2010.

SPIRTES, P.; GLYMOUR, C.; SCHEINES, R. **An algorithm for fast recovery of sparse causal graphs.** Social Science Computer Review, v. 9, p. 62-72, 1991.

VASCONCELLOS, Celso dos S. **Planejamento Projeto de Ensino-Aprendizagem e Projeto Político-Pedagógico.** Ladermos Libertad-1. 7º Ed. São Paulo, 2000.

VIEGAS, Thaís Ramos. **O uso de compiladores no ensino de algoritmos.** TCC (Especialização em Educação Básica Profissional) - Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul, Campus Osório, 2015. Disponível em <http://pergamum.ifrs.edu.br:8080/pergamumweb/vinculos/000030/000030b9.pdf> Acesso em: 04 jul. 2016.

VIER, Juliano; GLUZ, João; JAQUES, Patrícia A.; **Empregando Redes Bayesianas para modelar automaticamente o conhecimento dos aprendizes em lógica de**

programação. In: Revista Brasileira de Informática na Educação, Volume 23, Número 2, 2015, p. 45 – 59.

WINSLOW, L. E. **Programming Pedagogy** - A Psychological Overview, ACM SIGCSE Bulletin, v. 28, n. 3, pp 17-25, 1996.

WOLFF, Denise Luzia; SILVA, Cristina Almeida da; TAVARES, Mara Rosane Noble. **Forma de ingresso X situação dos alunos no curso superior de Tecnologia em Sistemas para Internet do IFRS**. # Tear: Revista de Educação, Ciência e Tecnologia, v. 5, n. 2, 2016. Disponível em < https://periodicos.ifrs.edu.br/index.php/tear/article/view/1997> Acesso em 09 fev. 2017.

APÊNDICE A – QUESTIONÁRIO FECHADO PRIMEIRA ESTUDO DE CASO

Levantamento de informações

O intuito desse formulário é obter informações a respeito do ensino-aprendizagem de programação em linguagem C, na disciplina de programação I, ministrada pelo Professor Drº Fabio Yoshimitsu Okuyama, no primeiro semestre de 2016.

* Required

1. **Qual seu sexo?** *
   Mark only one oval.
   - ◯ Feminino
   - ◯ Masculino

2. **Qual sua faixa etária?** *
   Mark only one oval.
   - ◯ Até 18 anos
   - ◯ De 19 à 25 anos
   - ◯ De 26 à 30 anos
   - ◯ De 31 à 40 anos
   - ◯ Acima de 40 anos
   - ◯ Prefiro não responder

3. **Você trabalha?** *
   Mark only one oval.
   - ◯ Sim
   - ◯ Não
   - ◯ Prefiro não responder

4. **Caso tenha respondido sim na pergunta anterior, quais turnos você trabalha?** *
   Check all that apply.
   - ☐ Manhã
   - ☐ Tarde
   - ☐ Noite
   - ☐ Madrugada
   - ☐ Prefiro não responder

5. **Você tem algum tempo livre para se dedicar ao curso?** *
   Mark only one oval.
   - ◯ Sim
   - ◯ Não
   - ◯ Prefiro não responder

6. **Você já tinha algum conhecimento de programação antes de ingressar no Cursos Superior de Sistemas para Internet?** *
Mark only one oval.

- ◯ Não
- ◯ Sim
- ◯ Prefiro não responder

7. **Caso tenha respondido sim na questão anterior, qual nível definiria esse conhecimento?**
Mark only one oval.

- ◯ Básico
- ◯ Intermediário
- ◯ Avançado
- ◯ Prefiro não responder

8. **Você já havia programado em uma dessas linguagens abaixo? (Responda apenas se você marcou sim na primeira pergunta)**
Check all that apply.

- ☐ Java
- ☐ C
- ☐ C++
- ☐ C#
- ☐ Java Script
- ☐ PHP
- ☐ Prefiro não responder
- ☐ Other: _____

9. **Você foi assíduo nas aulas, comprometendo-se com os horários?** *
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder
- ◯ Other: _____

10. **Como você classificaria sua participação em aula?** *
Mark only one oval.

- ◯ Prestava atenção nas explicação e fazia todas as atividades.
- ◯ Prestava atenção nas explicações e fazia apenas os exemplos que o professor apresentava.
- ◯ Prestava atenção nas explicações apenas.
- ◯ Fazia apenas as atividades.
- ◯ Copiava apenas os exemplos apresentado pelo professor e as atividades que o professor corrigia no quadro.
- ◯ Dificilmente participava.
- ◯ Prefiro não responder

11. **Você tirava todas as suas dúvidas com o professor?** *
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder

12. **Em relação as listas de atividades, quantos porcentos mais ou menos você fez?** *
Mark only one oval.

- ◯ Menos de 10%
- ◯ Em torno de 10%
- ◯ Em torno de 25%
- ◯ Em torno de 50%
- ◯ Em torno de 70%
- ◯ Em torno de 90%
- ◯ Sempre fez todas as atividades
- ◯ Prefiro não responder

13. **Quanto tempo você dedicava à disciplina semanalmente (sem considerar o horário de sala de aula)?** *
Mark only one oval.

- ◯ Menos de 2h
- ◯ Em torno de 2h
- ◯ Em torno de 3h
- ◯ Mais de 3h
- ◯ Prefiro não responder

14. **Você buscou conhecimento fora de sala de aula (video aula, livros, colegas, outros professores)?** *
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder

15. **Que tipo de dificuldade encontrava ao resolver as atividades?** *
Check all that apply.

- ☐ Compreender o que a atividade queria.
- ☐ Criar as fórmulas para solução das atividades.
- ☐ Estruturar o código na linguagem de programação.
- ☐ Fazer a compilação do programa.
- ☐ Prefiro não responder
- ☐ Other:

16. **Você considera que aprendeu todas as noções básicas de programação na linguagem C?** *

(variáveis; variáveis homogêneas (vetores e matriz); variáveis heterogêneas (registros); operações e expressões matemáticas e lógicas; estrutura de controle; funções; procedimentos; variáveis locais e globais; passagem por parâmetro e referência; tratamento de arquivos)
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder
- ◯ Other: _____

17. **É a primeira vez que cursa a disciplina de programação I? Se não, comente quantas vezes já cursou a mesma.** *
Mark only one oval.

- ◯ Sim
- ◯ Prefiro não responder
- ◯ Other: _____

APÊNDICE B - QUESTIONÁRIO FECHADO SEGUNDO ESTUDO DE CASO

Levantamento de informações

O intuito desse formulário é obter informações a respeito do ensino-aprendizagem de programação em linguagem C, na disciplina de programação I, ministrada pelo Professor Drº Fabio Yoshimitsu Okuyama, no segundo semestre de 2016, com o auxílio dos artefatos online (questionários, dicas, slides, etc) utilizados como apoio das aulas presenciais.

* Required

1. **Qual seu sexo?** *
   Mark only one oval.
   - ◯ Feminino
   - ◯ Masculino

2. **Qual sua faixa etária?** *
   Mark only one oval.
   - ◯ Até 18 anos
   - ◯ De 19 à 25 anos
   - ◯ De 26 à 30 anos
   - ◯ De 31 à 40 anos
   - ◯ Acima de 40 anos
   - ◯ Prefiro não responder

3. **Você trabalha?** *
   Mark only one oval.
   - ◯ Sim
   - ◯ Não
   - ◯ Prefiro não responder

4. **Caso tenha respondido sim na pergunta anterior, quais turnos você trabalha?** *
   Check all that apply.
   - ☐ Manhã
   - ☐ Tarde
   - ☐ Noite
   - ☐ Madrugada
   - ☐ Prefiro não responder

5. **Você tem algum tempo livre para se dedicar ao curso?** *
   Mark only one oval.
   - ◯ Sim
   - ◯ Não
   - ◯ Prefiro não responder

6. **Você já tinha algum conhecimento de programação antes de ingressar no Cursos Superior de Sistemas para Internet?** *
Mark only one oval.

- ◯ Não
- ◯ Sim
- ◯ Prefiro não responder

7. **Caso tenha respondido sim na questão anterior, qual nível definiria esse conhecimento?**
Mark only one oval.

- ◯ Básico
- ◯ Intermediário
- ◯ Avançado
- ◯ Prefiro não responder

8. **Você já havia programado em uma dessas linguagens abaixo? (Responda apenas se você marcou sim na primeira pergunta)**
Check all that apply.

- ☐ Java
- ☐ C
- ☐ C++
- ☐ C#
- ☐ Java Script
- ☐ PHP
- ☐ Prefiro não responder
- ☐ Other: _____

9. **É a primeira vez que cursa a disciplina de programação I? Se não, comente quantas vezes já cursou a mesma.** *
Mark only one oval.

- ◯ Sim
- ◯ Prefiro não responder
- ◯ Other: _____

10. **Você foi assíduo nas aulas, comprometendo-se com os horários?** *
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder
- ◯ Other:

11. **Como você classificaria sua participação em aula?** *
Mark only one oval.

- ◯ Prestava atenção nas explicação e fazia todas as atividades.
- ◯ Prestava atenção nas explicações e fazia apenas os exemplos que o professor apresentava.
- ◯ Prestava atenção nas explicações apenas.
- ◯ Fazia apenas as atividades.
- ◯ Copiava apenas os exemplos apresentado pelo professor e as atividades que o professor corrigia no quadro.
- ◯ Dificilmente participava.
- ◯ Prefiro não responder

12. **Você tirava todas as suas dúvidas com o professor?** *
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder

13. **Em relação as listas de atividades, quantos porcentos mais ou menos você fez?** *
Mark only one oval.

- ◯ Menos de 10%
- ◯ Em torno de 10%
- ◯ Em torno de 25%
- ◯ Em torno de 50%
- ◯ Em torno de 70%
- ◯ Em torno de 90%
- ◯ Sempre fez todas as atividades
- ◯ Prefiro não responder

14. **Quanto tempo você dedicava à disciplina semanalmente (sem considerar o horário de sala de aula)?** *
Mark only one oval.

- ◯ Menos de 2h
- ◯ Em torno de 2h
- ◯ Em torno de 3h
- ◯ Mais de 3h
- ◯ Prefiro não responder

15. **Você utilizou os artefatos online indicados pelo professor?** *
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder

16. **As atividades encontradas nos artefatos online lhe auxiliaram na disciplina?** *
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder

17. **O material de apoio encontrado nos artefatos online lhe auxiliaram na disciplina?** *
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder

18. **Você acredita que seu aprendizado seria o mesmo caso não fizesse o uso desses artefatos online?** *
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder
- ◯ Other: _____

19. **Os artefatos eram fáceis de serem utilizados?** *
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder

20. **O layout dos artefatos eram bons?** *
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder

21. **Você indicaria a utilização desses artefatos para as próximas turmas?** *
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder

22. **Ao realizar as atividades de programação, que tipo de dificuldade encontrava ao resolver as atividades?** *

Check all that apply.

- [ ] Compreender o que a atividade queria.
- [ ] Criar as fórmulas para solução das atividades.
- [ ] Estruturar o código na linguagem de programação.
- [ ] Fazer a compilação do programa.
- [ ] Prefiro não responder
- [ ] Other: _____

23. **Você considera que aprendeu todas as noções básicas de programação na linguagem C?** *

(variáveis; variáveis homogêneas (vetores e matriz); variáveis heterogêneas (registros); operações e expressões matemáticas e lógicas; estrutura de controle; funções; procedimentos; variáveis locais e globais; passagem por parâmetro e referência; tratamento de arquivos)
Mark only one oval.

- ◯ Sim
- ◯ Não
- ◯ Prefiro não responder
- ◯ Other: _____

24. **Você gostaria de deixar mais alguma informação à respeito do semestre?** *

_____

_____

_____

APÊNDICE C - MODELO DO TCLE UTILIZADO NO ESTUDO DE CASO MÚLTIPLO

TERMO DE CONSENTIMENTO LIVRE E ESCLARECIDO

Você está sendo convidado(a) como voluntário(a) a participar da pesquisa “Uma abordagem pedagógica para uso de TICs no ensino-aprendizagem de programação”, que constitui a Trabalho de Conclusão de Curso de Mestrado Profissional em Informática na Educação da aluna Thaís Ramos Viegas, orientada pelo Fabio Yoshimitsu Okuyama.

A JUSTIFICATIVA, OS OBJETIVOS E OS PROCEDIMENTOS: Apesar da demanda por profissionais qualificados na área de Tecnologia da Informação (TI) crescerem constantemente e apresentarem salários bastante atrativos, os índices de evasão nos cursos TI são elevados, apontando que estudos sobre as causas deste fenômeno se mostram importantes e urgentes. Nesse sentido, a presente pesquisa tem como principal objetivo criar uma proposta pedagógica para o ensino-aprendizagem de programação com auxílio das TICs. Para a elaboração da proposta pedagógica, o estudo iniciará com um levantamento bibliográfico sobre diversos temas relacionados com o trabalho, seguido de uma pesquisa documental em ementas e planos de ensino de disciplinas de programação afim de vincular a proposta com os elementos e objetivos da disciplina. Além disso, será realizado estudo de caso múltiplo em duas fases, com alunos do curso Superior em Sistemas para Internet do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul - Campus Porto Alegre, no ano de 2016. Na primeira fase, pretende-se levantar as principais falhas na relação do processo de ensino-aprendizagem de programação. Já na segunda fase, em uma nova turma do mesmo curso, serão aplicados artefatos através da plataforma online com o intuito de auxiliar as aulas presenciais. Esses artefatos correspondem a atividades criadas a partir dos conceitos da Taxionomia de Bloom, bem como materiais explicativos sobre o conteúdo a fim de auxiliar nas aulas de programação**.** No final do de cada estudo de caso, será realizado um questionário fechado com alunos e professores participantes da pesquisa, para complementar a verificação dos resultados obtidos com e sem a utilização dos artefatos como suporte das aulas presenciais. Espera-se que com esta proposta pedagógica, juntamente com os artefatos TICs desenvolvidos, possa propiciar aos alunos e professores uma experiência que auxilie no desenvolvimento no processo de ensino-aprendizagem de programação. Podendo diminuir o índice de evasão e retenção na disciplina de programação.

DESCONFORTOS E RISCOS E BENEFÍCIOS: Esta atividade pode trazer como risco a seus sujeitos participantes o não auxílio às aulas, caso isso ocorra, as aulas seguirão normalmente sem o uso dos artefatos. Ainda pode-se ter como risco aos sujeitos participantes o constrangimento ao responder os questionários fechados, podendo estes optar por não participar das atividades e/ou dos questionários.

GARANTIA DE ESCLARECIMENTO, LIBERDADE DE RECUSA E GARANTIA DE SIGILO: Você será esclarecido(a) sobre a pesquisa em qualquer aspecto que desejar. Você é livre para recusar-se a participar, retirar seu consentimento ou interromper a participação a qualquer momento. A sua participação é voluntária e a recusa em participar não irá acarretar qualquer penalidade ou perda de benefícios.

A pesquisadora irá tratar a sua identidade com padrões profissionais de sigilo. Os resultados permanecerão confidenciais. Seu nome ou o material que indique a sua participação não será divulgado. Você não será identificado(a) em nenhuma publicação que possa resultar deste estudo. Uma cópia deste consentimento informado será arquivada na Coordenação do Curso de Mestrado Profissional em Informática na Educação; e outra será fornecida a você.

CUSTOS DA PARTICIPAÇÃO, RESSARCIMENTO E INDENIZAÇÃO POR EVENTUAIS DANOS: A participação no estudo não acarretará custos para você e não será disponível nenhuma compensação financeira adicional.

DECLARAÇÃO DO PARTICIPANTE OU DO RESPONSÁVEL PELO PARTICIPANTE: Eu, _____, fui informada (o) dos objetivos da pesquisa acima de maneira clara e detalhada e esclareci minhas dúvidas. Sei que em qualquer momento poderei solicitar novas informações e motivar minha decisão se assim o desejar. A aluna pesquisadora Thaís Ramos Viegas e o professor orientador Fabio Yoshimitsu Okuyama certificaram-me de que todos os dados desta pesquisa serão confidenciais.

Em caso de dúvidas poderei chamar a aluna de mestrado Thaís Ramos Viegas, o professor orientador Fabio Yoshimitsu Okuyama ou o Diretor de pesquisa ou Coordenação do curso de Mestrado Profissional em Informática na Educação através dos e-mails pdpi@poa.ifrs.edu.br ou andre.peres@poa.ifrs.edu.br, ou dos telefones (51) 3930-6019 ou (51) 3930-6072. Também é possível encontrá-los no Câmpus Porto Alegre do IFRS, situado na Rua Cel. Vicente, 281, Bairro Centro, CEP 90.030-041, Porto Alegre/RS.

Declaro que concordo em participar desse estudo. Recebi uma cópia deste Termo de Consentimento Livre e Esclarecido e me foi dada a oportunidade de ler e esclarecer as minhas dúvidas.

Nome	Assinatura do Participante	Data
Nome	Assinatura do Pesquisador	Data
Nome	Assinatura da Direção de Pesquisa	Data

APÊNDICE D – MODELO DO TERMO DE ACEITE INSTITUCIONAL

TERMO DE AUTORIZAÇÃO INSTITUCIONAL

Porto Alegre, 15 de junho de 2016.

Ilustríssimo (a) Senhor (a)

Eu, *Thaís Ramos Viegas,* responsável principal pela Dissertação do Mestrado Profissional em Informática na Educação, venho, pelo presente, solicitar sua autorização para realizar este projeto de pesquisa no *Instituto Federal Câmpus Porto Alegre*, intitulado *Uma proposta pedagógica para uso de TICs no ensino-aprendizagem de programação,* e orientado pelo Professor Drº Fabio Yoshimitsu Okuyama*.*

Este projeto de pesquisa tem como objetivo *criar uma abordagem pedagógica com uso de TICs para auxiliar no ensino aprendizagem de programação*. Os procedimentos adotados serão: levantamento bibliográfico sobre os diversos temas relacionados ao trabalho, entre os quais podemos citar: ensino-aprendizagem, abordando os teóricos da educação e conceitos que norteiam e embasam o trabalho. Em especial, a aplicação de conceitos de Jean Piaget para o ensino-aprendizagem de programação; trabalhos correlatos sobre o uso de TICs na educação, trazendo o panorama do estado-da-arte na área da área da pesquisa; conceitos introdutórios do ensino de programação.

Adicionalmente será realizada pesquisa documental em ementas e planos de ensino de disciplinas de programação, visto que as ferramentas e metodologias deverão ser complementares ao cronograma e ementa da disciplina. Além disso, será utilizado estudo de caso múltiplo em duas fases, na primeira fase será realizado o acompanhamento em sala de aula com estudantes do curso de Tecnologia em Sistemas para Internet do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS) - Campus Porto Alegre, na disciplina de Programação I. No final desse acompanhamento, será aplicado um questionário fechado com os estudantes para complementar e ilustrar quantitativamente a investigação desse acompanhamento.

Na segunda fase do estudo de caso múltiplo, aplicado em nova turma do mesmo curso e disciplina, serão aplicados os artefatos através de plataforma online de maneira complementar as aulas presenciais. Esses artefatos compreendem atividades criadas a partir da Taxionomia de Bloom, bem como materiais explicativos sobre o conteúdo a fim de auxiliar nas aulas de programação. Sendo assim, os professores da disciplina de

programação irão utilizar os artefatos online com seus estudantes durante e/ou após as aulas.

Nesta fase, o estudo de caso múltiplo será usado para acompanhar o uso dos artefatos propostos no processo de ensino-aprendizagem. Posteriormente será feito um questionário fechado, tanto com os professores quanto com os estudantes, para complementar a verificação do uso dos artefatos e seus resultados em sala de aula. Os participantes da pesquisa serão os estudantes, maiores de idade, e professores da disciplina de Programação I, do Curso Superior de Sistemas para Internet.

Espera-se, com esta pesquisa, que a proposta pedagógica para o ensino-aprendizagem de programação aliada aos artefatos TICs desenvolvidos, proporcione aos estudantes e professores do curso Superior em Sistemas para Internet do IFRS - Campus Porto Alegre uma experiência diferenciada e mais efetiva no ensino-aprendizagem de programação. Auxiliando o professor no acompanhamento de desenvolvimento do aprendizado de seus estudantes. Proporcionando ao estudante atividades dirigidas e materiais de apoio. Podendo assim, diminuir o índice de evasão e retenção na disciplina de programação.

O período de geração de dados compreende de 04 de julho a 21 de dezembro de 2016. Qualquer informação adicional poderá ser obtida no IFRS – Câmpus Porto Alegre, diretamente com a Diretoria de Pesquisa ou Coordenação do Curso de Pós-graduação através dos e-mails dpi@poa.ifrs.edu.br ou andre.peres@poa.ifrs.edu.br ou dos telefones (51) 3930-6019 ou (51) 3930-6072; e também com os pesquisadores através do thais.viegas@gmail.com ou (51) 95526877.

Esta atividade pode trazer como risco a seus sujeitos participantes o não auxílio às aulas, caso isso ocorra, as aulas seguirão normalmente sem o uso dos artefatos. Ainda pode-se ter como risco aos sujeitos participantes o constrangimento ao responder os questionários fechados, podendo estes optar por não participar das atividades e/ou dos questionários, ou ainda indicarem que preferem não responder aquele questionamento.

As atividades poderão trazer benefícios como:

- Alunos: irão se beneficiar de artefatos para complementar seus estudos em programação;
- Professores: estes terão acesso à informações sobre o aprendizado dos alunos, podendo realizar intervenções para que estes atinjam melhor rendimento;
- Instituição de ensino: caso os artefatos auxiliem no processo de ensino aprendizagem, espera-se uma redução da retenção e consequente evasão.

- Sociedade: com melhoria nas taxas de sucesso no curso haverá melhor aproveitamento de recursos públicos envolvidos no processo de formação dos alunos.
- Área de conhecimento: havendo sucesso do uso da abordagem e seus artefatos, tal abordagem poderá ser generalizada para, eventualmente, auxiliar no ensino-aprendizagem de outros conteúdos.

A qualquer momento Vossa Senhoria poderá solicitar esclarecimento sobre o desenvolvimento do projeto de pesquisa que está sendo realizado e, sem qualquer tipo de cobrança, poderá retirar sua autorização. Os pesquisadores estão aptos a esclarecer estes pontos e, em caso de necessidade, dar indicações para solucionar ou contornar qualquer mal estar que possa surgir em decorrência da pesquisa.

Os dados obtidos nesta pesquisa NÃO serão utilizados na publicação de artigos científicos e que, assumimos a total responsabilidade de não publicar qualquer dado que comprometa o sigilo da participação dos integrantes de sua instituição como nome, endereço e outras informações pessoais não serão em hipótese alguma publicados. A participação de sua instituição será voluntária, portanto, não fornecemos por ela qualquer tipo de pagamento.

Autorização Institucional

Eu, Marcelo Augusto Rauh Schmitt, diretor geral no Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – Câmpus Porto Alegre, declaro que fui informado dos objetivos da pesquisa acima, e concordo em autorizar a execução da mesma nesta instituição. Caso necessário, a qualquer momento como instituição CO-PARTICIPANTE desta pesquisa poderemos revogar esta autorização, se comprovada atividades que causem algum prejuízo à esta instituição ou ainda, a qualquer dado que comprometa

o sigilo da participação dos integrantes desta instituição. Declaro também, que não recebemos qualquer pagamento por esta autorização bem como os participantes também não receberão qualquer tipo de pagamento.

Pesquisador:	Responsável pela Instituição:
Thaís Ramos Viegas	Marcelo Augusto Rauh Schmitt

Orientador:
Fabio Yoshimitsu Okuyama

Documento em duas vias:

1ª via instituição

2ª via pesquisadores

APÊNDICE E – ARTIGOS ENSINO-APRENDIZAGEM DE PROGRAMAÇÃO - 2014

EVENTO	QTD		TÍTULO TRABALHOS
XX Workshop de Informática na Escola			
WIE	71	1	Introdução à Programação e à Implementação de Processadores por Estudantes do Ensino Médio
		2	O ensino de programação para dispositivos móveis utilizando o MIT-App Inventor com alunos do ensino médio
		3	Produção e Avaliação de Videoaulas: Um Estudo de Caso no Ensino de Programação
		4	Um Estudo Observacional sobre a Disciplina Introdutória de Programação
XXV Simpósio Brasileiro de Informática na Educação			
SBIE	152	1	A importância da motivação dos estudantes e o uso de técnicas de engajamento para apoiar a escolha de jogos no ensino de programação
		2	A Importância dos Programas de Extensão no Ensino e Prática de Programação e Desenvolvimento de Protótipos
		3	Análise de Componentes Latentes da Aprendizagem de Programação para Mapeamento e Classificação de Perfis
		4	Curso Híbrido usando a Rede Social Facebook no Ensino de Programação de Computadores
		5	Influência da linguagem no ensino introdutório de programação
		6	Jogos Digitais para Ensino e Aprendizagem de Programação: uma Revisão Sistemática da Literatura
		7	KLouro: Um jogo educacional para motivar alunos iniciantes em programação
		8	LOGOBOT – Um Sistema Robótico Simulador da Linguagem Logo para Auxílio no Aprendizado de Programação
		9	Prog1Box: Um Ambiente Linux sem Distratores para o Aprendizado de Programação
		10	Relato da experiência do trabalho com jogos manuais de raciocínio lógico como reforço para as disciplinas de algoritmos e linguagem de programação
		11	Um Ambiente Virtual com Feedback Personalizado para Apoio a Disciplinas de Programação
		12	Um Arcabouço para Construção de Mecanismos de Análise de Códigos de Programação Introdutória
		13	Uma Metodologia Baseada em Semiótica para Elaboração e Análise de Práticas de Ensino de Programação com Robótica Pedagógica
Workshops do Congresso Brasileiro de Informática na Educação			
CBIE	91	1	Um Ambiente Virtual com Feedback Personalizado para Apoio a Disciplinas de Programação
		2	Pinte o 7: Sistema Interativo para o ensino de programação de computadores para o público infanto-juvenil

XXII Workshop sobre Educação em Computação			
WEI			Sem dados

IV Workshop de Desafios da Computação Aplicada à Educação			
DesafIE			Sem dados

REVISTA BRASILEIRA DE INFORMÁTICA NA EDUCAÇÃO			
RBIE	12	0	
	10	1	Integração de uma Metodologia de Ensino Presencial de Programação com um Sistema Tutor Inteligente
	11	0	

Jornada de Atualização em Informática na Educação			
JAI	4	0	

TOTAL	351	20	**Publicações em ensino de programação**

APÊNDICE F - ARTIGOS ENSINO-APRENDIZAGEM DE PROGRAMAÇÃO - 2015

EVENTO	QTD		TÍTULO TRABALHOS
XXI Workshop de Informática na Escola			
WIE	67	1	Ensino de programação para Olimpíada Brasileira de Informática
		2	Programação de Computadores e Robótica Educativa na Escola tendências evidenciadas nas produções do Workshop de Informática na Escola
		3	Robótica Educativa na aprendizagem de Lógica de Programação Aplicação e análise
		4	Investigando dois formatos de videoaulas de programação de jogos digitais para alunos do ensino médio
XXVI Simpósio Brasileiro de Informática na Educação			
SBIE	139	1	A Comparação da Realidade Mundial do Ensino de Programação para Iniciantes com a Realidade Nacional Revisão sistemática da literatura em eventos brasileiros
		2	A Teoria do Flow na contribuição do engajamento estudantil para apoiar a escolha de jogos no ensino de programação
		3	Análise dos efeitos do Pensamento Computacional nas habilidades de estudantes no ensino básico um estudo sob a perspectiva da programação de computadores
		4	Avaliando o Uso da Ferramenta Scratch para Ensino de Programação através de Análise Quantitativa e Qualitativa
		5	Caracterizando a pesquisa sobre autoavaliação na aprendizagem de programação para iniciantes
		6	Ensino de Lógica de Programação no Ensino Médio e suas implicações na Neurociências
		7	KidCoder Uma Proposta de Ensino de Programação de forma Lúdica
		8	Plataforma Robocode como Ferramenta Lúdica de Ensino de Programação de Computadores - Pesquisa e Extensão Universitária em Escolas Públicas de Minas Gerais
		9	Produção de Videoaulas de Programação em Java Acessíveis no Contexto de um Projeto de Capacitação Profissional para Pessoas Surdas
		10	RASPIBLOCOS Ambiente de Programação Didático Baseado em Raspberry Pi e Blockly
		11	Robótica Pedagógica Aplicada ao Ensino de Programação Uma Revisão Sistemática da Literatura
		12	Softwares Educacionais para o Ensino de Programação Um Mapeamento Sistemático
		13	Um Mapeamento Sistemático sobre Iniciativas Brasileiras em Ambientes de Ensino de Programação

Workshops do IV Congresso Brasileiro de Informática na Educação			
CBIE	170	1	Algo+ - Um app para o auxílio na apredizagem de programação
		2	Aprendendo linguagem de programação através da autoexplicação de exemplos em vídeo
		3	Ensinado Programação apoiada por um ambiente virtual e exercícios associados a cotidiano dos alunos compartilhando alternativas e lições aprendidas
		4	Estudo sobre o Sequenciamento Inteligente e Adaptativo de Enunciados em Programação de Computadores
		5	Experiência baseada em Gamificação no Ensino sobre Herança em Programação Orientada a Objetos
		6	Ferramenta para Auxílio na Aprendizagem de Lógica de Programação em Sistemas Informatizados
		7	Interdisciplinaridade, programação visual e robótica educacional relato de experiência sobre o ensino inicial de programação
		8	Jogos de Programar como uma Abordagem para os Primeiros Contatos dos Estudantes com à Programação
		9	Pinte o 7 sistema interativo de Programação Visual para ensino-aprendizagem integrável ao Moodle,
		10	Planejando um serious game para a prática de Programação
		11	Práticas de ensino de Programação de Computadores com Robótica Pedagógica e aplicação de Pensamento Computacional
		12	Um estudo sobre erros em programação Reconhecendo as dificuldades de programadores iniciantes
		13	Uma dinâmica para ensino de conceitos fundamentais de programação
		14	XP & Skills gamificando o processo de ensino de introdução a programação

XXIII Workshop sobre Educação em computação			
WEI	31	1	A Importância do Fator Motivacional no Processo Ensino-aprendizagem de programação
		2	Resgatando a Linguagem de Programacao Logo _ Uma experiência com calouros do ensino superior
		3	Saci – ainda outro ambiente para o ensino de programação
		4	Testando a Diversão em um Jogo Sério para o Aprendizado introdutório de programação
		5	Uma Abordagem Gamificada para o Ensino de programação orientada a objetos
		6	Uma ferramenta gamificada de apoio à disciplina introdutória de programação
		7	URI Online Judge Academic Integracao e Consolidacao da Ferramenta no Processo de EnsinoAprendizagem
		8	Uso de avaliac ̧ao por pares em disciplinas introdutorias de programação

IV Workshop de Desafios da Computação Aplicada à Educação			
DesafIE	15	1	Abordagens, Práticas e Desafios da Avaliação Automática de Exercícios de Programação

REVISTA BRASILEIRA DE INFORMÁTICA NA EDUCAÇÃO			
RBIE	15	1	Ensino-aprendizagem de programação: uma revisão sistemática da literatura
	15	0	
	13	1	Múltiplas Representações Externas no Suporte à Aquisição de Conhecimento em Programação de Computadores

Jornada de Atualização em Informática na Educação			
JAIE	5	1	Sobre cursos introdutórios de programação na modalidade MOOC utilizando Moodle

TOTAL	470	43	**Publicações em ensino de programação**

APÊNDICE G - ARTIGOS ENSINO-APRENDIZAGEM DE PROGRAMAÇÃO - 2016

EVENTO	QTD		TÍTULO TRABALHOS
XXII Workshop de Informática na Escola			
WIE	103	1	Dinâmicas com App Inventor no Apoio ao Aprendizado e no ensino aprendizagem de programação
		2	Ensino de Programação para Crianças através de Práticas Colaborativas nas Escolas
		3	Experiência no Uso de Ferramentas Online Gamificadas na Introdução à Programação de Computadores
		4	Saberes D'Avó Uma abordagem para o ensino de programação no Ensino Médio
		5	Um minicurso online de Algoritmos como apoio às disciplinas iniciais da graduação preparação, execução e resultados sobre a satisfação dos alunos
		6	Uma Experiência do Uso Do Hardware Livre Arduino no ensino de programação de computadores
		7	Uma Visão do Ensino de Computação nos Cursos Técnicos Integrados ao Ensino Médio em campi do Instituto Federal do Rio de Janeiro Uso de Programação no Apoio ao Apre
		8	Uso do Scratch no ensino de programação em Ponta Porã das séries inicias ao ensino superior
XXVII Simpósio Brasileiro de Informática na Educação			
SBIE	146	1	Ambiente para Aprendizagem de Programação fundamentado em Arquiteturas Pedagógicas
		2	Classificação de Códigos C usando medidas de similaridade para apoio ao Ensino em Programação
		3	Ensino de Lógica de Programação baseado na indução-dedução através de exemplo
		4	Juiz Online no ensino de Programação Introdutória Uma revisão sistemática da literatura
		5	Lord of Code Uma Ferramenta de Apoio ao Ensino da Programação
		6	Método de Ensino de Programação Mediada por Simulação Um Estudo de Caso no Curso Técnico Integrado em Informática
		7	O Desafio da Serpente - Usando gamification para motivar alunos em uma disciplina introdutória de programação
		8	O Ensino de Programação com Scratch e seu Impacto na Opção Profissional para Meninas
		9	Pensamento Computacional no Ensino de Programação Uma Revisão Sistemática da Literatura Brasileira

		10	Reconhecimento Automático de Representações de Rúbricas em Agrupamentos de Soluções de Exercícios de Programação
		11	Um Mapeamento Sistemático para auxiliar na escolha de plataformas EAD para o ensino-aprendizagem de Algoritmos e Programação de Computadores
		12	Um modelo lúdico para o ensino de conceitos de programação de computadores
		13	Uma solução livre e de baixo custo para prática e aprendizagem de programação e robótica
		14	Uso de Jogos em Cursos Introdutórios de Programação - Uma Revisão Sistemática
		15	Utilização de Problemas da Maratona de Competição de Programação e Juízes Eletrônicos como Estratégia de Ensino em um Curso de Graduação em Engenharia de Software

Workshops do V Congresso Brasileiro de Informática na Educação			
CBIE	173	1	Um modelo para promover o engajamento estudantil no aprendizado de programação utilizando gamification
		2	Despertando para a Programação com a Criação de Jogos
		3	Ensino de Programação em Robótica Móvel no Ensino Fundamental e Médio
		4	Mediação do erro no ensino de programação de computadores: fundamentos e aplicação da ferramenta FARMA-ALG
		5	Experiências no Uso da Metodologia Coding Dojo nas Disciplinas Básicas de Programação de Computadores em um Curso Interdisciplinar do Ensino Superior
		6	Um Estudo de Caso Sobre Competências do Pensamento Computacional Desenvolvidas na Programação em Blocos no Code.Org
		7	Uma Proposta de Oficina de Desenvolvimento de Jogos Digitais para Ensino de Programação
		8	Perfis de jogadores em contextos de ensino/aprendizagem em disciplinas de programação
		9	Modelagem Genérica de Aprendizes com Ênfase em Erros na Aquisição de Habilidades em Programação de Computadores
		10	NextStep: Um Protótipo para o Sequenciamento Inteligente e Adaptativo de Enunciados em Programação de Computadores
		11	Hello World: relato de experiência de um curso de iniciação à programação

XXIV Workshop sobre Educação em computação			
WEI	49	1	Desafios e oportunidades aos processos aos processos de ensino e de aprendizagem de programação para iniciantes
		2	Ensino de Linguagem de Programação com Ênfase na Aprendizagem Significativa
		3	Ensino de Programação para Futuros Não-Programadores Contextualizando os Exercícios com as Demais Disciplinas de mesmo Período Letivo
		4	Estímulo à prática multidisciplinar no ensino de Computação e Design através de um evento de programação focado em problema
		5	Hall of Fame_Shame um Padrao Pedagógico para o Ensino de Programação

		6	Information organization via computational thinking case study in a primary school classroom
		7	O uso de Robótica para aprendizado de Programação integrando alunos de Educação Básica e Ensino Superior
		8	PBL e robótica no ensino de conceitos de Lógica de Programação
		9	Plataforma Arduino como apoio ao ensino de programação no curso de Técnico em Informática integrado
		10	POOGame Um Jogo Serio para o Ensino de Programação Orientada a Objetos
		11	PortuCol uma pseudolinguagem inspirada em C ANSI para o Ensino de Lógica de Programação e Algoritmos
		12	Sistema de Apoio à Avaliação de Atividades de Programação por Reconhecimento Automático de Modelos de Soluções
		13	Torneios Baseados em Robocode para Incentivar Jovens a Aprender Programação

V Workshop de Desafios da Computação Aplicada à Educação			
DesafIE	13	1	As Tecnologias de Análise de Aprendizagem e os Desafios de prever desenpenhos de estudantes de programação

REVISTA BRASILEIRA DE INFORMÁTICA NA EDUCAÇÃO			
RBIE	12	1	Problemas e Dificuldades no Ensino de Programação: Um Mapeamento Sistemático
		2	Influência dos enunciados na resolução de problemas de programação introdutória

Jornada de Atualização em Informática na Educação			
JAI	3	1	Jogos no Design de Experiências de Aprendizagem de Programação Engajadoras

TOTAL	499	51	**Publicações em ensino de programação**

APÊNDICE H - ARTIGOS ENSINO-APRENDIZAGEM DE PROGRAMAÇÃO NO ES 2014

EVENTO	QTD		TÍTULO TRABALHOS
XX Workshop de Informática na Escola			
WIE	71	1	Produção e Avaliação de Videoaulas: Um Estudo de Caso no Ensino de Programação
		2	Um Estudo Observacional sobre a Disciplina Introdutória de Programação
XXV Simpósio Brasileiro de Informática na Educação			
SBIE	152	1	A Importância dos Programas de Extensão no Ensino e Prática de Programação e Desenvolvimento de Protótipos
		2	Influência da linguagem no ensino introdutório de programação
		3	KLouro: Um jogo educacional para motivar alunos iniciantes em programação
		4	Prog1Box: Um Ambiente Linux sem Distratores para o Aprendizado de Programação
		5	Um Ambiente Virtual com Feedback Personalizado para Apoio a Disciplinas de Programação
		6	Um Arcabouço para Construção de Mecanismos de Análise de Códigos de Programação Introdutória
Workshops do Congresso Brasileiro de Informática na Educação			
CBIE	91	1	Um Ambiente Virtual com Feedback Personalizado para Apoio a Disciplinas de Programação
XXII Workshop sobre Educação em Computação			
WEI			Sem dados
IV Workshop de Desafios da Computação Aplicada à Educação			
DesafIE			Sem dados
REVISTA BRASILEIRA DE INFORMÁTICA NA EDUCAÇÃO			
RBIE	10	1	Integração de uma Metodologia de Ensino Presencial de Programação com um Sistema Tutor Inteligente

Jornada de Atualização em Informática na Educação			
JAIE	4	0	

TOTAL	328	10	**Publicações em ensino de programação**

APÊNDICE I – ARTIGOS ENSINO-APRENDIZAGEM DE PROGRAMAÇÃO NO ES 2015

EVENTO	QTD		TÍTULO TRABALHOS
XXI Workshop de Informática na Escola			
WIE	67	0	
XXVI Simpósio Brasileiro de Informática na Educação			
SBIE	139	1	A Comparação da Realidade Mundial do Ensino de Programação para Iniciantes com a Realidade Nacional Revisão sistemática da literatura em eventos brasileiros
		2	A Teoria do Flow na contribuição do engajamento estudantil para apoiar a escolha de jogos no ensino de programação
		3	Caracterizando a pesquisa sobre autoavaliação na aprendizagem de programação para iniciantes
		4	Produção de Videoaulas de Programação em Java Acessíveis no Contexto de um Projeto de Capacitação Profissional para Pessoas Surdas
Workshops do IV Congresso Brasileiro de Informática na Educação			
CBIE	170	1	Algo+ - Um app para o auxílio na apredizagem de programação
		2	Ensinado Programação apoiada por um ambiente virtual e exercícios associados a cotidiano dos alunos compartilhando alternativas e lições aprendidas
		3	Interdisciplinaridade, programação visual e robótica educacional relato de experiência sobre o ensino inicial de programação
		4	Um estudo sobre erros em programação Reconhecendo as dificuldades de programadores iniciantes
		5	XP & Skills gamificando o processo de ensino de introdução a programação
XXI Workshop de Informática na Escola			
WEI	31	1	A Importância do Fator Motivacional no Processo Ensino-aprendizagem de programação
		2	Resgatando a Linguagem de Programacao Logo _ Uma experiência com calouros do ensino superior
		3	Testando a Diversão em um Jogo Sério para o Aprendizado introdutório de programação

		4	Uma ferramenta gamificada de apoio à disciplina introdutória de programação
		5	URI Online Judge Academic Integracao e Consolidacao da Ferramenta no Processo de EnsinoAprendizagem

IV Workshop de Desafios da Computação Aplicada à Educação			
DesafIE	15	0	

REVISTA BRASILEIRA DE INFORMÁTICA NA EDUCAÇÃO			
RBIE	15	1	Ensino-aprendizagem de programação: uma revisão sistemática da literatura

Jornada de Atualização em Informática na Educação			
JAIE	5	1	Sobre cursos introdutórios de programação na modalidade MOOC utilizando Moodle

TOTAL	442	16	**Publicações em ensino de programação**

APÊNDICE J - ARTIGOS ENSINO-APRENDIZAGEM DE PROGRAMAÇÃO NO ES 2016

EVENTO	QTD		TÍTULO TRABALHOS
XXII Workshop de Informática na Escola			
WIE	103	1	Experiência no Uso de Ferramentas Online Gamificadas na Introdução à Programação de Computadores
		2	Um minicurso online de Algoritmos como apoio às disciplinas iniciais da graduação preparação, execução e resultados sobre a satisfação dos alunos
		3	Uso do Scratch no ensino de programação em Ponta Porã das séries inicias ao ensino superior
XXVII Simpósio Brasileiro de Informática na Educação			
SBIE	146	1	Ambiente para Aprendizagem de Programação fundamentado em Arquiteturas Pedagógicas
		2	Classificação de Códigos C usando medidas de similaridade para apoio ao Ensino em Programação
		3	O Desafio da Serpente - Usando gamification para motivar alunos em uma disciplina introdutória de programação
		4	Reconhecimento Automático de Representações de Rúbricas em Agrupamentos de Soluções de Exercícios de Programação
		5	Uso de Jogos em Cursos Introdutórios de Programação - Uma Revisão Sistemática
		6	Utilização de Problemas da Maratona de Competição de Programação e Juízes Eletrônicos como Estratégia de Ensino em um Curso de Graduação em Engenharia de Software
Workshops do V Congresso Brasileiro de Informática na Educação			
CBIE	173	1	Um modelo para promover o engajamento estudantil no aprendizado de programação utilizando gamification
		2	Despertando para a Programação com a Criação de Jogos
		3	Mediação do erro no ensino de programação de computadores: fundamentos e aplicação da ferramenta FARMA-ALG
		4	Experiências no Uso da Metodologia Coding Dojo nas Disciplinas Básicas de Programação de Computadores em um Curso Interdisciplinar do Ensino Superior
		5	Perfis de jogadores em contextos de ensino/aprendizagem em disciplinas de programação

		6	Modelagem Genérica de Aprendizes com Ênfase em Erros na Aquisição de Habilidades em Programação de Computadores

XXII Workshop de Informática na Escola			
WEI	103	1	Desafios e oportunidades aos processos aos processos de ensino e de aprendizagem de programação para iniciantes
		2	Ensino de Programação para Futuros Não-Programadores Contextualizando os Exercícios com as Demais Disciplinas de mesmo Período Letivo
		3	Estímulo à prática multidisciplinar no ensino de Computação e Design através de um evento de programação focado em problema
		4	Hall of Fame_Shame um Padrao Pedagógico para o Ensino de Programação
		5	O uso de Robótica para aprendizado de Programação integrando alunos de Educação Básica e Ensino Superior
		6	POOGame Um Jogo Serio para o Ensino de Programação Orientada a Objetos

V Workshop de Desafios da Computação Aplicada à Educação			
DesafIE	13	0	

REVISTA BRASILEIRA DE INFORMÁTICA NA EDUCAÇÃO			
RBIE	12	1	Problemas e Dificuldades no Ensino de Programação: Um Mapeamento Sistemático
		2	Influência dos enunciados na resolução de problemas de programação introdutória

Jornada de Atualização em Informática na Educação			
JAIE	3	1	Jogos no Design de Experiências de Aprendizagem de Programação Engajadoras

TOTAL	553	24	**Publicações em ensino de programação**

APÊNDICE K - ARTIGOS CORRELATOS (2014)

TÍTULO DO TRABALHO			
Linguagem de programação utilizada	**Objetivo do trabalho**	**Aplicada em uma disciplina de programação**	**Evento**
Produção e Avaliação de Videoaulas: Um Estudo de Caso no Ensino de Programação			**XX Workshop de Informática na Escola**
Portugol	Verificação importancia ou não da utilização de video aulas	Sim	
Um Estudo Observacional sobre a Disciplina Introdutória de Programação			
Portugol	Verificação das dificuldades de aprendizado	Sim	
A Importância dos Programas de Extensão no Ensino e Prática de Programação e Desenvolvimento de Protótipos			**XXV Simpósio Brasileiro de Informática na Educação**
C	Projeto de extensão, prototipagem com arduino em C	Não	
Influência da linguagem no ensino introdutório de programação			
Phyton e C	Verificação da influência da linguagem adotada nas disciplinas introdutórias de programação	Não	
KLouro: Um jogo educacional para motivar alunos iniciantes em programação			
Phyton	Klouro (OA), possui o objetivo de ajudar os alunos a praticarem de forma divertida alguns conteúdos introdutórios de programação	Não	
Prog1Box: Um Ambiente Linux sem Distratores para o Aprendizado de Programação			
Phyton	ProgBox, acesso online restrito e controlado pelo professor de forma a evitar distratores ou possibilidades de fraude	Sim	
Um Ambiente Virtual com Feedback Personalizado para Apoio a Disciplinas de Programação			

Não especifica	Feeper, ferramenta web para apoiar a aprendizagem de programação em sala de aula, assim como em extraclasse	Sim	
Um Arcabouço para Construção de Mecanismos de Análise de Códigos de Programação Introdutória			
Phyton	Arcabouço para construção de mecanismos de análise de soluções de programação com foco no ensino introdutório	Sim	

Um Ambiente Virtual com Feedback Personalizado para Apoio a Disciplinas de Programação			**Congresso Brasileiro de Informática na Educação**
Java	Ambiente virtual composto de recursos como feedback personalizado, facilidades para a interação e auxílio ao professor para acompanhamento dos seus alunos (feeper)	Sim	

Integração de uma Metodologia de Ensino Presencial de Programação com um Sistema Tutor Inteligente			**Revista Brasileira de Informática na Educação**
Lógica	Halyen, ferramenta tecnológica de auxílio ao ensino presencial, através da escolha dinâmica da estratégia pedagógica, segundo o perfil, estado emocional e outras características de cada aluno	Sim	

APÊNDICE L - ARTIGOS CORRELATOS (2015)

TÍTULO DO TRABALHO			
Linguagem de programação utilizada	**Objetivo do trabalho**	**Aplicada em uma disciplina de programação**	**Evento**
A Comparação da Realidade Mundial do Ensino de Programação para Iniciantes com a Realidade Nacional Revisão sistemática da literatura em eventos brasileiros			**XXVI Simpósio Brasileiro de Informática na Educação**
Não específica	Revisão sistemática da literatura sobre o ensino de programação e algoritmos para alunos iniciantes do ensino superior brasileiro, chamados CS1 (Computer Science 1)	Não	
A Teoria do Flow na contribuição do engajamento estudantil para apoiar a escolha de jogos no ensino de programação			
Não específica	Propor diretrizes para auxiliar o professor na escolha adequada de jogos para o ensino de programação baseada em técnicas de engajamento, perfis de jogadores e características do estado de Flow, com o intuito de auxiliar o aumento da motivação dos alunos.	Não	
Caracterizando a pesquisa sobre autoavaliação na aprendizagem de programação para iniciantes			
Não específica	Mapeamento sistemático da literatura que visou identificar como se caracteriza a pesquisa em auto avaliação da aprendizagem de programação para iniciantes.	Não	
Produção de Videoaulas de Programação em Java Acessíveis no Contexto de um Projeto de Capacitação Profissional para Pessoas Surdas			
Java	Oferta de capacitações em Java acessíveis para pessoas surdas no contexto de EAD	Não	
Algo+ - Um app para o auxílio na apredizagem de programação			**IV Congresso Brasileiro de Informática na Educação**
Java	App Algo+, tem como objetivo facilitar o ensino dos conceitos básicos de programação de computadores	Não	
Ensinado Programação apoiada por um ambiente virtual e exercícios associados a cotidiano dos alunos compartilhando alternativas e lições aprendidas			

Não especifica	Criar alternativas para ampliar o espaço de encontros presenciais com a criação de uma sala de aula virtual e uso de enunciados de exercícios que nos permitissem aproximar dos interesses dos alunos por meio de situações ancoradas nos seus hábitos de lazer	Sim	
Interdisciplinaridade, programação visual e robótica educacional relato de experiência sobre o ensino inicial de programação			
Em bloco	Potencializar o entendimento e a aprendizagem de conceitos básicos da área de algoritmos e programação pelos alunos ingressantes de um curso de Bacharelado em Sistemas de Informação (BSI).	Sim	
Um estudo sobre erros em programação Reconhecendo as dificuldades de programadores iniciantes			
C	Avaliar e elencar os erros mais comuns cometidos por estes estudantes durante as práticas de programação	Não	
XP & Skills gamificando o processo de ensino de introdução a programação			
Não especifica	Explorar o uso da gamificação no tratamento do problema da motivação e participação de alunos cursando disciplinas introdutórias de programação	Sim	

A Importância do Fator Motivacional no Processo Ensino-aprendizagem de programação			**XXIII Workshop sobre Educação em Computação**
Phyton e C	Usar plataforma Arduino e programação visual para ensino de programação como fator motivacional	Sim	
Resgatando a Linguagem de Programacao Logo _ Uma experiência com calouros do ensino superior			
Logo	Proposta de ensino de programação de computadores, utilizando a linguagem de programação Logo como ferramenta de ensino.	Não	
Testando a Diversão em um Jogo Sério para o Aprendizado introdutório de programação			
Em bloco	Identificar os pontos fracos do jogo em promover a diversão.	Não	
Uma ferramenta gamificada de apoio à disciplina introdutória de programação			
C	Desenvolver e aplicar uma ferramenta de auxílio à realização de atividades de programação, nomeada Kodesh (Koding Shell).	Sim	
URI Online Judge Academic Integracao e Consolidacao da Ferramenta no Processo de Ensino Aprendizagem			

C	Contribuir efetivamente com os esforços de professores e estudantes do portal URI Online Judge que contém, além de inúmeras facilidades para a prática de programação, a possibilidade de acompanhamento das atividades pelo docente.	Não	

Ensino-aprendizagem de programação: uma revisão sistemática da literatura			**Revista Brasileira de Informática na Educação**
Não especifica	Apresentar uma revisão sistemática da literatura referente às abordagens para o ensino-aprendizagem de programação, publicados nos últimos cinco anos (2009 a 2013), nos quatro mais importantes eventos nacionais da área, o Simpósio Brasileiro de Informática na Educação, o Workshop de Informática na Escola, Workshop de Educação em Computação, e o Simpósio Brasileiro de Jogos e Entretenimento Digital, além das duas mais relevantes revistas nacionais na área, a Revista Brasileira de Informática na Educação e a Revista Novas Tecnologias na Educação.	Não	

Sobre cursos introdutórios de programação na modalidade MOOC utilizando Moodle			**Jornada de Atualização em Informática na Educação**
Não especifica	Discutir dois tópicos relevantes à computação, como introduzir conceitos de programação e como gerenciar um curso Web aberto e massivo (MOOC).	Não	

APÊNDICE M - ARTIGOS CORRELATOS (2016)

TÍTULO DO TRABALHO			
Linguagem de programação utilizada	**Objetivo do trabalho**	**Aplicada em uma disciplina de programação**	**Evento**
Experiência no Uso de Ferramentas Online Gamificadas na Introdução à Programação de Computadores			**XXII Workshop de Informática na Escola**
Não especifica	Aplicação de 3 ferramentas online	Sim	
Um minicurso online de Algoritmos como apoio às disciplinas iniciais da graduação preparação, execução e resultados sobre a satisfação dos alunos			
Portugol	Curso de suporte a disciplina de programação	Mini curso	
Uso do Scratch no ensino de programação em Ponta Porã das séries inicias ao ensino superior			
Scratch	Oferta de cursos para alunos que não são de cursos de info	Curso	
Ambiente para Aprendizagem de Programação fundamentado em Arquiteturas Pedagógicas			**XXVII Simpósio Brasileiro de Informática na Educação**
Haskell	Abordagem que permite melhorar a aprendizagem de programação em cursos introdutórios	Sim	
Classificação de Códigos C usando medidas de similaridade para apoio ao Ensino em Programação			
C	Auxiliar professores na geração de uma base, propondo uma melhoria em uma abordagem de classificação de códigos em linguagem C baseada em medidas de similaridade para agrupar códigos de programação por tema ou tipo de problema.	Não	
O Desafio da Serpente - Usando gamification para motivar alunos em uma disciplina introdutória de programação			

Phyton	Relato uma experiência em andamento em que conceitos de gamification foram usados para propor desafios diários a um grupo de alunos ao longo de oito semanas, obtendo bons índices de comprometimento com os estudos e a aceitação do público-alvo.	Sim	
Reconhecimento Automático de Representações de Rúbricas em Agrupamentos de Soluções de Exercícios de Programação			
C	Apoiar o processo de avaliação de programação, este trabalho propõe uma estratégia baseada em técnicas de clustering e de Análise de Componentes Principais (PCA) para reconhecer, a partir de soluções desenvolvidas por alunos, exemplos de soluções que representem, em um esquema de rúbricas, os escores atribuídos por um professor.	Não	
Uso de Jogos em Cursos Introdutórios de Programação - Uma Revisão Sistemática			
Não especifica	Realizar uma revisão sistemática da literatura acerca do uso de jogos digitais no ensino introdutório de programação em cursos ligados a área de Computação.	Não	
Utilização de Problemas da Maratona de Competição de Programação e Juízes Eletrônicos como Estratégia de Ensino em um Curso de Graduação em Engenharia de Software			
Não especifica	Analisou a estratégia de ensino alicerçada nas maratonas de programação e nos juízes eletrônicos e a influência desta estratégia no desempenho dos alunos nas disciplinas do curso de graduação em Engenharia de Software da Universidade de Brasília – UnB.	Sim	

Um modelo para promover o engajamento estudantil no aprendizado de programação utilizando gamification			**V Congresso Brasileiro de Informática na Educação**
Programação Orientada a Objetos	Investigar a influência positiva de gamification no engajamento e desempenho dos estudantes no aprendizado de programação. Cod[edu] contempla a taxonomia de Bloom	Sim, mas não com turma de primeiro semestre	
Despertando para a Programação com a Criação de Jogos			

Scratch	Desmitificar que aprender programação é difícil. Despertar o interesse e atrair os alunos, a metodologia utilizada foi desenvolver o projeto em torno da temática de criação de jogos.	Não
Mediação do erro no ensino de programação de computadores: fundamentos e aplicação da ferramenta FARMA-ALG		
Pascal	Apresenta dois momentos: o primeiro visa avançaar na discussão, ainda nova, do papel da mediação do erro no ensino e aprendizagem de programação de computadores, a partir dos preceitos encontrados nos estudos de Vigotski e da Psicologia Histórico-Cultural. E, o segundo, avalia os impactos da mediaçãao do erro através do uso da ferramenta FARMA-ALG.	Sim
Experiências no Uso da Metodologia Coding Dojo nas Disciplinas Básicas de Programação de Computadores em um Curso Interdisciplinar do Ensino Superior		
C, C++, Processing	Investigar a adoção da metodologia Coding Dojo em um curso de natureza interdisciplinar, onde nem todos os participantessão totalmente da área das ciências, engenharias e computação.	Sim
Perfis de jogadores em contextos de ensino/aprendizagem em disciplinas de programação		
Não especifica	Verificar se perfis de jogadores podem ser utilizados como meio para agrupar usuários/alunos em contextos educacionais.	Não
Modelagem Genérica de Aprendizes com Ênfase em Erros na Aquisição de Habilidades em Programação de Computadores		
C++	Abordagem para avaliação e acompanhamento da aquisição de perícias no domínio da programação de computadores	Não

Desafios e oportunidades aos processos aos processos de ensino e de aprendizagem de programação para iniciantes		
Scratch	Apresenta a proposta de uma abordagem para minimizar alguns dos problemas vivenciados pelos estudantes.	Não

Ensino de Programação para Futuros Não-Programadores Contextualizando os Exercícios com as Demais Disciplinas de mesmo Período Letivo		
Phyton	Relato da metodologia de ensino-aprendizagem concebida, em grande parte baseada nos trabalhos de Píccolo *et al.*. (2010) e Zanini e Raabe (2012), bem como os resultados	Sim
Estímulo à prática multidisciplinar no ensino de Computação e Design através de um evento de programação focado em problema		
Não especifica	Descreve a concepção e execução do hackathon Code Arena no intuito de motivar a organização de práticas multidisciplinares como estímulo ao ensino e aprendizado em Computação e Design.	Não
Hall of Fame_Shame um Padrao Pedagógico para o Ensino de Programação		
C	Padrão pedagógico voltado ao ensino de programação, em cursos de nível superior na área de Computação. Este padrão é centrado na exposição e discussão de bons e maus exemplos de códigos produzidos pelos alunos, formando o que denominou-se respectivamente de “Hall of Fame” e “Hall of Shame”.	Sim
O uso de Robótica para aprendizado de Programação integrando alunos de Educação Básica e Ensino Superior		
Blocos	Interação entre alunos com diferentes níveis de formação para programação de robôs a fim permitir um processo de aprendizado por meio de discussões e colaborações na realização de tarefas, utilizando ferramentas tecnológicas.	Não
POOGame Um Jogo Serio para o Ensino de Programação Orientada a Objetos		
Java	Apresenta a elaboração de um jogo sério para o ensino de programação orientada a objetos (POO), denominado POOGame. O desenvolvimento do jogo foi baseado em jogos de RPG, de forma que o jogador controla um personagem e a partir de batalhas é possível invocar criaturas e controlar suas ações utilizando comandos baseado na linguagem JAVA.	Não

Problemas e Dificuldades no Ensino de Programação: Um Mapeamento Sistemático			**Revista Brasileira de Informática na Educação**
Não especifica	Apresentar os resultados de uma revisão da literatura conduzida para coletar e avaliar evidências em problemas e dificuldades no ensino e aprendizagem de programação.	Não	
Influência dos enunciados na resolução de problemas de programação introdutória			
Não especifica	Verifica se a forma no qual o enunciado está escrito influencia ou não na resolução das atividades.	Não	

Jogos no Design de Experiências de Aprendizagem de Programação Engajadoras			**Jornada de Atualização em Informática na Educação**
Não especifica	Indicar direcionamentos e discutir sobre os desafios pertinentes a integração dos jogos e de abordagens baseadas em jogos no design de experiências introdutórias de programação mais engajadoras.	Não	

APÊNDICE N – ATIVIDADES CONSPROG

Atividades[31] de linguagem de programação C ANSI fundamentadas na Taxonomia dos Objetivos Educacionais de Domínio Cognitivo

1 VARIÁVEIS

8.1 CONHECIMENTO

1) Variáveis podem ser entendidas como
   a. espaço na memória ROM, utilizada para armazenar informações.
   b. espaço na memória RAM, utilizada para armazenar informações.
   c. espaço na memória RAM utilizada para excluir informações.
   d. espaço na memória ROM utilizada para excluir informações.

2) Na linguagem de programação C ANSI identificadores são:
   a. nomes dados à variáveis, e devem respeitar algumas regras.
   b. nomes dados à variáveis e não possui regras.
   c. informações armazenadas do usuário e devem respeitar algumas regras.
   d. informações armazenadas do usuário e não possui regras.

3) Para declarar uma variável devemos:
   a. definir um identificador e mostrar este ao usuário.
   b. definir o tipo e pedir ao usuário o identificador.
   c. definir um identificador e o tipo de dado que poderá armazenar.
   d. definir um identificador e pedir o tipo de dado para o usuário.

4) Na linguagem C, letras maiúsculas e minúsculas em nome de variável são tratadas _____, logo a variável nome é _____ a variável Nome.
   a. indiferentemente / igual
   b. diferentemente / igual
   c. indiferentemente / diferente
   d. diferentemente / diferente

5) As variáveis em um programa na linguagem C devem ser declaradas:
   a. antes de ser utilizada
   b. depois de ser utilizada
   c. não precisa declarar
   d. não faz diferença onde é declarada a variável

6) Uma declaração de variável em C deve conter primeiramente o _____, depois o _____ e por último um ; (ponto-e-vírgula).

[31] As atividades aqui apresentadas foram criadas pela a autora ou adaptadas dos seguintes autores:

- CARVALHO, Victorio Albani de Lógica de programação: Curso Técnico em Informática / Victorio Albani de Carvalho. – Colatina: CEAD / Ifes, 2010.
- DEITEL, Paul; DEITEL, Harvey. C: como programar. 6ª ed. Pearson Education – Br, 2011.
- OKUYAMA, Fabio Yoshimitsu ; MILETTO, Evandro Manara ; NICOLAO, Mariano (Org.). Desenvolvimento de software I: conceitos básicos. Porto Alegre, RS: Bookman, 2014. 223 p. (Tekne) ISBN 9788582601457

a. nome / tipo
b. tipo / nome
c. tipo / valor
d. nome / valor

7) Uma variável que necessite armazenar valores inteiros para efetuar cálculo é do tipo:
a. double
b. float
c. int
d. char

8) Um programa pede para armazenar o sexo de um cliente, para isso utilizamos o tipo de variável:
a. char
b. int
c. double
d. float

9) Para calcular a média de um aluno, precisamos armazená-la na variável mediaAluno, que deve ser do tipo:
a. int
b. void
c. float
d. char

10) O tipo de variável informa _____ uma informação deve ser armazenada.
a. como
b. onde
c. porque
d. se

11) São tipos de variáveis em Linguagem C ANSI:
a. dobro, inteiro, nome
b. int, float, char
c. reais, inteiro, caracteres
d. int, reais, char

8.2 COMPREENSÃO

1) Casa de câmbio é o local no qual as pessoas vão para trocar moedas, como por exemplo: o cliente tem R$100,00 (Cem reais) e deseja trocar esse valor por dólar, sabendo que o dólar está valendo R$ 4,06 (Quatro reais e seis centavos), o cliente receberá U$ 24,63 (Vinte e quatro dólares e sessenta e três centavos de dólar). Sabendo disso, diga, em valor numérico, quantas variáveis são necessárias para resolver esse problema?
a. 4
b. 2
c. 1
d. 5

2) Uma empresa X necessita de um programa para fazer o cálculo da comissão de seus vendedores, para isso necessita saber o valor total da venda e também qual a porcentagem que o vendedor ganha de comissão sob a venda. Quantas variáveis são necessárias para esse programa?
a. 3
b. 2
c. 4
d. 1

3) Pafúncio comprou um carro novo, encheu o tanque de combustível e zerou a quilometragem, para que na próxima vez que encher o tanque consiga fazer a média de consumo. Após rodar 510 quilometros, Pafúncio precisou abastecer seu carro novamente, ao completar o tanque, marcou 38 litros de combustível, gastando R$144,02. Quantas variáveis são necessárias para calcular o consumo médio de combustível do carro do Pafúncio?
   a. 3
   b. 2
   c. 4
   d. 1

4) Supondo que as variáveis codAluno, sexo e nota sejam utilizadas para armazenar o código do aluno, sexo do aluno, e a nota do aluno. Essas variáveis seriam de que tipos respectivamente?
   a. float, char, int
   b. int, char, float
   c. int, int, int
   d. float, float, float

5) Um caixa eletrônico possui como variáveis valor e saldo, o usuário pode sacar, depositar e verificar saldo. Logo, ao declará-las devemos informar que são do tipo:
   a. int
   b. char
   c. float
   d. boolean

6) Para calcular a área de um retângulo é necessário conhecer a altura e a largura do mesmo. O resultado do cálculo é armazenado numa variável AreaRetangulo (AreaRetangulo = altura * largura). Qual tipo de dado deve ser declarado para cada uma dessas variáveis?
   a. int, float, float
   b. int, int, int
   c. float, float, int
   d. float, float, float

7) Sabendo que a fórmula de bhaskara é: $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$, diga quais são os tipos dos dados para cada uma das variáveis:
   a. int b, int a, int c, float x
   b. float b, float a, float c, float x1, float x2
   c. int b, int a, int c, int x1, int x2
   d. float b, float a, float c, float x

8) Complete o código abaixo com o tipo de variável correta:

```
#include <stdio.h>
int main(){
        _____ x;
        printf("Digite sua idade: ");
        scanf("%d",&x);
        printf("Você tem: %d anos", x);
}
```

   a. float
   b. char
   c. int
   d. double

9) Complete o código com o tipo de variável correta:

```
#include <stdio.h>
int main(){
        _____ codFunc;
```

```
        _____ sexo;
        _____ salario;
        printf("Digite seu código de funcionário: ");
        scanf("%s",&codFunc);
        printf("Digite o valor de seu sexo: ");
        scanf("%f",&sexo);
        printf("Digite o valor de seu salário: ");
        scanf("%f",&salario);
        printf("%s seu salário é de %f", nome, salario);
    }
```

a. float, float, float
b. char, float, int
c. int, char, float
d. int, int, int

10) Complete o código com o tipo de variável correta:

```
    #include <stdio.h>
    int main(){
        _____ nota1;
        _____ nota2;
        _____ media;
        printf("Digite sua nota: ");
        scanf("%f",&nota1);
        printf("Digite sua nota: ");
        scanf("%f",&nota2);
        media = (nota1 + nota2)/2;
        printf("Sua média é %f", media);
    }
```

a. float, float, float
b. char, char, float
c. int, int, int
d. float, float, char

8.3 APLICAÇÃO

1) Identificadores são os nomes das variáveis, sabendo disso, marque V (verdadeiro) ou F (falso) de acordo com as regras básicas para formação dos mesmos:

( ) Podem ter qualquer tamanho.
( ) Pode ser utilizado caracteres especiais.
( ) O primeiro caractere deve ser sempre uma letra ou sublinhado.
( ) Não são permitidos espaços em branco.

Marque a resposta que corresponde a sequência correta de cima para baixo:

a. V – V – V – V
b. V – F – V – V
c. F – V – F – V
d. V – V – F – V

2) Para criação de identificador devem-se respeitar algumas regras, escolha a resposta que apresentam identificadores válidos segundo a Linguagem de Programação C ANSI:

a. 2valores, número, base
b. v@lor, numero, nota
c. nota1, _base, altura
d. nota1, base, int

3) Ao criar um identificador para uma variável deve-se cuidar para não utilizar palavras reservadas, logo, assinale a alternativa que pode ser considerada um identificador válido:
   a. inteiro
   b. float
   c. char
   d. int

4) Perímetro é uma linha que forma o contorno de uma figura num plano. Para se descobrir o valor do perímetro de um círculo, é necessário conhecer o seu raio. Logo, a fórmula para efetuar esse calculo é: $p = 2\pi r$. Sabendo disso, assinale a alternativa que apresenta a declaração das variáveis com o tipo de dado adequado a cada uma delas.
   a. int p; int r;
   b. p float; r float;
   c. p int; r int;
   d. float p; float r;

5) Um programa que calcula do dobro pede ao usuário um valor inteiro, esta variável é denominada val1. Declare de forma correta essa variável:
6) Luisa é estudante de educação física e precisa de um sistema que calcule o índice de massa corpórea. Sabendo que a fórmula é: imc = peso / (altura * altura), crie as variáveis necessárias para esse programa, separando-as por ;. Ex.: int val1; int val2;
   ➔ Lembre-se que os nomes das variáveis devem ter sentido para o programa.

7) Pool é uma empresa de piscinas e é a pioneira do ramo, para agilizar o processo das vendas George precisa de um programa que calcule o valor final da piscina. Esse cálculo é feito a partir do volume total da piscina multiplicado pelo valor de m³. Levando em conta que todas as piscinas da empresa são retangulares ou quadradas, o cálculo do volume é base * altura * comprimento. Crie as variáveis necessárias para esse programa, separando-as por ;. Ex.: int val1; int val2.
   ➔ Lembre-se que os nomes das variáveis devem ter sentido para o programa.

8.4 ANÁLISE

1) Dado o enunciado: Pedro tem uma locadora e quer que um programa ao ler o código de um DVD faça a soma dos valores das locações. Quais variáveis seriam necessárias?
   a. nomeDVD, genero, valor
   b. codDVD, valor, x, somaTotal
   c. codDVD, nomeDVD, valor, somaTotal, locacoes
   d. codDVD, valor, somaTotal

2) Sabendo que existem as variáveis: valor1, valor2, media; Qual seria o enunciado correspondente a estas variáveis?
   a. Faça um programa que leia do usuário três valores e calcule a media deles.
   b. Faça um programa que leia um valor e calcule a media do usuário.
   c. Faça um programa que calcule a média de dois valores digitados pelo usuário.
   d. Faça um programa que ao receber dois valores do usuário multiplique-os calculando a média dos valores.

3) Um programa pede duas informações para o usuário: base e altura, qual enunciado corresponderia a este programa?
   a. Desenvolva um programa que calcule a área de um retângulo.
   b. Desenvolva um programa que calcule o volume de um retângulo.
   c. Desenvolva um programa que peça para o usuário a base, a altura e a área de um retângulo.
   d. Desenvolva um programa que peça ao usuário três valores e calcule a área de um retângulo.

4) Escreva V para verdadeiro e F para falso nas declarações abaixo, avaliando se a variável corresponde ao tipo declarado: [2]

( ) int Endereco, NFilhos;
( ) char nomeAluno, email;
( ) double salario, Peso, sexo;

Marque a resposta correta na sequência de cima para baixo:

a. V, F, V
b. F, V, F
c. F, V, V
d. V, V, V

5) Dado o programa abaixo, verifique se o mesmo possui os tipos de variáveis adequados para o mesmo e selecione Verdadeiro caso o programa esteja correto e Falso se não estiver:

```
#include <stdio.h>
int main(){
int val1;
int val2;
int divisão;
printf("Digite o valor 1: ");
scanf("%d",&val1);
printf("Digite o valor 2: ");
scanf("%d",&val2);
divisao = val1/val2;
}
```

a. Verdadeiro
b. Falso

6) João quer descobrir o valor final de seu salário que é calculado através da multiplicação entre as horas trabalhadas e o valor da hora trabalhada. Assim sendo, assinale a alternativa que apresenta a resposta correta para este enunciado:

a.
```
#include <stdio.h>
int main(){
int hora;
int valHora;
float salTotal;
printf("Digite as horas trabalhadas: ");
scanf("%d",&hora);
printf("Digite o valor da hora trabalhada: ");
scanf("%d",&valHora);
salTotal = hora * valHora;
printf("Seu salário é %f", salTotal);
}
```

b.
```
#include <stdio.h>
int main(){
float hora;
float valHora;
float salTotal;
printf("Digite as horas trabalhadas: ");
scanf("%f",&hora);
printf("Digite o valor da hora trabalhada: ");
scanf("%f",&valHora);
salTotal = hora * valHora;
printf("Seu salário é %f", salTotal);
}
```

c. .
```
#include <stdio.h>
int main(){
printf("Digite as horas trabalhadas: ");
scanf("%f",&hora);
printf("Digite o valor da hora trabalhada: ");
```

```
        scanf("%f",&valHora);
        float hora;
        float valHora;
        float salTotal;
        salTotal = hora * valHora;
        printf("Seu salário é %f", salTotal);
}
d. #include <stdio.h>
int main(){
        char hora;
        char valHora;
        char salTotal;
        printf("Digite as horas trabalhadas: ");
        scanf("%c",&hora);
        printf("Digite o valor da hora trabalhada: ");
        scanf("%c",&valHora);
        salTotal = hora * valHora;
        printf("Seu salário é %c", salTotal);
}
```

8.5 SÍNTESE

1) Décimo terceiro salário é a gratificação de natal compulsória, instituída pela Lei 4090/62, e corresponde a um mês não trabalhado. O seu valor baseia-se na remuneração aferida em dezembro, salvo se o ganho do empregado é variável, quando, então, deve guardar equivalência com a média obtida durante o ano. Seu pagamento deve ocorrer no período de 1º a 20 de dezembro. O setor de Recursos Humanos deseja um programa capaz de calcular o valor a ser pago do 13º salario de um funcionário que não trabalhou todos os meses do ano. Com base no enunciado, escolha a afirmação que melhor identifica as variáveis a serem utilizadas no algoritmo:
   a. O programa deve considerar quantos meses o funcionário trabalhou ao longo do ano e qual o valor do salario desse funcionário. O resultado é obtido com base no valor total pago divido por 12 meses.
   b. O programa deve replicar o valor do salario do funcionário do mês de dezembro.
   c. O programa deve armazenar o valor do salário de cada mês do ano. Então se escolhe o maior valor para ser utilizado como 13º salario.
   d. O programa deve armazenar o valor do salário de cada mês do ano, bem como quantos meses foram trabalhados. Então o resultado é armazenado com base na soma dos salários divididos pelo número de meses trabalhados.

2) O valor de uma motocicleta direto da fábrica é calculado baseando-se no lucro da montadora, nos impostos e no custo básico de produção. Faça um programa que calcule o valor dos impostos, o valor do lucro dessa fabrica e o custo básico contido no preço cobrado pela fabrica. Com base no enunciado, escolha a afirmação que melhor corresponde a interpretação correta do programa a ser produzido.
   a. O programa deve conter uma variável para o preço da motocicleta, bem como o percentual de impostos, percentual de lucro e custo básico de produção. Ao final serão calculados os 3 valores (imposto, lucro, custo) multiplicando-se preço da motocicleta por cada percentual.
   b. O programa deve conter inicialmente o preço da motocicleta, o valor do imposto, o valor do lucro, o valor do custo. Sendo esses utilizados para calcular os percentuais solicitados como resultado.
   c. O programa deve conter inicialmente os percentuais do imposto, do lucro, do custo. Sendo esse utilizado para calcular o valor do preço de fabrica, do valor do imposto, o valor do lucro, o valor do custo.
   d. O programa deve conter inicialmente o preço da motocicleta. Sendo que os 3 valores (imposto, lucro, custo) são obtidos dividindo-se os preço por 3.

3) Uma padaria gostaria de um sistema capaz de calcular o preço a ser cobrado por produtos que são pesados na balança. Com base no enunciado acima, escolha a afirmação abaixo que melhor representa elementos de tipos de dados:
   a. O sistema deve possuir uma variável do tipo float para armazenar o peso do produto e outra variável do tipo float com o preço do produto. O resultado da multiplicação desses valores será armazenado numa variável do tipo float.
   b. O sistema deve possuir uma variável do tipo int para armazenar o peso do produto e outra variável do tipo int com o preço do produto. O resultado da multiplicação desses valores será armazenado numa variável do tipo float.
   c. O sistema deve possuir uma variável do tipo float para armazenar o peso do produto e outra variável do tipo int com o preço do produto. O resultado da multiplicação desses valores será armazenado numa variável do tipo int.
   d. O sistema deve possuir duas variáveis do tipo float para armazenar os pesos dos produtos e outras duas variáveis do tipo float com os preços dos produtos. O resultado da multiplicação desses valores será armazenado numa variável do tipo float.

4) O supermercado deseja criar um sistema de venda de produtos. Para demonstrar que você detem conhecimento da criação do sistema, você deverá fazer um pequeno programa capaz de vender até 3 produtos, ou seja, deverá ler a quantidade de cada produto e o preço desse. Ao final, o sistema apresentará o preço total a ser cobrado do cliente. Com base no enunciado acima, escolha a afirmação abaixo que melhor representa elementos de tipos de dados:
   a. O sistema deve possuir uma variável do tipo float para armazenar a quantidade do produto e outra variável do tipo int com o preço do produto. O resultado da multiplicação desses valores será armazenado numa variável do tipo int.
   b. O sistema deve possuir três variáveis do tipo float para armazenar a quantidade de cada produto e outras três variáveis do tipo float com o preço de cada produto. O resultado da multiplicação desses valores será armazenado numa variável do tipo int.
   c. O sistema deve possuir três variáveis do tipo float para armazenar a quantidade de cada produto e outra variável do tipo float com o preço do produto. O resultado da multiplicação desses valores será armazenado numa variável do tipo float.
   d. O sistema deve possuir três variáveis do tipo float para armazenar a quantidade de cada produto e outras três variáveis do tipo float com o preço de cada produto. O resultado da multiplicação desses valores será armazenado numa variável do tipo float

8.6 AVALIAÇÃO

1) Faça um programa capaz de calcular a media de 3 notas de um aluno. Com base no enunciado acima, escolha o programa com a melhor solução:

a.
```
#include <stdio.h>
    int main(){
        int abacaxi = 1;
        int mamao = 2;
        int abobrinha = 3;
        int suco1 = abacaxi + mamao + abobrinha;
        int sucoVerde = suco1 / 3;
    }
```

b.
```
#include <stdio.h>
int main(){
        int a = 1;
        int b = 2;
        int c = 3;
        int d = a + b + c;
        int e = d / 3;
    }
```

c.
```
#include <stdio.h>
    int main()    {
```

```
            int prova1 = 1;
            int prova2 = 2;
            int prova3 = 3;
            int soma = prova1 + prova2 + prova3;
            int media = soma / 3;
        }
```

d.
```
#include <stdio.h>
        int main() {
            int prova1 = 1;
            int prova2 = 2;
            int prova3 = 3;
            int media = soma / 3;
            int soma = prova1 + prova2 + prova3;
        }
```

2) Faça um programa capaz de calcular o troco a ser entregue pelo caixa de um supermercado. Com base no enunciado acima, escolha o programa com a melhor solução:

a.
```
#include <stdio.h>
        int main(){
            float valor = 65.32;
            float recebido = 70.00;
            float troco = recebido - valor;
        }
```

b.
```
#include <stdio.h>
        int main() {
            float recebido = 70.00;
            float troco = recebido;
            float valor = 65.32;
            troco = recebido - valor;
        }
```

c .
```
#include <stdio.h>
        int main(){
            float $$dinheiro = 70.00;
            float precoProdutosLimpeza = 65.32;
            float 1ºtroco = $$dinheiro - precoProdutosLimpeza;
        }
```

d.
```
#include <stdio.h>
        int main() {
            dinheiro = 70.00;
            preco = 65.32;
            troco = dinheiro - preco;
        }
```

3) Assinale os identificadores que estão com o tipo de dado adequado para cada variável:
   a. int idade
   b. int salario
   c. char sexo
   d. float diaDeNascimento
   e. char qtdDeFilhos
   f. float peso
   g. int comprimento
   h. float quilometragem

4) Assinale a alternativa que apresente um programa que realize a média de 3 valores de forma mais adequada:

a.
```
#include <stdio.h>
    int main(){
            float val1 = 4.5;
```

```
        float val2 = 7.3;
        int val3 = 5;
        float soma = val1 + val2 + val3;
        float media = soma / 3;
    }
```

b.
```
#include <stdio.h>
    int main(){
        float val1 = 4.5;
        float val2 = 7.3;
        int val3 = 5;
        float soma = val1 + val2 + val3;
        int media = soma / 3;
    }
```

c.
```
#include <stdio.h>
    int main(){
        int val1 = 4.5;
        int val2 = 7.3;
        int val3 = 5;
        int soma = val1 + val2 + val3;
        int media = soma / 3;
    }
```

d.
```
#include <stdio.h>
    int main(){
        float val1 = 4.5;
        float val2 = 7.3;
        int val3 = 5;
        int soma = val1 + val2 + val3;
        float media = soma / 3;
    }
```

2 ENTRADA E SAÍDA DE DADOS

8.7 CONHECIMENTO

1) A função _____ da biblioteca padrão exibe informações na tela.
   a. scanf
   b. printf
   c. #input<stdio.h>
   d. output

2) A função _____ da biblioteca padrão é usada para obter dados do teclado.
   a. scanf
   b. printf
   c. #input<stdio.h>
   d. output

3) O especificador de conversão _____ é usado em uma string de controle de formato de scanf para indicar que um inteiro será fornecido ao programa e em uma string de controle de formato de printf para indicar a impressão (saída) de um inteiro pelo programa.
   a. %d
   b. %f
   c. %c
   d. %i

4) O especificador de conversão _____ é usado em uma string de controle de formato de scanf para indicar que um float será fornecido ao programa e em uma string de controle de formato de printf para indicar a impressão (saída) de um float pelo programa.
   a. %d
   b. %f
   c. %c
   d. %i

5) O especificador de conversão _____ é usado em uma string de controle de formato de scanf para indicar que um char será fornecido ao programa e em uma string de controle de formato de printf para indicar a impressão (saída) de um char pelo programa.
   a. %d
   b. %f
   c. %c
   d. %i

6) A seqüência de escape \n representa o caractere de _____ que faz com que o cursor se posicione no início da próxima linha na tela.
   a. Letra n
   b. Exclusão de linha
   c. Nova linha
   d. \n

7) Na linguagem C pode-se inserir comentários no codigo, fazendo com que o texto situado entre _____ e _____ não apareça quando o programa é executado.
   a. */ e */
   b. / e /
   c. /* e */
   d. /* e /*

8.8 COMPREENSÃO

1) Verdadeiro ou falso: Um programa em C que imprime três linhas de saída deve conter três instruções printf.
   a. Verdadeiro
   b. Falso

2) Verdadeiro ou Falso: Quando a função printf é chamada, ela sempre começa a imprimir no início de uma nova linha.
   a. Verdadeiro
   b. Falso

3) As strings “%d”, “%c” e “%f” são normalmente utilizadas na função:
   a. main
   b. void
   c. stdio
   d. printf

4) Para armazenar o salário de um funcionário criou-se a variável salFunc, assim para ler a informação do usuário deve-se utilizar a função scanf com qual código de formatação?
   a. %c
   b. %d
   c. %e
   d. %f

8.9 APLICAÇÃO

1) Escreva o comando para imprimir a frase: “Hello world!”

2) Escreva o comando para armazenar um valor inteiro na variável idade, visto que essa informação é solicitada ao usuário:

3) Sabendo que um programa “x” calcula a área de um círculo, escreva o comando necessário para pedir o valor da variável raio ao usuário. Lembre-se que a medida de um raio pode ser um valor real.

4) Escreva o comando que mostre na tela um caractere lido do usuário sabendo que a variável que armazena esta informação tem como identificador algumCaracter:

8.10 ANÁLISE

1) Utilize V para verdadeiro e F para falso para os trechos de código à seguir:
   ( ) char sexo; scanf(“%c”, &sexo);
   ( ) float metros; scanf(“%d”; &metros);
   ( ) int idade; scanf(“%d”; idade);
   Marque a alternativa que corresponde a sequencia correta de cima para baixo:
   a. V, V, V
   b. V, V, F
   c. F, F, F
   d. V, F, F

2) Assinale a alternativa que apresenta o trecho de código correto para a variável salario:
   a. float salario; scanf(“%f”, &salario); printf(“Seu salario eh: R$%f”, salario);
   b. float salario; scanf(“%d”, &salario); printf(“Seu salario eh: R$%d”, salario);
   c. float salario; scanf(“%f”, salario); printf(“Seu salario eh: R$%f”, salario);
   d. float salário; scanf(“%f”, &salário); printf(“Seu salario eh: R$%f”, salário);

3) Jair é um astrólogo, porém tem muita dificuldade em descobrir o ano em que a pessoa nasceu ao saber de sua idade. Marque a alternativa correta para mostrar ao Jair a resposta de sua dificuldade:
   a. printf(“Seu cliente nasceu em: %c”, ano);
   b. printf(“Seu cliente nasceu em: %d”, ano);
   c. scanf(“Seu cliente nasceu em: %d”, ano);
   d. scanf(“%d”, Seu cliente nasceu em:, &ano);

8.11 SÍNTESE

1) Em uma única linha, escreva o trecho de código necessário para criar uma variável e armazenar a altura do usuário. Considere que o identificador da variável é altura:

2) Sabendo que o identificador de uma variável é media, em uma única linha, escreva o trecho de código necessário para apresentar ao usuário sua média com a frase: Sua média é:

3) Christopher deseja um programa para calcular o fatorial de um terminado valor. Através dos identificadores valor e valFat, em uma única linha crie as variáveis, leia o valor e mostre o resultado ao usuário. ****Não inclua a fórmula****.

8.12 AVALIAÇÃO

1) Numa fruteira, as bergamotas entraram em promoção, sendo que o cliente pode comprar pacotes de 1.5 Kg. Portanto, faça um programa que leia do usuário a quantidade de pacotes que serão comprados e o preço do Kg da bergamota. Ao final, imprima o valor a ser cobrado do cliente. Com base no enunciado, escolha o programa com a melhor solução:

a.
```
int main(int argc, char *argv[]) {
    int quantidade;
    float preco;
    float valor;
    printf("Digite a quantidade de pacotes: ");
    scanf("%d", &quantidade);
    printf("Digite o preço de cada pacote: ");
    scanf("%f", &preco);
    valor = preco * quantidade;
    printf("O valor a ser cobrado eh R$ %f", valor);
    return 0;
}
```

b.
```
int main(int argc, char *argv[]) {
    int quantidade;
    float preco;
    float valor;
    printf("Digite a quantidade de pacotes: ");
    scanf("%d", &quantidade);
    printf("Digite o preço de cada pacote: ");
    scanf("%d", &preco);
    valor = preco * quantidade;
    printf("O valor a ser cobrado eh R$ %d", valor);
    return 0;
}
```

c.
```
int main(int argc, char *argv[]) {
    int quantidade;
    float preco;
    float valor;
    printf("Digite a quantidade de pacotes: ");
    scanf("%d", &quantidade);
    printf("Digite o preço do Kg da bergamota: ");
    scanf("%f", &preco);
    valor = preco * quantidade * 1.5;
    printf("O valor a ser cobrado eh R$ %f", valor);
    return 0;
}
```

d.
```
int main(int argc, char *argv[]) {
    int quantidade;
    float preco;
    float valor;
    printf("Digite a quantidade de pacotes: ");
    scanf("%d", &quantidade);
    printf("Digite o preço do Kg da bergamota: ");
    scanf("%d", &preco);
    valor = preco * quantidade * 1.5;
    printf("O valor a ser cobrado eh R$ %d", valor);
    return 0;
}
```

2) Faça um programa capaz de calcular a media de 3 notas de um aluno. Com base no enunciado, escolha o programa com a melhor solução:

a.
```
int main(int argc, char *argv[]) {
    int n1, n2, n3, media;
    printf("Digite a primeira nota: ");
    scanf("%d", n1);
    printf("Digite a segunda nota: ");
    scanf("%d", n2);
    printf("Digite a terceira nota: ");
    scanf("%d", n3);
    media = (n1+n2+n3)/3;
    printf("a media das notas sera %d", media);
    return 0;
}
```

b.
```
int main(int argc, char *argv[]) {
    int n1, n2, n3, media;
    printf("Digite a primeira nota: ");
    scanf("%d", &n1);
    printf("Digite a segunda nota: ");
    scanf("%d", &n2);
    printf("Digite a terceira nota: ");
    scanf("%d", &n3);
    media = (n1+n2+n3)/3;
    printf("a media das notas sera %d", media);
    return 0;
}
```

c.
```
int main(int argc, char *argv[]) {
    int n1, n2, n3, media;
    printf("Digite a primeira nota: ");
    scanf("%d", &n1);
    printf("Digite a segunda nota: ");
    scanf("%d", &n2);
    printf("Digite a terceira nota: ");
    scanf("%d", &n3);
    media = (n1+n2+n3)/3;
    printf("a media das notas sera %d", &media);
    return 0;
}
```

d.
```
int main(int argc, char *argv[]) {
    int n1, n2, n3, media;
    printf("Digite a primeira nota: ");
    scanf("%d", n1);
    printf("Digite a segunda nota: ");
    scanf("%d", n2);
    printf("Digite a terceira nota: ");
    scanf("%d", n3);
    media = (n1+n2+n3)/3;
    printf("a media das notas sera %d", &media);
    return 0;
}
```

3 OPERADORES ARITMÉTICOS

8.13 CONHECIMENTO

1) Normalmente, os cálculos são realizados por instruções _____.

a. aritmética
b. lógicas
c. c ANSI
d. de entrada e saída

2) O sinal de porcentagem (%) na linguagem C ANSI indica o operador _____.
a. Dividendo
b. Divisor
c. Resultado
d. Resto

8.14 COMPREENSÃO

1) Qual alternativa contém a sequência correta de prioridade dos operadores matemáticos?
a. * / () %
b. () */ + -
c. () + * - /
d. / * () + -

2) Verdadeiro ou Falso: Os operadores aritméticos *, /, % e - possuem o mesmo nível de precedência.
a. Verdadeiro
b. Falso

3) De acordo com as regras de precedência, selecione o(s) operador(es) que tem a associatividade da esquerda para a direita:
a. ( )
b. * / %
c. + -
d. < <= > >=
e. == !=
f. =

8.15 APLICAÇÃO

1) Sabendo que a=48, b=6 e c=3, qual será o resultado da expressão: x = a % 15 * b + a / c – b;
a. 58
b. 28
c. 60
d. 48

8.16 ANÁLISE

1) Dada a equação $y = ax^3 + 7$, qual instrução em C a seguir é correta para ela?
a. y = a * x * x * x + 7;
b. y = a * x * x * (x + 7);
c. y = (a * x) * x * (x + 7) ;
d. y = a * x * (x * x + 7);

2) A empresa SuperBigMaster está entrando na liquida Porto Alegre e precisa que seu sistema de 30% de desconto em todos os produtos vendidos na loja, retornando o valor final para venda. Qual das fórmulas abaixo corresponde ao que deve ser implementado?
   a. valFinal = valProduto * 30%;
   b. valFinal = valProduto + (valProduto * 30%);
   c. valFinal = valProduto * 0.7;
   d. valFinal = valProduto – 30%;

8.17 SÍNTESE

1) Tendo como identificador salario e salFinal, em uma única linha escreva o trecho de código que armazene os dados, crie as variáveis, calcule o aumento de 5% no salario e mostre o resultado para ao usuário com a seguinte frase: Seu salário será R$

2) Sabendo que o usuário irá digitar valores inteiros para as variáveis valor1 e valor2, em uma única linha de código escreva o código que armazene os dados, crie as variáveis, some os valores e mostre o resultado para ao usuário com a seguinte frase: A soma é

8.18 AVALIAÇÃO

1) Ana Matha é estudante de licenciatura em matemática e está fazendo estágio, sua turma tem dificuldades em calcular a área de um triangulo retângulo. Considerando que os identificadores são: base, altura e area assinale a alternativa que melhor apresenta a solução deste programa.
   a.
```
int main(int argc, char *argv[]) {
        float base;
        float altura;
        float area;
        printf("Digite a base do triangulo: ");
        scanf("%f", &base);
        printf("Digite a altura do triangulo: ");
        scanf("%f", &altura);
        area = base * altura / 2;
        printf("A área do triângulo é R$ %f", area);
        return 0;
    }
```
   b.
```
int main(int argc, char *argv[]) {
        float base;
        float altura;
        float area;
        printf("Digite a base do triangulo: ");
        scanf("%f", base);
        printf("Digite a altura do triangulo: ");
        scanf("%f", altura);
        area = base * altura / 2;
        printf("A área do triângulo é R$ %f", &area);
        return 0;
    }
```
   c.
```
int main(int argc, char *argv[]) {
        float base;
        float altura;
        float area;
        printf("Digite a base do triangulo: ");
        scanf("%f", &base);
```

```
        printf("Digite a altura do triangulo: ");
        scanf("%f", &altura);
        area = base * (altura / 2);
        printf("A área do triângulo é R$ %f", area);
        return 0;
    }
```

d.
```
int main(int argc, char *argv[]) {
    float base;
    float altura;
    float area;
    printf("Digite a base do triangulo: ");
    scanf("%d", &base);
    printf("Digite a altura do triangulo: ");
    scanf("%d", &altura);
    area = base * altura / 2;
    printf("A área do triângulo é R$ %d", area);
    return 0;
}
```

2) Para calcular a velocidade média de um percurso feito por um ciclista é necessário conhecer a quantidade de horas que se levou no percurso e também a distância percorrida. Assinale a alternativa que melhor apresenta a solução este programa.

a.
```
int main(int argc, char *argv[]) {
    float qtdHoras;
    float distancia;
    float velMedia;
    printf("Digite quantas horas levou para chegar ao destino: ");
    scanf("%f", &qtdHoras);
    printf("Digite a distância percorrida: ");
    scanf("%f", &distancia);
    velMedia = qtdHoras / distancia;
    printf("A velocidade média foi %f", velMedia);
    return 0;
}
```

b.
```
int main(int argc, char *argv[]) {
    float qtdHoras;
    float distancia;
    float velMedia;
    printf("Digite quantas horas levou para chegar ao destino: ");
    scanf("%f", &qtdHoras);
    printf("Digite a distância percorrida: ");
    scanf("%f", &distancia);
    velMedia = qtdHoras * distancia;
    printf("A velocidade média foi %f", velMedia);
    return 0;
}
```

c.
```
int main(int argc, char *argv[]) {
    float qtdHoras;
    float distancia;
    float velMedia;
    printf("Digite quantas horas levou para chegar ao destino: ");
    scanf("%f", &qtdHoras);
    printf("Digite a distância percorrida: ");
    scanf("%f", &distancia);
    velMedia = distancia * qtdHoras;
    printf("A velocidade média foi %f", velMedia);
    return 0;
}
```

d.
```
int main(int argc, char *argv[]) {
    float qtdHoras;
```

```
        float distancia;
        float velMedia;
        printf("Digite quantas horas levou para chegar ao destino: ");
        scanf("%f", &qtdHoras);
        printf("Digite a distância percorrida: ");
        scanf("%f", &distancia);
        velMedia = distancia / qtdHoras;
        printf("A velocidade média foi %f", velMedia);
        return 0;
    }
```

4 OPERADORES RELACIONAIS

8.19 CONHECIMENTO

1) Os símbolos >= significa:
   a. Menor igual
   b. Maior igual
   c. Maior
   d. Mais igual

2) O sinal de diferente na linguagem C ANSI é representado pelos símbolos.
   a. <>
   b. ><
   c. !
   d. !=

3) O sinal de igualdade na linguagem C ANSI é representado pelos símbolos:
   a. equal
   b. ==
   c. =
   d. !!=

8.20 COMPREENSÃO

1) Para identificar se um valor x é menor que um valor y, a comparação deve ser feita de que forma?
   a. x >= y
   b. x <= y
   c. x < y
   d. x > y

2) Sabendo que a=5 e b=10, marque a alternativa que apresentará como resultado Verdadeiro.
   a. a > b
   b. a == b
   c. a >= b
   d. a < b

3) Jefferson quer desenvolver um programa que verifique se um valor é par, com o uso da %, qual alternativa solucionaria o problema de Jefferson?
   a. valor %2 = 0
   b. valor %2 >= 1

c. valor %2 == 0
d. valor % 2 == 1

8.21 APLICAÇÃO

1) Escreva um trecho de código que compare dois valores, mostrando o maior valor para o usuário:

2) Escreva um trecho de código que verifique se o valor digitado pelo usuário é positivo

8.22 ANÁLISE

1) Dada a comparação: a<=c quais valores mostrariam como resposta Falso?
   a. a = 33 c = 97
   b. a = 2 c = 22
   c. a = 1 c = -10
   d. a = -1 c = -18

2) Dada a comparação: a>=c quais valores mostrariam como resposta Verdadeiro?
   a. a = -418 c = 127
   b. a = -4 c = -333
   c. a = 7 c = 77
   d. a = 0 c = 18

3) Sabendo que A=1, B=18 e C=2, informe se as expressões abaixo são verdadeiras ou falsas.
   1. (A+C) > B
   2. B >= (A + 2)
   3. C == (B –A)
   4. (B + A) <= C

Marque a alternativa que corresponde a resposta correta da esquerda para a direita:
a. F | V | F | F
b. V | V | V | V
c. V | F | V | V
d. F | F | F | F

8.23 SÍNTESE

1) Desenvolva um programa que seja capaz de dizer se o usuário pode ou não fazer carteira de motorista, recebendo as seguintes informações: nome e ano de nascimento.

2) Sabe-se que um triângulo escaleno possui três lados diferentes, e para que o mesmo seja formado corretamente, é necessário que o comprimento do maior dos lados seja menos que a soma dos comprimentos dos lados menores. Assim sendo, faça um programa que receba o valor dos 3 lados do triângulo, e informe se esses formam ou não um triângulo escaleno.

8.24 AVALIAÇÃO

1) Verifique qual código apresenta a melhor solução para o seguinte problema: João precisa de um programa que receba o salário de um funcionário e o mesmo informe a qual setor ele pertence, levando em conta a tabela abaixo:

Setor	Faixa salarial
Diretoria	Maior que 5000,00
Gerência	De 3200,00 a 4999,99
Vendas	Até 3199,99

a.
```
int main(int argc, char *argv[]) {
    float sal;
    printf("Digite seu salario: ");
    scanf("%f", &sal);
    if(sal > 3200)
        printf("Vendas");
    else if(sal >=3200)
        printf(“Gerência”);
    else
        printf(“Diretoria”);
    return 0;
}
```

b.
```
int main(int argc, char *argv[]) {
    float sal;
    printf("Digite seu salario: ");
    scanf("%f", &sal);
    if(sal >= 5000)
        printf("Diretoria");
    if(sal >=3200)
        printf(“Gerência”);
    if(sal < 3200)
        printf(“Vendas”);
    return 0;
}
```

c.
```
int main(int argc, char *argv[]) {
    float sal;
    printf("Digite seu salario: ");
    scanf("%f", &sal);
    if(sal >= 5000)
        printf("Diretoria");
    else if(sal >=3200)
        printf(“Gerência”);
    else
        printf(“Vendas”);
    return 0;
}
```

d.
```
int main(int argc, char *argv[]) {
    float sal;
    printf("Digite seu salario: ");
    scanf("%f", &sal);
    if(sal <= 5000)
        printf("Diretoria");
    else if(sal <=3200)
        printf(“Gerência”);
    else
        printf(“Vendas”);
    return 0;
}
```

8.25 CONHECIMENTO

1) O operador lógico && é utilizado como:
   a. or
   b. and
   c. negação
   d. soma

2) O símbolo utilizado como operador lógico “ou” é:
   a. or
   b. ||
   c. &&
   d. <>

3) O sinal de exclamação (!) na linguagem C ANSI indica uma _____.
   a. afirmação
   b. conclusão
   c. atenção
   d. negação

8.26 COMPREENSÃO

1) Quando utilizamos o operador lógico &&, o resultado só será verdadeiro se todas as condições forem:
   a. Falsas
   b. Verdadeiras
   c. Diferentes
   d. Iguais

2) Quando utilizamos o operador lógico ||, o resultado será verdadeiro se:
   a. Pelo menos uma das condições for verdadeira
   b. Todas as condições forem verdadeiras
   c. Todas as condições forem falsas
   d. Todas as condições forem diferentes

3) Quando utilizamos o operador lógico ||, o resultado só será falso se todas as condições forem:
   a. Falsas
   b. Verdadeiras
   c. Diferentes
   d. Iguais

4) Quando utilizamos o operador lógico &&, o resultado será falso se:
   a. Pelo menos uma das condições for falsa
   b. Todas as condições forem verdadeiras
   c. Todas as condições forem falsas
   d. Todas as condições forem diferentes

8.27 APLICAÇÃO

1) Qual resultado lógico para a equação (b>=c)&&(a!=b)? Considere os seguintes valores: a=10, b=15, c=3
   a. V
   b. F

2) Qual resultado lógico para a equação (((a % 15 * b) >= (15 * b + a)) || !(b==c))? Considere os seguintes valores: a=30, b=4, c=43
   a. V
   b. F

8.28 ANÁLISE

1) Uma vez que a=V, b=V e c=F, qual comparação abaixo apresentará F como resposta?
   a. a && b || c && a
   b. (a || b) || (c && a)
   c. (!a && b) || (c && a)
   d. (!a || b) || (c || a)

2) Uma vez que a=V, b=V e c=F, qual comparação abaixo apresentará V como resposta?
   a. a && b || c && a *
   b. (a || b) && (c && a)
   c. (!a && b) && (c && a)
   d. (!a && b) || (c && a)

8.29 SÍNTESE

1) Desenvolva um programa que seja capaz de dizer se o usuário deve se alistar ao serviço militar através do recebimento da idade e sexo. Lembre-se que a obrigatoriedade do serviço militar é para homens que tenham pelo menos 18 anos.

2) Sabendo que a fórmula do cálculo do índice de massa corpórea é peso/altura², através da tabela abaixo, apresente o programa que retorne ao usuário sua situação:

IMC	**Homem**	**Mulher**
Obesidade mórbida	Maior que 43	Maior que 39
Obesidade moderada	De 30 a 39,9	De 29 a 38,9
Obesidade leve	De 25 a 29,9	De 24 a 28,9
Normal	De 20 a 24,9	De 19 a 23,9
Abaixo do normal	Menor que 20	Menor que 19,9

8.30 AVALIAÇÃO

1) Ano bissexto refere-se ao ano que possui 366 dias ao invés de 365. Para sabermos se um ano é bissexto, existem algumas regras: a cada quatro anos será ano bissexto; de 100 em 100 anos, não será ano bissexto; de 400 em 400 anos, é ano bissexto; sendo que as últimas regras prevalecem sobre as primeiras. Sabendo disso, marque a alternativa que melhor soluciona esse problema.
   a. int main(int argc, char *argv[]) {

```
        int ano;
        printf("Digite o ano a ser verificado:");
        scanf("%d", &ano);
        if((ano%400==0)||((ano%100!=0)&&(ano%400==0))
                printf("É bissexto")
        else
                printf("Não pe bissexto");
        return 0;
    }
```

b.
```
    int main(int argc, char *argv[]) {
        int ano;
        printf("Digite o ano a ser verificado:");
        scanf("%d", &ano);
        if((ano%400==0)||((ano%100==0)&&(ano%400==0))
                printf("É bissexto")
        else
                printf("Não pe bissexto");
        return 0;
    }
```

c.
```
    int main(int argc, char *argv[]) {
        int ano;
        printf("Digite o ano a ser verificado:");
        scanf("%d", &ano);
        if((ano%400==0)
                printf("É bissexto")
        if else((ano%100!=0)&&(ano%400==0))
                printf("É bissexto");
        else
                printf("Não pe bissexto");
        return 0;
    }
```

d.
```
    int main(int argc, char *argv[]) {
        int ano;
        printf("Digite o ano a ser verificado:");
        scanf("%d", &ano);
        if((ano%4==0)
                printf("É bissexto")
        else
                printf("Não pe bissexto");
        return 0;
    }
```

6 ESTRUTURAS DE CONTROLE

8.31 CONHECIMENTO

1) A instrução _____ é usada na tomada de decisões na Linguagem de programação C ANSI.
   a. if
   b. main
   c. printf
   d. scanf

2) Para que um programa tome decisões com base na veracidade ou falsidade de alguma instrução ou fato é necessário uma _____.

a. chave
b. situação
c. condição
d. ação

3) Em uma decisão se a condição for atendida a instrução no corpo da estrutura if é _____.
a. anulada
b. ignorada
c. retornada
d. executada

4) A instrução if permite ao programador tomar uma decisão, se a condição for falsa, a instrução do corpo é _____.
a. anulada
b. ignorada
c. retornada
d. executada

5) Normalmente, as condições em instruções if são formadas usando operadores de _____ e _____.
a. igualdade / relacionais
b. soma / subtração
c. igualdade / divisão
d. não usa operadores

6) O formato de uma instrução if é :
a. if (instrução) condição;
b. if {instrução} condição;
c. if {condição} instrução;
d. if (condição) instrução;

7) A estrutura de seleção switch é chamada estrutura de _____ porque seleciona uma entre muitas ações diferentes.
a. repetição múltipla
b. seleção múltipla
c. adaptação múltipla
d. ação múltipla

8) A estrutura switch consiste em uma série de rótulos _____ e de um default _____.
a. case / opcional
b. case / obrigatório
c. if / opcional
d. if / obrigatório

9) A instrução switch é seguida de uma _____ entre parênteses que é _____ com cada um dos case.
a. expressão controle / armazenado
b. expressão controle / comparado
c. expressão lógica / comparado
d. expressão lógica / armazenado

10) A instrução _____ faz com que o controle do programa continue com a primeira instrução _____ a estrutura switch.
a. continue / após
b. continue / incial
c. break / inicial
d. break / após

11) Se não houver nenhuma _____, o caso _____ é executado e uma mensagem de erro é impressa.
    a. disparidade / default
    b. disparidade / break
    c. igualdade / default
    d. igualdade / break

8.32 COMPREENSÃO

1) Verdadeiro ou Falso: Ao finalizar a instrução *if* deve-se incluir um ponto e vírgula (;) no final como o exemplo a seguir: if(nota>6); printf(“Aprovado”);
    a. Verdadeiro
    b. Falso

2) Qual das frases abaixo define melhor a instrução *if*?
    a. *if* é uma instrução da Linguagem de Programação C ANSI que pertence às estruturas de repetição.
    b. *if* é uma instrução da Linguagem de Programação C ANSI que pertence às estruturas de controle.
    c. *if* é um identificador da Linguagem de Programação C ANSI que pertence às estruturas de repetição.
    d. *if* é um identificador da Linguagem de Programação C ANSI que pertence às estruturas de controle.

3) Verdadeiro ou Falso: O uso da instrução *else* é opcional?
    a. Verdadeiro
    b. Falso

4) Qual das frases abaixo define melhor a instrução *else*?
    a. Caso a comparação da instrução *if* resultar em falso, os comandos contidos na instrução *else* serão executadas.
    b. Caso a comparação da instrução *if* resultar em verdadeiro, os comandos contidos na instrução *else* serão executadas.
    c. Caso a comparação da instrução *if* resultar em falso, os comandos contidos na instrução *else* não serão executadas.
    d. Caso a comparação da instrução *if* resultar em verdadeiro, os comandos contidos na instrução *else* não serão executadas.

5) Suponha que o usuário forneceu a letra C como um conceito (grau). C é _____ automaticamente a cada case em switch. Se acontecer alguma igualdade (case 'C' :) as instruções para aquele case _____.
    a. armazenado / são executadas
    b. armazenado / não são executadas
    c. comparado / não são executadas
    d. comparado / são executadas

6) Verdadeiro ou Falso: Em uma estrutura switch em que a cláusula default está colocada por último, a instrução break não é exigida ali.
    a. Verdadeiro
    b. Falso

7) Verdadeiro ou Falso: Embora as cláusulas case e a cláusula do caso default em uma estrutura switch possam ocorrer em qualquer ordem, é considerado uma boa prática de programação colocar a cláusula default primeiro.
    a. Verdadeiro
    b. Falso

8) Verdadeiro ou Falso: A estrutura de seleção switch exige o caso default.
   a. Verdadeiro
   b. Falso

8.33 APLICAÇÃO

1) Em uma única linha crie uma estrutura *if* que verifique se um número é par ou ímpar. Leve em conta que o identificador da variável é valor e que deve mostrar a mensagem: O valor digitado é

2) Godofredo é um estudante de programação e está desenvolvendo um programa que faz parte da avaliação do semestre. Porém Godofredo está com dificuldades em criar a estrutura correta para verificar se um valor digitado pelo usuário é negativo, positivo ou nulo. Ajude Godofredo criando esta estrutura em uma única linha, levando em conta que o nome da variável é num. A mensagem para o usuário deve ser simples, apresentando na tela apenas a resposta.

3) Para doar sangue é necessário ter entre 18 e 67 anos. Sabendo que a variável se chama idade, em uma única linha crie a estrutura que informe ao usuário se ele pode ou não ser doador. Use alguns dos operadores lógicos OU (||) e E (&&) e as frases “Pode doar sangue” ou “Não pode doar sangue”.

4) Crie uma estrutura switch para escrever o número digitado pelo usuário (0 à 10) por extenso. Por exemplo: se o usuário digitou o número 7, o programa deverá apresentar a palavra sete. Não esqueça de colocar uma mensagem de erro caso o usuário digite um valor que não esteja no intervalo de 0 à 10.

5) Uma empresa paga seus empregados como gerentes (que recebem um salário fixo mensal), trabalhadores comuns (que recebem um salário fixo por hora para as primeiras 40 horas de trabalho e 1,5 vez seu salário por hora normal para as horas extras trabalhadas), trabalhadores por comissão (que recebem $250 mais 5,7% de suas vendas brutas) ou trabalhadores por empreitada (que recebem uma quantia fixa por item para cada um dos itens produzidos — cada trabalhador por empreitada dessa empresa trabalha com apenas um tipo de item). Escreva um programa que calcule o pagamento semanal de cada empregado. Você não sabe de antemão o número de empregados. Cada tipo de empregado tem seu código próprio de pagamento: gerentes possuem o código 1, trabalhadores comuns, o código 2, trabalhadores por comissão, o código 3, e trabalhadores por empreitada, o código 4. Desenvolva o bloco de código switch para calcular o pagamento de cada empregado com base em seu código de pagamento. Dentro do switch peça ao usuário para entrar com os fatos adequados à necessidade de seu programa para calcular o pagamento de cada empregado com base em seu código.

8.34 ANÁLISE

1) Considerando o trecho de código abaixo, onde x é uma variável do tipo int:

```
if((x<10)&&(x>0)){
   printf("viva!");
}
```

   ASSINALE A ALTERNATIVA **FALSA**
   a. o texto "viva!" nunca será impresso pois a condição é impossível de ser satisfeita.
   b. se x o valor de x for 10 (dez), nada será impresso.
   c. se x o valor de x for 0 (zero), nada será impresso.
   d. se x for igual a 5, o programa imprimirá o texto "viva!"

2) Qual o valor final da variável “a” após a execução do código abaixo?

```
int x = 5;
int a =7;
if (a == 5) {
   a = 1;
}
else {
    if (x == 7) a = 2;
    else a = 20;
}
```

a. 1
b. 2
c. 20
d. 7

3) Qual o valor final da variável “a” após a execução do código abaixo?

```
int x = 7;
int a =7;
if(a==5) {
    a=1;
}
else {
   if (x==7) a=2;
   else a = 20;
}
```

a. 1
b. 2
c. 20
d. 7

4) Uma empresa de autoescola necessita de um programa que verifique através da idade da pessoa se a mesma pode ou não tirar a habilitação, lembre-se que no Brasil somente pessoas com ou acima dos 18 anos tem permissão para tentar tirar carteira de motorista. Assinale a alternativa que apresenta a solução para esse problema:
   a. if (idade => 18) printf (“Pode tirar habilitação);
   b. if (idade <= 18) printf (“Pode tirar habilitação);
   c. if (idade > 17) printf (“Pode tirar habilitação);
   d. if (idade > 18) printf (“Pode tirar habilitação);

5) Que mensagem irá ser mostrada para o usuário ao executar o código abaixo? Considere que a entrada de dado é 7.

```
#include<stdio.h>
int main(){
        int x;
        scanf(“%d”, &x);
        switch(x){
                case ‘1’:
                        printf(“100”);
                        break;
                case ‘2’:
                        printf(“200”);
                        break;
                case ‘3’:
                        printf(“300”);
                        break;
                default:
                        printf(“600”);
                        break;
        }
}
```

a. 100

b. 200
c. 300
d. 600

6) Que mensagem irá ser mostrada para o usuário ao executar o código abaixo? Considere que a entrada de dado é 11.

```
#include<stdio.h>
int main(){
        int x;
        scanf("%d", &x);
        switch(x){
                case '1':
                        printf("100");
                        break;
                case '14':
                case '15':
                case '16':
                        printf("200");
                        break;
                case '17':
                        printf("300");
                        break;
                default:
                        printf("600");
                        break;
        }
}
```

a. 100
b. 200
c. 300
d. 600

7) Que mensagem irá ser mostrada para o usuário ao executar o código abaixo? Considere que a entrada de dado é 15.

```
#include<stdio.h>
int main(){
        int x;
        scanf("%d", &x);
        switch(x){
            case '1':
                printf("100");
                break;
            case '14':
            case '15':
            case '16':
                printf("200");
                break;
            case '17':
                printf("300");
                break;
            default:
                printf("600");
                break;
        }
}
```

a. 100
b. 200
c. 300
d. 600

8.35 SÍNTESE

1) Em uma escola X os estudantes que tem média final acima de 7 estão aprovados, já aqueles que possuem como média final entre 4 e 6,9 ficam de recuperação, os demais reprovam. Sabendo que o identificador é mediaFinal, em uma única linha desenvolva o código necessário para solucionar esse problema que apresentará ao usuário mensagens simples: “Digite sua média final:”; “Você está aprovado” ou “Você está reprovado” ou “Você está em recuperação”.
*** O código deverá apresentar os comandos que ficam entre “int main(){” e “return 0;}” ***

2) Desenvolva um programa sabendo que o mesmo deve pedir um valor inteiro ao usuário e retornar o dia da semana que este corresponde. Caso o valor não corresponda a um valor válido, informe ao usuário o mesmo:

8.36 AVALIAÇÃO

1) O sistema que calcula o desconto a ser dado em um produto de acordo com a forma de pagamento que é armazenada na variável modo. Os descontos são dados da seguinte forma: 1 - Em dinheiro: 10%; 2 - No débito: 5%; 3 - Parcelado: sem desconto; No final o programa retorna o valor a ser pago. Assinale a alternativa que melhor representa a solução para este programa:

a.
```
int main(int argc, char *argv[]) {
        int modo;
        float valorPago, valorProduto;
        printf("Digite a forma de pagamento: 1 - Dinheiro 2 - Débito 3 - Parcelado ");
        scanf("%d", &modo);
        printf("Digite o valor do produto: ");
        scanf("%f", &valorProduto);
        if(modo==1) valorPago = valorProduto - valorProduto * 0.10;
        else if(modo==2) valorPago = valorProduto - valorProduto * 0.10;
        else if(modo==3) valorPago = valorProduto;
        else printf(“Opção inválida”);
        printf("Digite o valor a ser pago é R$%f ", valorPago);
        return 0;
}
```

b.
```
int main(int argc, char *argv[]) {
        int modo;
        float valorPago, valorProduto;
        printf("Digite a forma de pagamento: 1 - Dinheiro 2 - Débito 3 - Parcelado ");
        scanf("%d", &modo);
        printf("Digite o valor do produto: ");
        scanf("%f", &valorProduto);
        if(modo==1) valorPago = valorProduto - valorProduto * 0,10;
        else if(modo==2) valorPago = valorProduto - valorProduto * 0,10;
        else if(modo==3) valorPago = valorProduto;
        else printf(“Opção inválida”);
        printf("Digite o valor a ser pago é R$%f ", valorPago);
        return 0;
}
```

c.
```
int main(int argc, char *argv[]) {
        int modo;
        float valorPago, valorProduto;
        printf("Digite a forma de pagamento: 1 - Dinheiro 2 - Débito 3 - Parcelado ");
        scanf("%d", &modo);
        printf("Digite o valor do produto: ");
        scanf("%f", &valorProduto);
        if(modo==1); valorPago = valorProduto - valorProduto * 0.10;
```

```
        else if(modo==2); valorPago = valorProduto - valorProduto * 0.10;
        else if(modo==3); valorPago = valorProduto;
        else printf(“Opção inválida”);
        printf("Digite o valor a ser pago é R$%f ", valorPago);
        return 0;
}
```

d.
```
int main(int argc, char *argv[]) {
        int modo;
        float valorPago, valorProduto;
        printf("Digite a forma de pagamento: 1 - Dinheiro 2 - Débito 3 - Parcelado ");
        scanf("%d", &modo);
        printf("Digite o valor do produto: ");
        scanf("%f", &valorProduto);
        if(modo==1) valorPago = valorProduto - valorProduto * 0.10;
        else if(modo==2) valorPago = valorProduto - valorProduto * 0.10;
        else if(modo==3) valorPago = valorProduto;
        else printf(“Opção inválida”);
        printf("Digite o valor a ser pago é R$%f ", valorProduto);
        return 0;
}
```

2) O sistema que calcula o desconto a ser dado em um produto de acordo com a forma de pagamento que é armazenada na variável modo. Os descontos são dados da seguinte forma: 1 - Em dinheiro: 10%; 2 - No débito: 5%; 3 - Parcelado: sem desconto; No final o programa retorna o valor a ser pago. Assinale a alternativa que melhor representa a solução para este programa:

a.
```
int main(int argc, char *argv[]) {
        int modo;
        float valorPago, valorProduto;
        printf("Digite a forma de pagamento: 1 - Dinheiro 2 - Débito 3 - Parcelado ");
        scanf("%d", &modo);
        printf("Digite o valor do produto: ");
        scanf("%f", &valorProduto);
        switch(modo){
            case ‘1’:
                valorPago = valorProduto - valorProduto * 0.10;
                break;
            case ‘2’:
                valorPago = valorProduto - valorProduto * 0.10;
                break;
            case ‘3’:
                valorPago = valorProduto;
                break;
            default:
                printf(“Opção inválida”);
         }
        printf("O valor a ser pago é R$%f ", valorPago);
        return 0;
}
```

b.
```
int main(int argc, char *argv[]) {
        int modo;
        float valorPago, valorProduto;
        printf("Digite a forma de pagamento: 1 - Dinheiro 2 - Débito 3 - Parcelado ");
        scanf("%d", &modo);
        printf("Digite o valor do produto: ");
        scanf("%f", &valorProduto);
        switch(&modo){
            case ‘1’:
                valorPago = valorProduto - valorProduto * 0.10;
                break;
            case ‘2’:
```

```
                valorPago = valorProduto - valorProduto * 0.10;
                break;
            case ‘3’:
                valorPago = valorProduto;
                break;
            default:
                printf(“Opção inválida”);
                break;
        }
    printf("O valor a ser pago é R$%f ", valorPago);
    return 0;
}
```

c.
```
int main(int argc, char *argv[]) {
    int modo;
    float valorPago, valorProduto;
    printf("Digite a forma de pagamento: 1 - Dinheiro 2 - Débito 3 - Parcelado ");
    scanf("%d", modo);
    printf("Digite o valor do produto: ");
    scanf("%f", valorProduto);
    switch(modo){
        case ‘1’:
            valorPago = valorProduto - valorProduto * 0.10;
            break;
        case ‘2’:
            valorPago = valorProduto - valorProduto * 0.10;
            break;
        case ‘3’:
            valorPago = valorProduto;
            break;
        default:
            printf(“Opção inválida”);
        }
    printf("O valor a ser pago é R$%f ", valorPago);
    return 0;
}
```

d.
```
int main(int argc, char *argv[]) {
    int modo;
    float valorPago, valorProduto;
    printf("Digite a forma de pagamento: 1 - Dinheiro 2 - Débito 3 - Parcelado ");
    scanf("%d", &modo);
    printf("Digite o valor do produto: ");
    scanf("%f", &valorProduto);
    switch(modo);
        case ‘1’:
            valorPago = valorProduto - valorProduto * 0.10;
            break;
        case ‘2’:
            valorPago = valorProduto - valorProduto * 0.10;
            break;
        case ‘3’:
            valorPago = valorProduto;
            break;
        default:
            printf(“Opção inválida”);
    printf("O valor a ser pago é R$%f ", valorPago);
    return 0;
}
```

8.37 CONHECIMENTO

1) Um loop é um grupo de instruções que o computador executa _____ enquanto alguma _____ de continuação de loop permanecer _____.
   a. repetidamente / igualdade / verdadeira
   b. repetidamente / condição / falsa
   c. repetidamente / condição / verdadeira
   d. rapidamente / condição / verdadeira

2) A repetição controlada por contador também é conhecida como repetição _____ porque sabe-se de antemão quantas vezes o _____ será executado.
   a. indefinida / loop
   b. definida / if
   c. definida / loop
   d. indefinida / if

3) A variável de controle contador é _____ pela expressão _____.
   a. decrementada / contador++
   b. incrementada / contador+
   c. incrementada / contador*
   d. incrementada / contador++

4) O while é uma _____, ou seja, um trecho de código será executado enquanto uma determinada condição for _____.
   a. estrutura de repetição / verdadeira
   b. estrutura de repetição / falsa
   c. estrutura de controle / verdadeira
   d. estrutura de controle / falsa

5) O trecho de código dentro do bloco do *while* será executado até que a _____ especificada resulte em _____.
   a. condição / verdadeira
   b. condição / falsa
   c. variável / vardadeira
   d. variável / falsa

6) No while a expressão de repetição é verificada cada vez que o laço é _____.
   a. iniciado
   b. finalizado
   c. analisado
   d. excluido

7) No comando *do…while* o bloco de comandos _____ pelo menos uma vez, _____ do resultado da expressão lógica.
   a. é executado / independente
   b. pode ser executado / independente
   c. é executado / dependendo
   d. pode ser executado / dependendo

8.38 COMPREENSÃO

1) Verdadeiro ou Falso: A repetição controlada por contador exige:
( ) O nome de uma variável de controle (ou contador do loop).
( ) O valor inicial da variável de controle.
( ) O incremento (ou decremento) pelo qual a variável de controle é somada cada vez que o loop é realizado.
( ) A condição que testa o valor final da variável de controle (i.e., se o loop deve continuar).

Marque a alternativa que corresponde à sequência correta de cima para baixo:
a. V, V, V, V
b. F, F, F, F
c. F, V, F, V
d. V, V, F, V

2) Verdadeiro ou Falso: Como os valores em ponto flutuante podem ser valores aproximados, controlar a contagem de loops com variáveis de ponto flutuante pode resultar em valores precisos de contadores e exames corretos da condição de terminação.
a. Verdadeiro
b. Falso

3) Sabendo que a estrutura do *for* é: for(variável de controle com valor inicial; valor final da variável controle; incremento da variável controle), respona V para Verdadeiro e F para Falso.
( ) As três expressões na estrutura *for* são opcionais.
( ) Se expressão2 for omitida, a linguagem C presume que a condição é verdadeira, criando assim um loop infinito.
( ) A expressão1 pode ser omitida.
( ) A expressão3 não pode ser omitida mesmo que o incremento seja calculado por instruções no corpo da estrutura *for* ou se nenhum incremento se faz necessário.
( ) A expressão de incremento na estrutura *for* age como uma instrução independente do C no final do corpo do *for*.

Marque a alternativa que corresponde à sequência correta de cima para baixo:
a. V, V, F, F, V
b. F, F, F, V, V
c. V, F, V, F, F
d. V, V, V, F, F

4) A sintaxe do comando *while* é:

```
while(condição){
        comando
}
```

No qual:
a. condição é uma variável e comando pode ser apenas uma instrução ou um bloco de instruções.
b. condição é uma expressão lógica e comando pode ser apenas uma instrução ou um bloco de instruções.
c. condição é uma variável e comando pode ser apenas uma instrução.
d. condição é uma expressão lógica e comando pode ser apenas uma instrução.

5) Sabendo que o usuário digitou 10, quantas vezes o laço de repetição será executado?

```
#include<stdio.h>
int main(){
                int valor;
                scanf(%d, &valor);
                while(valor >= 20){
                        valor ++ ;
                }
                return 0;
```

```
}
```

a. 10
b. 9
c. 11
d. Loop infinito

6) Verdadeiro ou Falso: Em um loop em que deve realizar a soma de 10 valores, no bloco de instruções do while(controle<=10) não se pode esquecer de decrementar a variável controle que inicia em 1.
   a. Verdadeiro
   b. Falso

8.39 APLICAÇÃO

1) Monte a estrutura *for* em que a variável de controle assuma os valores da seguinte sequência: 2, 5, 8, 11, 14, 17, 20.

2) Escreva uma instrução ou um conjunto de instruções em C para realizar a soma dos inteiros ímpares entre 1 e 99 usando uma estrutura for. Admita que as variáveis inteiras soma e contagem já foram declarados.

3) Crie uma estrutura *for* capaz de mostrar a tabuada do 10 da seguinte forma:

10 x 1 = 10	10 x 6 = 60
10 x 2 = 20	10 x 7 = 70
10 x 3 = 30	10 x 8 = 80
10 x 4 = 40	10 x 9 = 90
10 x 5 = 50	10 x 10 = 100

4) Crie um trecho de código while para o problema a seguir: Marcos quer um programa pergunte e some a idade de 20 pessoas.

5) Crie um trecho de código while no qual o usuário deve informar quantos alunos possui e em seguida perguntar a nota final de cada um, realizando a soma das mesmas e por fim apresentar a média das notas da turma.

8.40 ANÁLISE

1) Marque a alternativa em que apresente o resultado da variável acumulador:

```
#include <stdio.h>
int main(){
      int acumulador = 0;
      for(int x = 0; x < 10; x++)
            acumulador = acumulador + x;
      printf("%d", acumulador);
      return 0;
}
```

a. 10
b. 45
c. 9
d. 50

2) Dado o programa abaixo, verifique se o código contém erro(s), se possuir marque a(s) alternativa(s) que informe o número da linha que apresenta o erro:

```
1   #include<stdio.h>
2   int main(){
3   int i = 0;
4   int soma = 0;
5   for( i<= 10; i++){
6   soma = soma + i;
7   i++;
8   }
9   return 0;
10  }
```

a. Não existem erros
b. Linha 5
c. Linha 6
d. Linha 7

3) O programa abaixo deve imprimir ao usuário apenas os valores ímpares, verifique se o código contém erro(s), se possuir marque a(s) alternativa(s) que informe o número da linha que apresenta o erro:

```
1   #include<stdio.h>
2   int main(){
3       for(int i = 0; i<= 10; i - - ){
4           i++;
5           printf(“%d”, i);
6       }
7       return 0;
8   }
```

a. Não existem erros
b. Linha 3
c. Linha 4
d. Linha 5

4) Dado o código abaixo, marque a alternativa que melhor responde essa execução:

```
int x = 0;
while( x == 0 ){
   printf("O que acontece?\n");
   printf(“%d\n”, x);
}
```

a. Imprime na tela:
   O que acontece?
   0
b. Imprime na tela:
   O que acontece? 0
c. Não imprime nada, o código contém erro
d. Imprime na tela infinitamente:
   O que aconte?
   0

5) Dado o programa abaixo, marque a alternativa que corresponde à saída de dados:

```
#include <stdio.h>
    int main( ){
int linha, coluna;
        printf("\n");
linha = 1;
while (linha < 4){
    printf( "\t" );
    coluna = 1;
while (coluna < linha){
   printf( "*" );
   coluna += 1;
}
```

```
            printf( "\n" );
            linha += 1;
        }
        return 0;
}
```

a.
```
****
****
****
****
```
b.
```
*
**
***
****
```
c.
```
****
***
**
*
```
d.
```
*
**
***
```

6) Ao executar o loop abaixo teremos como saída:

```
int x=0;
while(x<=20){
            int soma = soma + x;
            x = x + 2;
}
printf(“%d”, soma);
```

a. 110
b. 210
c. 20
d. 90

7) Levando em conta que o usuário digitou o valor 5 para a variável x, marque a alternativa que melhor apresenta a saída de dados na linha 9 do programa abaixo:

```
1 #include<stdio.h>
2 int main(){
3           int x=10;
4           do{
5                   x++;
6           } while(x=6);
7           printf(“Digite um valor:”);
8           scanf(“%d”, &x);
9           printf(“O valor de x é: %d”, x);
10 }
```

a. O valor de x é 5
b. O valor de x é 10
c. Nunca executar a linha 9 pois entra num loop infinito
d. Não irá executar pois o código possui erro

8.41 SÍNTESE

1) Desenvolva um programa que receba do usuário um valor inteiro positivo e some todos os valores pares de 0 até o valor digitado por ele e conte quantos números ímpares existem nesse intervalo. No final apresente os resultados.

2) Faça um programa que leia valores inteiros até que seja digitado 0 (zero), o programa deve retornar quantos valores pares e quantos valores ímpares foram digitados, bem como o maior e o menor valor.

3) Desenvolva um programa mostre um menu com as 4 operações básicas (soma, subtração, multiplicação e divisão) e pergunte ao usuário que tipo de operação deseja fazer, e com quantos números. O programa deve mostrar o resultado da operação e ter a opção 5 no menu principal para que o usuário saia do programa.

8.42 AVALIAÇÃO

1) Xuxa, a rainha dos baixinhos, criou uma música que tem o seguinte formato:

n patinhos foram passear
Além das montanhas
Para brincar
A mamãe gritou: Quá, quá, quá, quá
Mas só n patinhos voltaram de lá.

Que se repete de 5 até nenhum patinho volte

Marque a alternativa que apresente a melhor resposta para o problema apresentado acima:

a.
```
#include <stdio.h>
int main( ){
   printf("5 patinhos foram passear \nAlém das montanhas \nPara brincar \nA mamãe gritou: Quá, quá, quá, quá \nMas só 4 patinhos voltaram de lá.");
   printf("4 patinhos foram passear \nAlém das montanhas \nPara brincar \nA mamãe gritou: Quá, quá, quá, quá \nMas só 3 patinhos voltaram de lá.");
   printf("3 patinhos foram passear \nAlém das montanhas \nPara brincar \nA mamãe gritou: Quá, quá, quá, quá \nMas só 2 patinhos voltaram de lá.");
   printf("2 patinhos foram passear \nAlém das montanhas \nPara brincar \nA mamãe gritou: Quá, quá, quá, quá \nMas só 1 patinhos voltaram de lá.");
   printf("1 patinho foi passear \nAlém das montanhas \nPara brincar \nA mamãe gritou: Quá, quá, quá, quá \nMas nenhum patinho voltou de lá.");
      return 0;
}
```

b.
```
#include <stdio.h>
int main( ){
      int patinho = 5;
      for(int i = patinho; i > 0; i - - ){
            if( i = = 1)
              printf("%d patinho foi passear \nAlém das montanhas \nPara brincar \nA mamãe gritou: Quá, quá, quá, quá \nMas nenhum patinho voltou de lá.", i);
            else{
              patinho - -;
              printf("%d patinhos foram passear \nAlém das montanhas \nPara brincar \nA mamãe gritou: Quá, quá, quá, quá \nMas só %d patinhos voltaram de lá.", i, patinho);
              }
```

```
        }
        return 0;
    }
```

c.
```
#include <stdio.h>
int main( ){
    for(int i=5; i > 0; i - -){
        witch( i ){
            case 5:
              printf("5 patinhos foram passear \nAlém das montanhas \nPara brincar \nA mamãe gritou: Quá, quá, quá, quá \nMas só 4 patinhos voltaram de lá.");
              break;
            case 4:
              printf("4 patinhos foram passear \nAlém das montanhas \nPara brincar \nA mamãe gritou: Quá, quá, quá, quá \nMas só 3 patinhos voltaram de lá.");
              break;
            case 3:
              printf("3 patinhos foram passear \nAlém das montanhas \nPara brincar \nA mamãe gritou: Quá, quá, quá, quá \nMas só 2 patinhos voltaram de lá.");
              break;
            case 2:
              printf("2 patinhos foram passear \nAlém das montanhas \nPara brincar \nA mamãe gritou: Quá, quá, quá, quá \nMas só 1 patinhos voltaram de lá.");
              break;
            case 1:
              printf("1 patinho foi passear \nAlém das montanhas \nPara brincar \nA mamãe gritou: Quá, quá, quá, quá \nMas nenhum patinho voltou de lá.");
        }
    }
  return 0;
}
```

d. Nenhuma das alternativas

2) Sabe-se que existem 4 tipos sangue (A, B, O, AB) e o hospital X quer um programa que controle as doações. O hospital recebe diariamente 25 doações, logo, marque a alternativa que mostre um programa que apresente quanto cada tipo sanguínio obteve de doação:

a.
```
#include<stdio.h>
int main(){
    int doadoresA, doadoresB, doadoresO, doadoresAB, tipo;
    int x = 1;
    while(x <= 25){
        printf("Digite 1- Tipo A 2- TipoB 3- Tipo O 4- Tipo AB: ");
        scanf("%d", &tipo);
        if(tipo == 1) doadoresA++;
        else if(tipo == 2) doadoresB++;
        else if(tipo == 3) doadoresO++;
        else if(tipo == 4) doadoresAB++;
        else prinf("Tipo sanguíneo inválido!");
        x++;
    }
    printf("No dia de hoje obteve-se: \nTipo A: %d \nTipo B: %d \n TipoO: %d \n Tipo AB: %d", doadoresA, doadoresB, doadoresO, doadores AB);
}
```

b.
```
#include<stdio.h>
```

```
int main(){
        int doadoresA, doadoresB, doadoresO, doadoresAB;
        int x = 1;
        int tipo = 0;
                while(x <= 25){
                        while((tipo<1)||(tipo>4)){
                                printf(“Digite 1- Tipo A 2- TipoB 3- Tipo O 4- Tipo AB: “)
                                scanf(“%d”, &tipo);
                        }
                        if(tipo == 1) doadoresA++;
          else if(tipo == 2) doadoresB++;
          else if(tipo == 3) doadoresO++;
          else (tipo == 4) doadoresAB++;
          x++;
        }
        printf(“No dia de hoje obteve-se: \nTipo A: %d \nTipo B: %d \n TipoO: %d \n Tipo AB:
    %d”, doadoresA, doadoresB, doadoresO, doadores AB);
}
```

c.

```
#include<stdio.h>
int main(){
                int doadoresA, doadoresB, doadoresO, doadoresAB, tipo;
                int x = 1;
                while(x <= 25){
                        printf(“Digite 1- Tipo A 2- TipoB 3- Tipo O 4- Tipo AB: “);
                        if(tipo == 1) doadoresA++;
          else if(tipo == 2) doadoresB++;
          else if(tipo == 3) doadoresO++;
          else if(tipo == 4) doadoresAB++;
          else prinf(“Tipo sanguíneo inválido!”);
          x++;
                }
        printf(“No dia de hoje obteve-se: \nTipo A: %d \nTipo B: %d \n TipoO: %d \n Tipo AB:
    %d”, doadoresA, doadoresB, doadoresO, doadores AB);
}
```

d.

```
#include<stdio.h>
int main(){
                int doadoresA, doadoresB, doadoresO, doadoresAB, tipo;
                int x = 1;
     printf(“Digite 1- Tipo A 2- TipoB 3- Tipo O 4- Tipo AB: “);
     scanf(“%d”, &tipo);
                while(x <= 25){
                        if(tipo == 1) doadoresA++;
          else if(tipo == 2) doadoresB++;
          else if(tipo == 3) doadoresO++;
          else if(tipo == 4) doadoresAB++;
          else prinf(“Tipo sanguíneo inválido!”);
                }
        printf(“No dia de hoje obteve-se: \nTipo A: %d \nTipo B: %d \n TipoO: %d \n Tipo AB:
    %d”, doadoresA, doadoresB, doadoresO, doadores AB);
}
```