UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE CIÊNCIAS NATURAIS E EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM EDUCAÇÃO EM CIÊNCIAS:
QUÍMICA DA VIDA E SAÚDE

Luciana Vescia Lourega

CONJUNTO ESTRUTURADO DE ATIVIDADES DIDÁTICAS PARA O ENSINO INTRODUTÓRIO DE PROGRAMAÇÃO

Santa Maria, RS
2022

Luciana Vescia Lourega

CONJUNTO ESTRUTURADO DE ATIVIDADES DIDÁTICAS PARA O ENSINO INTRODUTÓRIO DE PROGRAMAÇÃO

Tese apresentada ao Programa de Pós-Graduação em Educação em Ciências: Química da Vida e Saúde, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do título de **Doutora em Educação em Ciências**.

Orientador: Ricardo Andreas Sauerwein

Santa Maria, RS
2022

Luciana Vescia Lourega

CONJUNTO ESTRUTURADO DE ATIVIDADES DIDÁTICAS PARA O ENSINO INTRODUTÓRIO DE PROGRAMAÇÃO

Tese apresentada ao Programa de Pós-Graduação em Educação em Ciências: Química da Vida e Saúde, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do título de **Doutora em Educação em Ciências**.

Aprovado em 26 de setembro de 2022

Ricardo Andreas Sauerwein, Dr. (UFSM)
(Presidente/Orientador)

Carmen Vieira Mathias, Drª. (UFSM)

Dioni Paulo Pastorio, Dr. (UFRGS)

Giovani Rubert Librelloto, Dr. (UFSM)

Márcio Marques Martins, Dr. (UNIPAMPA)

Santa Maria, RS
2022

AGRADECIMENTOS

Dedico esta tese a todas as pessoas que me apoiaram durante esse processo. A minha filha, que muitas noites, dormiu no sofá, pois queria estar ao meu lado e não me deixar sozinha, serei eternamente grata pela tua doçura.

Aos meus pais, que sempre me incentivaram a estudar e seguir em frente, independentemente das dificuldades.

Aos meus irmãos, que me incentivam, principalmente meu irmão Rogerio, que me apoiou quando me encontrava desanimada e cansada.

Aos meus amigos que me apoiaram nos momentos difíceis dessa caminhada.

Ao meu esposo que esteve ao meu lado, me auxiliando nos momentos de dificuldades e nos cuidados com nossa filha.

Ao meu orientador que sempre me conduziu com paciência e serenidade, estando ao meu lado e mostrando durante todo o trabalho os melhores caminhos a seguir. Obrigada, professor, você é um exemplo que pretendo seguir.

E, por último, agradeço a Nossa Senhora, que no momento mais difícil dessa caminhada, o falecimento do meu pai, me acolheu e me guiou, me dando a certeza que hoje ele está bem, olhando por todos nós.

RESUMO

CONJUNTO ESTRUTURADO DE ATIVIDADES DIDÁTICAS PARA O ENSINO INTRODUTÓRIO DE PROGRAMAÇÃO

AUTORA: Luciana Vescia Lourega
ORIENTADOR: Ricardo Andreas Sauerwein

Este trabalho foi realizado no âmbito da metodologia *Educational Design Research* (EDR), que busca encaminhar soluções para problemas vivenciados na prática docente. A presente pesquisa tem o propósito de criar, desenvolver, implementar e avaliar uma metodologia de ensino baseada em um conjunto de atividades didáticas (ADS) focadas em fornecer ritmo de estudo e *feedback* qualificado aos alunos da disciplina introdutória de programação no curso técnico em informática integrado ao ensino médio de uma Instituição Federal de ensino. As ADS foram desenvolvidas para realizar um acompanhamento contínuo do desempenho dos alunos, a fim de que o alto índice de evasão e reprovação sejam reduzidos. Durante esse processo metodológico, o professor consegue identificar as lacunas apresentadas pelos alunos e retomar o conteúdo individualmente e/ou em grupo, não deixando que as dúvidas se acumulem e se tornem obstáculos para o aprendizado. O retorno aos alunos foi realizado por meio de uma devolutiva do aprendizado, levando em consideração a compreensão leitora, a abstração, o raciocínio lógico e as noções matemáticas, aspectos que representam os quatro pilares essenciais a serem avaliados no ensino da referida disciplina. Os resultados obtidos indicam que as ADS promovem uma maior participação dos alunos, entendendo que sua realização é importante para que as dificuldades encontradas possam ser sanadas no momento em que forem sendo identificadas, não havendo um acúmulo de dúvidas, o que pode gerar desinteresse e abandono da disciplina. Também foi possível concluir que, ao comparar os métodos nas respectivas avaliações analisadas, os alunos apresentaram um melhor rendimento, principalmente devido ao *feedback* do professor e ao número de avaliações propostas, constatando-se um aumento significativo no número de alunos aprovados e uma acentuada redução no número daqueles em recuperação. Sendo assim, os alunos foram estimulados a participar das atividades propostas, criando um ritmo, concentrando seus esforços nas dificuldades encontradas e apresentando uma evolução em seus conhecimentos.

Palavras-chave: Ensino. Algoritmo. Ritmo de estudo. Atividades didáticas. *Feedback* qualificado.

ABSTRACT

STRUCTURED SET OF TEACHING ACTIVITIES FOR INTRODUCTORY PROGRAMMING

AUTHOR: Luciana Vescia Lourega
ADVISOR: Ricardo Andreas Sauerwein

This work was carried out within the scope of the Educational Design Research (EDR) methodology, which seeks to provide solutions to problems experienced in teaching practice. The present research aims to create, develop, implement and evaluate a teaching methodology based on a set of didactic activities (DAs) focused on providing study rhythm and qualified feedback to students of the introductory programming discipline in the technical course in informatics, integrated to secondary education at the educational Federal Institution. The DAs were developed to carry out a continuous monitoring of student performance, so that the high rate of evasion and failure are reduced. During this methodological process, the teacher is able to identify the gaps presented by the students and resume the content individually and/or in groups, not letting doubts accumulate and become obstacles to learning. Feedback to students was carried out through a feedback of learning, taking into account reading comprehension, abstraction, logical reasoning and mathematical notions, aspects that represent the four essential pillars to be evaluated in the teaching of Programming I. The results obtained indicate that the DAs promote greater student participation, understanding that their realization is important so that the difficulties encountered can be remedied at the moment that they are identified, without an accumulation of doubts which can generate disinterest and abandonment of discipline. It was also possible to conclude that when comparing the methods in the respective evaluations analyzed, the students showed a better performance, mainly due to the teacher's feedback and the number of proposed evaluations, noting a significant increase in the number of approved students and a marked reduction in the number of recovering students. Thus, students were encouraged to participate more and more in the proposed activities, creating a rhythm, focusing their studies on the difficulties encountered and presenting a better evolution in their knowledge.

Keywords: Teaching. Algorithm. Study rhythm. Didactic activities. Qualified feedback.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1. INTRODUÇÃO

O ensino e a aprendizagem de Programação de Computadores apresentam desafios que persistem após 50 anos de pesquisa na área. No Brasil, o trabalho com esse conteúdo ocorre em cursos superiores na área de Computação e em alguns cursos técnicos profissionalizantes, subsequentes ou integrados ao ensino médio. Pereira Junior *et al* (2005) comprovou, através de eventos promovidos nas áreas de Informática e Educação (WEI e SBIE, respectivamente), que a comunidade científica brasileira tem buscado constantemente, propor soluções para os problemas dentro do tema adotado, porém, ainda é pouco se comparado a pesquisas em outras áreas e domínios. Dessa maneira, percebe-se que a área necessita de mais pesquisas e, além disso, é importante observar em qual nível de escolaridade elas estão sendo concentradas.

Segundo Aureliano e Tedesco (2012), ao realizar uma análise de artigos que focam nos processos introdutórios sobre ensino e aprendizagem de programação, 61% das pesquisas abordam estudos no contexto da educação superior e 19% apresentam estudos no contexto dos níveis fundamental, médio e técnico. Esses dados são oriundos de artigos publicados nos anos de 2002 a 2011, nos dois principais eventos nacionais, o Simpósio Brasileiro de Informática na Educação (SBIE) e o Workshop de Informática na Escola (WIE). Dessa forma, os dados apresentados mostram claramente a carência de pesquisas que abordem o processo de ensino e aprendizagem de programação para alunos iniciantes que estejam cursando os níveis fundamental, médio e técnico.

Já no ano de 2013 em uma outra revisão da literatura realizada nos últimos cinco anos (2009 a 2013), também descreve que o principal foco das pesquisas realizada no ensino de programação continua sendo o ensino superir, sendo que nesse mesmo ano as pesquisas tiveram um significativo aumento (SILVA *et al.*, 2013). Após, em outra revisão da literatura publicizada em 2016 (ZANETTI *et al.*, 2016) o ensino médio técnico passa a ter mais estudos, ampliando as pesquisas nesse nível de ensino, onde a carência de pesquisa ainda é considerada alta.

De acordo com as diretrizes curriculares para os cursos de Informática e Computação do MEC, a disciplina de Programação faz parte da área de formação básica em Computação e Informática, juntamente com as matérias de Computação, Algoritmos e Arquitetura de computadores (BRASIL, 2016). A disciplina requer conhecimentos e habilidades básicas relacionadas à escrita, à interpretação de texto, aos conceitos matemáticos, ao raciocínio lógico, à capacidade de abstração e à solução de problemas. A maioria dos estudantes que ingressam no ensino médio tem, em sua formação, muitas lacunas em relação a essas habilidades e isso faz com que o número de reprovações na área seja elevado, assim como eleva o índice de

abandono. Contudo, Rodrigues (2004) relata que essa disciplina tem um dos maiores índices de reprovação em todas as instituições de ensino brasileiras, o que a torna ponto de reflexão por parte de professores preocupados com a melhoria da qualidade no processo de ensino e aprendizagem dos alunos, ratificando a necessidade de alterações didáticas e metodológicas de apresentação.

A inserção de novas metodologias relacionadas ao ensino e à aprendizagem de programação tem sido muito estudada e aplicada nos últimos anos. Sendo assim, constata-se a necessidade de abandonar a forma tradicional de ensino, na qual o conteúdo é abordado de forma expositiva e se concentra apenas na solução de problemas e em avaliações baseadas quase que exclusivamente em notas parciais e finais. Conforme Pereira Junior e Rapkiewicz (2004), no modo tradicional de ensino, o professor explana o conteúdo através de uma ferramenta[1] para desenvolver os algoritmos, apresenta alguns exemplos e propõe exercícios para a turma. Segundo Berssanette (2016), o modo tradicional não é o suficiente para motivar os alunos a se interessarem pela disciplina, pois, com essa metodologia, não fica clara a importância de certos conteúdos para sua formação.

> O ensino de Algoritmos na maioria das Instituições se dá através de aulas convencionais e essas não privilegiam o que se faz necessário para a aprendizagem desse conteúdo. São necessárias ações pedagógicas que contemplem esse processo, ou seja, são necessárias ações pedagógicas voltadas à aprendizagem para a Resolução de Problemas que permitem desenvolver capacidades como atenção, concentração, pensamento criativo e outras habilidades perceptuais psicomotoras indispensáveis para agilizar o raciocínio (RAPKIEWICZ *et al*., 2006, p. 4).

Nesse contexto, é importante que o professor busque estratégias e metodologias de ensino que possam auxiliar e contribuir com a aprendizagem dos estudantes, a fim de facilitar a compreensão sobre os conteúdos de programação. Nesse contexto, diversos métodos, metodologias, ferramentas, práticas e técnicas adotadas têm sido descritos no ensino de programação, relatando a preocupação de professores em auxiliar seus alunos, através de alternativas para melhor apoiá-los em sala de aula (Berssanette (2016); Zacarias e Melo (2019)). No entanto, apesar de todas as estratégias e/ou metodologias criadas, o processo de ensino e aprendizagem da disciplina de programação de computadores ainda tem encontrado muitas dificuldades (ARIMOTO e OLIVEIRA, 2019).

Professores preocupados em auxiliar seus alunos buscam alternativas para melhor apoiá-los em sala de aula. Na maioria das vezes, os professores têm dificuldades em identificar/reconhecer as habilidades prévias dos alunos de forma a aproveitá-las melhor

[1] Ferramentas utilizadas por professores para o ensino e aprendizagem inicial de programação, tendo como base a língua portuguesa. Como exemplo, cita-se o Portugol, Visualg, entre outras.

(SCHULTZ, 2003) e isso dificulta tanto aos alunos quanto aos professores a voltarem seus esforços paras as lacunas apresentadas durante o desenvolvimento da disciplina, sendo importante que o docente busque estratégias e metodologias de ensino que possam auxiliá-lo a contribuir para a aprendizagem dos estudantes, a fim de facilitar a compreensão dos conteúdos.

Uma das preocupações atuais na educação é saber como ensinar e como avaliar. Com isso, orienta-se-para o uso de metodologias que considerem o ensino-aprendizagem baseado em competências e habilidades (BRASIL, 2018). As competências e habilidades relativas à formação em Computação para a Educação no Ensino Médio estão inseridas na Base Nacional Comum Curricular (BNCC) e contribuem para o desenvolvimento de suas competências gerais, estando organizadas em três eixos: pensamento computacional[2], mundo digital[3] e cultura digital[4].

Desse modo, com a implementação da metodologia baseada no desenvolvimento das habilidades dos alunos, deseja-se melhor entender e acompanhar suas dificuldades de maneira mais próxima, auxiliando-os no processo de aprendizagem e, consequentemente, preparando-os para os desafios do mundo atual (BRASIL, 2018). As pesquisas nessa área estabeleceram um consenso de que a programação é uma atividade altamente complexa que envolve subtarefas ligadas a diferentes domínios do conhecimento e a uma variedade de processos cognitivos (AMBRÓSIO *et al*., 2011). Assim, valoriza-se um conjunto de habilidades que consolidam, aprofundam e ampliam a formação integral (BRASIL, 2018). Dentre as diversas habilidades que a área de programação exige, foi identificado um conjunto de quatro habilidades para serem desenvolvidas e avaliadas — a compreensão leitora, a abstração, o raciocínio lógico e as noções matemáticas —, sendo algumas delas citadas em outros trabalhos (AMBRÓSIO *et al*., 2011).

A fim de desenvolver uma proposta diferenciada e que supra as dificuldades encontradas no dia-a-dia da sala de aula, este trabalho se propõe a elaborar, implementar e avaliar um conjunto de atividades didáticas (ADs) que visam desenvolver, no aluno, um ritmo de estudo que o auxilie em suas dúvidas no momento em que cada conteúdo está sendo ministrado, não acumulando lacunas em seu aprendizado. Assim, a proposta a ser construída conduz e orienta

[2] Envolve as capacidades de compreender, analisar, definir, modelar, resolver, comparar e automatizar problemas e suas soluções, de forma metódica e sistemática, por meio do desenvolvimento de algoritmos (BRASIL, 2018).
[3] Envolve as aprendizagens relativas às formas de processar, transmitir e distribuir a informação de maneira segura e confiável em diferentes artefatos digitais, compreendendo a importância contemporânea de codificar, armazenar e proteger a informação (BRASIL, 2018).
[4] Envolve aprendizagens voltadas a uma participação mais consciente e democrática por meio das tecnologias digitais, o que supõe a compreensão dos impactos da revolução digital e dos avanços do mundo digital na sociedade contemporânea (BRASIL, 2018).

o estudante até o final da disciplina, demonstrando aos alunos a importância do entendimento de cada etapa ao longo do ano.

A disciplina introdutória de programação possui um rol de assuntos que interagem entre si. Os assuntos a serem abordados possuem uma sequência importante a ser seguida, sendo a compreensão contínua do conteúdo um fio condutor imprescindível do conhecimento, pois se os alunos não compreenderem um tópico, o aprendizado do próximo conteúdo é comprometido, acarretando em desânimo ou até mesmo em desistência da disciplina. Sendo assim, as ADs que compõem a proposta foram elaboradas em conformidade aos conteúdos programáticos a serem ministrados durante o ano. O professor ministra o conteúdo, socializa uma lista de problemas para serem desenvolvidos, corrige e, após, aplica a AD específica do conteúdo abordado. Em seguida, o professor os corrige, levando em consideração critérios[5] que foram desenvolvidos especificamente para analisar se o discente conseguiu compreender o assunto ministrado, proporcionando, caso necessário, uma retomada nas lacunas encontradas no desenvolvimento da AD.

Com esse processo sendo repetido a cada vez que os conteúdos forem sendo desenvolvidos, pretende-se: (1) buscar com que o aluno perceba a necessidade de desenvolver as tarefas proporcionadas pelo professor; (2) auxiliar os discentes nas lacunas apresentadas logo após o conteúdo ser desenvolvido, não acumulando dúvidas que possam comprometer o aprendizado do próximo assunto a ser trabalhado.

1.1.PROBLEMA DE PESQUISA E OBJETIVOS

Devido ao alto índice de reprovação e evasão na disciplina de Programação I do curso Técnico em Informática Integrado ao Ensino Médio de uma Instituição Federal de ensino onde a pesquisa foi desenvolvida, várias metodologias têm sido aplicadas, como o uso da gamificação e a sala de aula invertida, técnicas que já publicitadas pela comunidade científica. Apesar dos esforços, não tem sido observado nenhuma melhora no aproveitamento dos discentes na disciplina de Programação I. Analisando as dificuldades, percebe-se que os alunos não possuem uma conduta de estudos diários, nem sempre desenvolvem as listas de problemas propostas e não esclarecem suas dúvidas. Com isso, eles acabam não ampliando ou melhorando as habilidades básicas necessárias a um programador. Diante desse cenário, surgiu a ideia de desenvolver uma nova abordagem pedagógica, na qual professores e alunos possam

[5] Os critérios criados que irão balizar a correção das atividades estão relacionados com o estudo específico de cada comando e as habilidades (compreensão leitora, abstração, raciocínio lógico e noções matemáticas) necessárias ao aluno para realizar o estudo de programação.

acompanhar o processo de ensino no decorrer da disciplina, baseando-se em habilidades necessárias requeridas e, consequentemente, adquirindo as competências desejadas. Mas como fazê-lo? Como o professor pode mostrar aos discentes a importância em praticar diariamente o desenvolvimento de programas? A fim de delimitar a proposta de trabalho e mostrar os caminhos percorridos para o desenvolvimento da pesquisa, abaixo serão descritos a problemática proposta e os respectivos objetivos gerais e específicos para solucioná-la.

1.1.1 Problema de pesquisa

Como melhorar o desempenho e diminuir o índice de reprovação dos alunos na disciplina Programação I em um curso técnico integrado ao ensino médio?

1.1.2 Objetivo geral

Desenvolver, aplicar e investigar um conjunto de ADs na forma de proposição, análise e avaliação das tarefas, baseadas nas habilidades: abstração, interpretação, raciocínio lógico e noções de matemática, nos conceitos de Ausubel e *Education Design Research*, a fim de reconhecer como influenciam no desempenho dos estudantes na disciplina de Programação I.

1.1.3 Objetivos específicos

Constituem-se como objetivos específicos desta proposta:

1) construir AD na forma de tarefa para cada tópico da ementa da disciplina, para que todo o conteúdo seja contemplado na íntegra;
2) elaborar um conjunto de tarefas distribuídas ao longo do ano letivo para introduzir os comandos básicos de programação utilizando a linguagem C;
3) elaborar um conjunto de questões no final de cada AD, a fim de verificar a eficácia das ADs em relação a sua viabilidade didática durante o seu processo de desenvolvimento;
4) desenvolver um questionário geral, a fim de que os alunos possam avaliar todo o conjunto de AD desenvolvido durante o ano letivo;
5) desenvolver uma matriz de análise da produção dos alunos em cada tarefa, no intuito de acompanhar o desenvolvimento dos alunos;
6) analisar a evolução dos alunos por meio das habilidades investigadas;
7) comparar a metodologia proposta neste trabalho com a metodologia utilizada nos anos de 2019 e 2020.

1.2 ESTRUTURAÇÃO DA TESE

O presente trabalho está estruturado em seis capítulos. O primeiro capítulo, já descrito acima, traz uma pequena introdução sobre o tema abordado neste trabalho e apresenta uma justificativa, bem como o problema de pesquisa e os objetivos traçados na sua construção.

No capítulo 2, será realizada uma revisão da bibliografia, que traz o contexto histórico referente às linguagens de programação, às dificuldades encontradas na área e a algumas estratégias de ensino e aprendizagem utilizadas no ensino da disciplina de Programação I, assim como serão expostas as principais metodologias de ensino-aprendizagem utilizadas nos últimos anos.

No capítulo 3, será descrito o conceito da *Educational Design Research* (EDR), suas características e, sua importância no processo da pesquisa educacional.

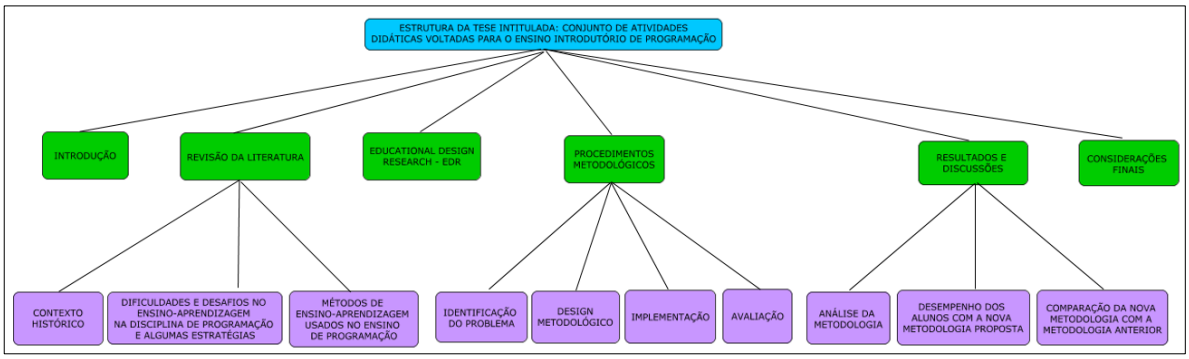
No capítulo 4, serão apresentados os procedimentos metodológicos que balizaram a construção desta pesquisa, utilizando, para isso, as fases que compõem a EDR.

No capítulo 5, serão descritos e discutidos os resultados referentes aos dados coletados.

Para finalizar, no capítulo 6, serão apresentadas as considerações finais, juntamente com sugestões para que futuras pesquisas possam ser desenvolvidas dando sequência à proposta deste trabalho.

A seguir, será apresentada, por meio da Figura 1, a organização desta tese, possibilitando observar claramente a estrutura seguida neste trabalho.

Figura 1 – Organização da tese



Fonte: Da Autora (2020).

2 REVISÃO DA LITERATURA

Neste capítulo, inicialmente, será descrito, de maneira geral e resumida, um contexto histórico de programação e suas linguagens, destacando a importância na evolução da sociedade. Posteriormente, as dificuldades e os desafios no processo de ensino-aprendizagem serão relatados e, por fim, serão conhecidas as metodologias usadas, desde as tradicionais, até as metodologias ativas atuais, sendo que o Método EDR, por ser o método usado neste estudo, será detalhado em um capítulo à parte.

2.1 CONTEXTO HISTÓRICO

A programação é uma prática antiga, que iniciou no século XIX com Joseph-Marie Jacquard. Em 1801, Jacquard criou um tear que podia tecer a partir de um padrão de cartões de madeira perfurados e conectados por cordões lidos automaticamente, gerando padrões decorativos (COSTA, 2008). Em 1821, Charles Babbage iniciou o projeto com uma máquina diferencial: uma máquina de cálculos financiada pelo governo da Inglaterra. Como o projeto não deu certo, ele criou a máquina analítica: o primeiro computador mecânico. Posteriormente, em 1842, Ada Augusta Bryon foi responsável por desenvolver o primeiro programa para a máquina analítica de Babbage, sendo considerada a primeira programadora da história (COSTA, 2008). Em 1890, Herman Hollerith projetou a mesa de Hollerith, uma máquina que funcionava com cartões perfurados e serviu para realizar o censo da população americana (VILLAÇA, 2014).

No início do século XX, Alan Turing formalizou o conceito de algoritmo e computação através de um modelo teórico e formal chamado de máquina de Turing. Assim, formulou a tese de que qualquer modelo de computação prático ou tem capacidade equivalente a de uma máquina de Turing ou tem um subconjunto dela (PIRES *et al*, 2019). Anos mais tardes, em 1940, surgem as primeiras linguagens de computador e Konrad Zuse destaca-se, pois projetou os computadores da série ‘Z’ e criou a linguagem de programação conhecida como Plankalkül entre os anos de 1943 a 1945 (SEBESTA, 2018). Posteriormente, surgiu o Assembly, em 1947, e, na sequência, outras linguagens importantes foram inventadas na década de 50, tais como: Fortran, Algol, Cobol e Lisp nos anos de 1954, 1958, 1959 e 1958, respectivamente. A partir daí as linguagens foram evoluindo e, em 1964, foi criada a linguagem Basic, por Eugene Kurtz e George Kemeny. Essa linguagem teve uma grande aceitação e foi uma das principais, a qual permitiu o

acesso da população aos computadores, pois, até então, apenas cientistas da computação tinham acesso (DANTAS, 1989). Na década de 1970, ocorreu o desenvolvimento das linguagens Pascal, no ano de 1971, e C, no ano de 1972. Nessa mesma década surgiram outras linguagens, como Scheme, Prolog, Ada e Forth, nos anos de 1975, 1972, 1979 e 1970, respectivamente.

A partir dos anos 80, as empresas buscaram investir na melhoria das linguagens já criadas nas décadas de 60 e 70. Dentre alguns exemplos, destacam-se as linguagens C++, uma evolução da linguagem C, com nível intermediário, orientada a objetos, e a Objective-C, em 1983 (SEBESTA, 2018). Na mesma época, tivemos a criação das seguintes linguagens: Miranda, em 1982, Common Lisp, em 1984, Object Pascal, em 1985, Eiffel, em 1986, Haskell 1.0, em 1987 e Perl, em 1987.

Na década de 90, as modernas linguagens de programação vieram com um foco inovador para a rede mundial de computadores e para a orientação a objetos, tais como: as linguagens Python e Java, em 1991, e PHP, em 1995. Na mesma década, foram criadas as linguagens Ruby e Lua, em 1993, e JavaScript e Delphi, em 1995 (SILVA, 2016). A partir dos anos 2000, continuamos a ver novas importantes linguagens serem desenvolvidas, como o Visual Basic.Net, em 2000, C#, em 2001, Scala, em 2003, JavaFx, em 2007, e Go, em 2003. O desenvolvimento de cada uma dessas linguagens está atrelado aos cenários, que, conforme os períodos, foram se modificando e, se algumas ideias não fossem compatíveis com determinada plataforma, novas linguagens eram criadas.

Entretanto, com o passar do tempo, o computador pessoal foi ficando mais acessível aos usuários e as linguagens de programação passaram a privilegiar cada vez mais a interoperacionalidade, o que levou às linguagens padronizadas. Atualmente, as novas linguagens surgem principalmente em torno de questões ainda não bem atendidas pelas linguagens estabelecidas, de modo que, conforme o surgimento de novas demandas da sociedade, novas linguagens são criadas. Com isso, diante do desenvolvimento dos computadores e de suas necessidades, os programadores foram sendo desafiados a aprender e desenvolver habilidades necessárias para associar a linguagem binária e literal da máquina à linguagem complexa dos seres humanos.

2.2 DIFICULDADES E DESAFIOS NO ENSINO-APRENDIZAGEM NA DISCIPLINA DE PROGRAMAÇÃO E ALGUMAS ESTRATÉGIAS

O ensino e aprendizagem de programação e suas complexidades têm sido tema de vários trabalhos nacionais e internacionais. Há muito tempo, procura-se encontrar alternativas que auxiliem o aluno a compreender conceitos relacionados à programação de computadores (MORAES; NETO; OSÓRIO, 2020). Ensinar programação não é somente transmitir aos estudantes comandos e instruções que o computador deverá executar, trata-se de um processo complexo, que envolve lógica, raciocínio e habilidade por parte do aluno no conhecimento do problema a ser resolvido e na elaboração da solução para ele, muitas vezes limitando-se a uma esfera teórica (GOSSMANN, 2017).

Programar é uma tarefa desafiadora e, para amenizar as dificuldades no seu aprendizado, diversas abordagens metodológicas têm sido adotadas para seu ensino em cursos de graduação, técnico e subsequente. Existem muitos relatos de experiência, como os trazidos por Diemer *et al.* (2018) e Pereira (2019) que oferecem cursos de programação para quem não tem experiência anterior, mas ainda não se tem dados precisos sobre qual a melhor metodologia a ser empregada e qual o sucesso real desses cursos para os estudantes. De acordo com Berssanette (2016), uma das dificuldades de ensino da disciplina está justamente relacionada à ausência de metodologias adequadas, pois geralmente é ministrada de maneira tradicional, apenas com a teorização.

Segundo Ambrósio *et al.*, (2011) e com base na BNCC, as pesquisas nessa área consensualizaram que a programação é uma atividade altamente complexa que envolve subtarefas ligadas a diferentes domínios do conhecimento e a uma variedade de processos cognitivos. Assim, valoriza-se um conjunto de habilidades que, conforme os autores, podem se sobrepor, apenas refletindo diferentes designações usadas. Dentre as designações citadas, foi identificado um conjunto de quatro habilidades: a compreensão leitora, a abstração, o raciocínio lógico e as noções matemáticas. Seus conceitos são apresentados a seguir:

- **Compreensão leitora:** segundo Coll (1990 apud MIRANDA, 2016), a leitura é o processo no qual o leitor realiza um trabalho ativo de compreensão e interpretação do texto a partir de seus objetivos, decodificando e analisando as sentenças presentes no corpo textual. A compreensão é o momento no qual você percebe quais são as principais palavras de cada ideia e quais são os dados e as informações possíveis de serem coletadas.
- **Abstração:** para Souza (2015), a abstração geralmente procura isolar os elementos das coisas e de suas relações para poder compreendê-los melhor.

- **Raciocínio lógico:** segundo Menzies (1996 apud BRESOLIN, 2015), raciocínio lógico é um processo estruturado de pensamento, bem como a aprendizagem do conteúdo de um conceito em toda sua extensão, o que permite chegar a uma determinada conclusão ou à resolução de um problema.
- **Noções matemáticas:** conforme Moura (2007), aprender matemática não é só aprender uma linguagem, é adquirir também modos de ação que possibilitam lidar com outros conhecimentos necessários à sua satisfação, às necessidades de natureza integrativa, a fim de construir solução para problemas tanto do indivíduo quanto do coletivo.

Dentre várias habilidades contempladas e desejadas na BNCC para os cursos de Informática, as habilidades citadas acima são essenciais para que os alunos tenham êxito no processo de aprendizagem da disciplina que compõe o ensino inicial de programação. Sendo assim, é necessário que, junto com os conteúdos, sejam criadas situações para o desenvolvimento de habilidades. É importante ressaltar que um aluno, ao desenvolver as habilidades seguindo orientações de um educador, vai aprender a usá-las de maneira adequada e conveniente.

Neste trabalho, tais habilidades foram identificadas como pilares que sustentam esse ensino. Esses pilares descritos já foram identificados por vários autores, como Falkembach *et al.* (2003), que descreveram a dificuldade de interpretação do próprio problema muito antes da dificuldade de interpretação de algum tipo de representação. Outros autores preconizam a necessidade de domínio de habilidades matemáticas prévias que sejam ao menos integradas (HENDERSON, 1986) ou desejáveis (KOLIVER; DORNELES; CASA, 2004). Já para Baeza-Yates (1995), o próprio nível de conhecimento prévio de lógica matemática é discutido. Além disto, Koliver, Dorneles e Casa (2004) afirmam que a apresentação de princípios básicos da lógica é suficiente para a resolução da maior parte dos problemas propostos para uma disciplina de nível introdutório, enquanto Nobre e Menezes (2002 apud SCHULTZ, 2003) descreveram as dificuldades em trabalhar a capacidade de abstração do aluno, seja na busca de possíveis soluções, seja na escolha das estruturas de dados. A necessidade de o professor fazer o aluno compreender a abstração envolvida com toda simbologia utilizada é corroborada por Rodrigues (2002).

Para Lima *et al*. (2019, p. 2),

> diferentemente de alguns conteúdos, como os relacionados à Biologia, à Matemática e ao Português, a disciplina de algoritmos[6] aborda um conhecimento difícil de se fazer

[6] Algoritmo é uma sequência de passos lógicos necessários para executarmos uma tarefa. Em outras palavras, é como se fosse um passo a passo para resolver um problema, com instruções simples e exatas.

> presente na grade curricular do ensino básico, principalmente em escolas públicas. Dessa forma, muitos alunos que saem diretamente do ensino fundamental e ingressam nos cursos de informática, nunca ouviram falar sobre o que é, e do que se trata a construção de um algoritmo, dificultando seu desempenho na disciplina. Os alunos percebem a disciplina e o conteúdo visto como algo distante da sua realidade e de difícil compreensão.

Quando o aluno se depara com um novo conhecimento que não fez parte de sua formação inicial e que exige conhecimentos específicos e integrados, ele tem a sensação de uma grande frustração. Dessa maneira, para aqueles que possuem lacunas na formação, atender a essas demandas é um grande desafio. Segundo Branco e Schuvartz (2007), os cursos da área de computação e informática enfrentam um grande problema com as disciplinas de introdução à programação de computadores, as quais visam ensinar como utilizar o computador para solucionar problemas. Acadêmicos iniciantes, ao se depararem com a disciplina, sentem-se incapazes de programar, devido ao conjunto de habilidades que a programação exige, bem como a capacidade para solucionar problemas através do raciocínio lógico, da habilidade matemática, da capacidade de abstração, entre outras.

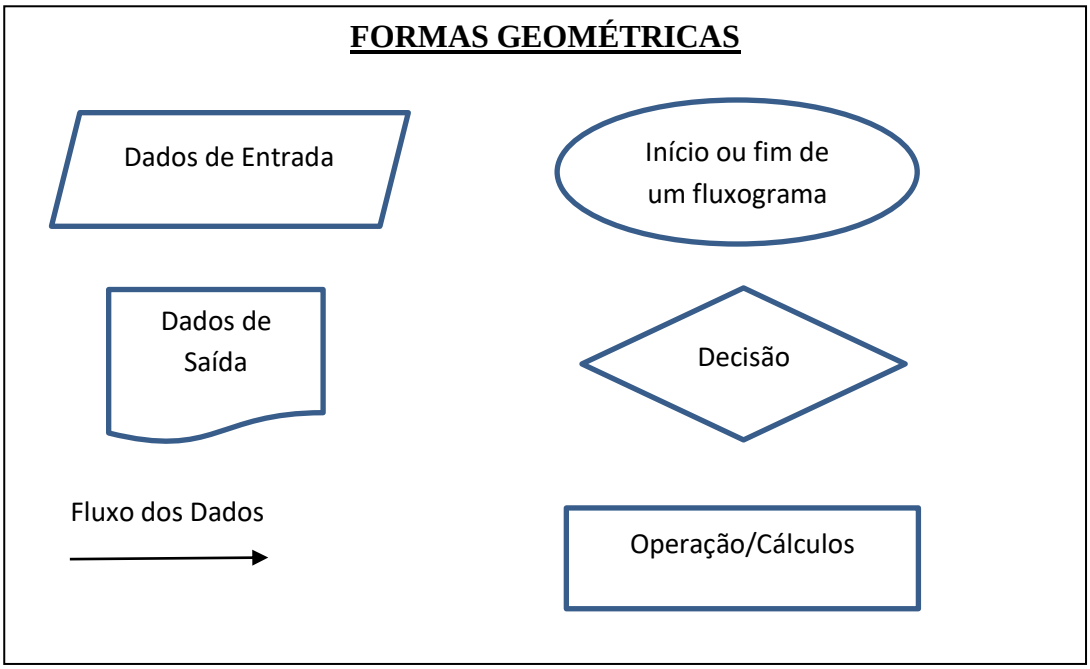
Na maioria das vezes, os professores têm dificuldades em identificar ou reconhecer tais habilidades prévias nos alunos de forma a aproveitá-las melhor (SCHULTZ, 2003) e isso dificulta tanto os alunos quanto os professores a focarem seus esforços nas lacunas apresentadas durante o desenvolvimento da disciplina. Contudo, é importante que o professor busque estratégias de ensino que possam auxiliá-lo a contribuir com a aprendizagem dos estudantes, em busca de facilitar a compreensão sobre os conteúdos de programação.

Uma das estratégias utilizadas no ensino e aprendizagem de programação é o desenvolvimento de algoritmos por fluxogramas. Segundo Slack *et al.* (1997), um fluxograma é uma técnica de mapeamento que permite o registro de ações de algum tipo e pontos de tomada de decisão que ocorrem no fluxo real. De acordo com Tavares (2007, p. 75),

> ele organiza a informação de uma maneira linear. Ele é utilizado para mostrar o passo a passo de determinado procedimento, e normalmente inclui um ponto inicial e outro ponto final. Um fluxograma é normalmente usado para melhorar a performance de um procedimento.

Ainda seguindo o pensamento de Tavares (2007), fluxogramas são fáceis de ler, pois as informações estão organizadas de uma maneira lógica e sequencial. Em informática, um fluxograma pode ser visto como um diagrama para a representação de um algoritmo. Para compor um diagrama é necessário que o aluno saiba reconhecer as formas geométricas que a compõe. A Figura 2 mostra as principais formas utilizadas no desenvolvimento dos fluxogramas.

Figura 2 – Formas geométricas utilizadas em um fluxograma.



Fonte: Da Autora (2020).

Ao unir essas formas geométricas, os alunos podem resolver os problemas lógicos com maior facilidade, percebendo a construção do algoritmo de uma maneira mais clara e objetiva. Com ele, é possível perceber as etapas do seu raciocínio no sentido de mapeá-las para expor suas percepções e abstrações do tema explorado. Os caminhos percorridos pelo aluno ficam perceptíveis a todos e a ele mesmo, como uma concretização do aprendizado em si. O discente se reconhece na sua aprendizagem, ele percebe os caminhos que levaram à sua tomada de consciência para o conhecimento, tornando-o significativo. Segundo Halley *et. al* (2008, p. 2),

> o uso de fluxogramas se deve a três razões principais. Primeiro: fluxogramas possuem uma sintaxe mínima. Quando se reduz o foco em sintaxe, pode-se aumentar o esforço em análise. Segundo: fluxogramas é uma representação universal. Nenhum outro sistema visual alcançou a aceitação dos fluxogramas. Terceiro: fluxogramas são mais fáceis para estudantes iniciantes em computação do que estrutura de código.

O estudo realizado por Crews e Ziegler (1998) verificou que estudantes iniciantes em algoritmos cometem menos erros, têm maior confiança e resolvem problemas representados por algoritmos simples e médios mais rapidamente quando utilizam fluxogramas. Já na pesquisa realizada por Scanlan (1989), com 554 alunos, verificou-se que ao utilizarem fluxogramas, os alunos obtiveram: um tempo menor para a compreensão, menor número de erros em algoritmos complexos, ganho na confiança e queda no tempo para resoluções dos algoritmos. Posteriormente, Medina e Fertig (2005) vão ao encontro desses resultados, pois apontam que fluxogramas são bastante úteis para representações de algoritmos em um nível alto de abstração.

Helley *et al*. (2008) analisaram os estudos de Scanlan (1989), Crews e Ziegler (1998) e Da Silva (2001) e concluíram que a melhor forma de assimilar um conteúdo é por meio de visualizações gráficas. Nesse contexto, o aluno pode fazer anotações em forma visual (desenhos, setas, observações, comentários) de tudo aquilo que acredita ser importante para, posteriormente, poder relembrar facilmente.

Assim, percebe-se que, por meio dos fluxogramas, as informações ficam mais organizadas de forma a mostrar ao aluno a sequência de seu raciocínio. Em programação, os fluxogramas facilitam a abordagem inicial do ensino, fazendo com que, aos poucos, o aluno adquira as habilidades necessárias para desenvolver os problemas propostos, sem se desmotivar no início da disciplina, como ocorre na maioria das vezes, pois o aluno se depara com a cobrança de vários conceitos, não conseguindo construir um caminho para a resolução de seus algoritmos. Após o aluno já ter um primeiro contato com o ensino de programação e, por meio dos fluxogramas, realizar suas resoluções, foi necessário pensar qual seria a melhor estratégia para que esses fluxogramas se tornassem linhas de código.

Há muito tempo, a ideia de ensinar programação passa pela etapa inicial de apresentar pseudocódigos aos alunos. Conforme Aguilar (2011, p. 61), o pseudocódigo é uma linguagem de especificação (descrição) de algoritmos. Goodrich e Tamassia (2008, p. 23) complementam que o pseudocódigo “é uma mistura de linguagem natural e estruturas de programação de alto nível usada para descrever as ideias principais da implementação genérica de uma estrutura de dados ou algoritmo”.

Com o passar do tempo o pseudocódigo desenvolvido em papel pelos alunos passou a ser desenvolvido em ambientes construídos especificamente para compilar códigos escritos na linguagem corrente, usualmente chamado de Portugol. Esses ambientes foram sendo aperfeiçoados e foram ficando cada vez mais parecidos com as linguagens de alto nível, apresentando conceitos como declarações de variáveis, linha de código, palavras-chave, comentários, dentre outros. No Quadro 1, é apresentado um comparativo dos principais tipos de dados e instruções entre uma linguagem de programação de alto nível, no caso a linguagem C, e a mesma instrução em pseudocódigo.

Quadro 1 – Comparativo entre instruções de linguagem de alto nível e o pseudocódigo

Instrução em Linguagens de Alto Nível	Instrução em Pseudocódigo
int	inteiro
float	real
char	raracter
printf	escreva
scanf	leia
for	para
while	enquanto
do...while	faça...enquanto
if	se
switch/case	escolha...caso

Fonte: (GOSMANN, 2017).

Observa-se, no Quadro 1, que as instruções são muito parecidas e que a estrutura montada em um ambiente desenvolvido para compilar pseudocódigos é muito próxima às instruções de uma linguagem de programação de alto nível. Diante dessa situação, optou-se por não mais iniciar o ensino de programação utilizando os pseudocódigos, mas usar diretamente uma linguagem de alto nível, a linguagem C.

Com o uso de uma linguagem de alto nível, o aluno passa a contar com o compilador a fim de verificar se o seu programa está sintática e semanticamente correto. Ele pode realizar tentativas para verificar se seu algoritmo está correto, sendo que essas tentativas de compilar o programa, faz com que o aluno utilize o método de tentativa e erro, fazendo um mau uso do compilador.

No intuito de detectar esse vício de aprendizagem e poder auxiliar o aluno a não utilizar esse tipo de resolução é que foram propostas atividades em que ele precisa explicar o programa desenvolvido e como estruturou seu pensamento para resolver o problema proposto. Além disso, muitas vezes, as resoluções desenvolvidas pelos alunos são apresentadas e debatidas em sala de aula e, caso o discente tenha utilizado essa metodologia para realizar a resolução de seus algoritmos, terá dificuldade em explicar seu código aos colegas. Percebe-se que essa maneira não é adequada e não auxilia no aprendizado.

Outra estratégia que merece destaque é o ensino de programação relacionado aos conceitos da Teoria da Aprendizagem Significativa. O Aprendizado Significativo surgiu como

um dos conceitos principais da teoria de David Ausubel. Nela, o aprendizado do aluno está relacionado com os conceitos que ele já conhece, ou seja, com seu conhecimento prévio. Para Ausubel, esses conhecimentos prévios são definidos como conceito subsunçor[7] (BARBOSA, 2011). Em termos simples, subsunçor é o nome que se dá a um conhecimento específico, existente na estrutura de conhecimentos do indivíduo, que permite dar significado a um novo conhecimento que lhe é apresentado ou que é por ele descoberto (MOREIRA, 2011).

Para Berssanette (2016, p. 40),

> essa teoria preconiza que o conhecimento se organiza em estruturas cognitivas, que são conjuntos de conhecimentos que o indivíduo possui sobre um determinado assunto. A aprendizagem torna-se significativa quando os conhecimentos anteriores são inter-relacionados ao novo conteúdo a ser estudado o qual passa a ser incorporado às estruturas de conhecimento, adquirindo significado especial. Este conceito não consiste numa simples associação e sim uma interação relevante entre os conhecimentos que se possui e os que será conhecido.

Para que a Aprendizagem Significativa ocorra é preciso entender o processo de modificação do conhecimento, ao invés do comportamento em um sentido externo e observável, e reconhecer a importância que os processos mentais têm nesse desenvolvimento. As ideias de Ausubel também se caracterizam por basearem-se em uma reflexão específica sobre a o ensino e a aprendizagem escolar, ao invés de tentar somente generalizar e transferir a aprendizagem, conceitos ou princípios explicativos extraídos de outras situações ou contextos. Com isso, duas condições são necessárias para haver Aprendizagem Significativa, sendo elas: (1) o material de aprendizagem deve ser potencialmente significativo e (2) o aprendiz deve apresentar uma pré-disposição para aprender.

A primeira condição implica que o material de aprendizagem tenha significado lógico (isto é, seja relacionável de maneira não-arbitrária e não-literal a uma estrutura cognitiva apropriada e relevante) e que o aprendiz tenha, em sua estrutura cognitiva, ideias-âncora relevantes com as quais esse material possa ser relacionado. Enquanto a segunda condição implica que, independentemente de quão potencialmente significativo seja o material a ser aprendido, se a intenção do aprendiz for simplesmente de memorizá-la, arbitrária e literalmente, tanto o processo de aprendizagem quanto seu produto serão mecânicos (ou automáticos) (MOREIRA, 2019).

Dessa forma, Ausubel (2000) contrasta a Aprendizagem Significativa com a aprendizagem mecânica, na qual as novas informações adquiridas têm pouca ou nenhuma interação com conceitos relevantes existentes na estrutura cognitiva. Nesse caso, a nova

[7] A palavra “subsunçor” não existe em português; trata-se de uma tentativa de aportuguesar a palavra inglesa “subsumer”. Seria, mais ou menos, equivalente a inseridor, facilitador ou subordinador.

informação é armazenada de maneira arbitrária, não havendo interação entre a nova informação e a informação já existente. Assim, o conhecimento adquirido fica arbitrariamente distribuído na estrutura cognitiva, sem se ligar aos conceitos subsunçores específicos e, muitas vezes, sem fazer sentido para o aluno (MOREIRA, 2019).

Conforme Brezolin (2015, p. 35 apud AUSUBEL, 2000), a Aprendizagem Significativa apresenta algumas vantagens sobre a aprendizagem mecânica:

> - O conhecimento assimilado de maneira significativa é fixado e recordado por mais tempo.
> - Habilidades para aprender outros conceitos relacionados de uma maneira mais simples, mesmo que a informação seja esquecida.
> - Auxiliar na aprendizagem mesmo que a informação seja esquecida.

Por suas características e vantagens, a Aprendizagem Significativa deveria ser mais utilizada com nossos alunos. No entanto, esses estudantes passam anos de suas vidas estudando de acordo com a aprendizagem mecânica, aquela praticamente sem significado, puramente memorística, informações que são memorizadas para as provas e logo depois, são esquecidas. Assim como Ausubel fala em subsunçores, cada teoria construtivista tem seu construto básico. Para Rogers (1998), a Aprendizagem Significativa é aquela que representa algum significado para o aluno, seja na conduta, em uma futura orientação ou em suas atitudes e personalidade.

Deste modo, a estrutura cognitiva da Aprendizagem Significativa, considerada como uma estrutura de subsunçores inter-relacionados e hierarquicamente organizados, é uma estrutura dinâmica caracterizada por dois processos principais, a diferenciação progressiva e a reconciliação integradora.

Para Moreira (2010), "a diferenciação progressiva é o processo de atribuição de novos significados a um dado subsunçor (um conceito ou uma proposição, por exemplo) resultante da sucessiva utilização desse subsunçor para dar significado a novos conhecimentos". Assim, por meio de sucessivas interações, um dado subsunçor vai, progressivamente, adquirindo novos significados, ficando mais rico, mais refinado, mais diferenciado e mais capaz de servir de ancoradouro para novas aprendizagens significativas. Enquanto a reconciliação integradora, ou integrativa, é um processo da dinâmica da estrutura cognitiva, simultâneo ao da diferenciação progressiva, que consiste em eliminar diferenças aparentes, resolver inconsistências, integrar significados, fazer superordenações[8].

[8] Para Moreira (2016), a aprendizagem superordenada envolve processos de abstração, indução, síntese, que levam a novos conhecimentos que passam a subordinar aqueles que lhes deram origem.

2.3 MÉTODOS DE ENSINO-APRENDIZAGEM USADOS NO ENSINO DE PROGRAMAÇÃO

No intuito de aumentar a efetividade do ensino-aprendizagem, diversas pesquisas têm sido realizadas nos últimos anos na área de ensino de programação. Desse modo, várias estratégias estão sendo construídas e podem ser classificadas de acordo com os procedimentos metodológicos propostos em cada trabalho, mostrando os esforços que estão sendo realizados para auxiliar o ensino de programação. Conforme Pereira (2004), na literatura, existem três vertentes que buscam por soluções para minimizar os problemas atuais de aprendizagem dos alunos: Ferramentas, Estratégias e Ferramentas e Estratégias/Metodologias.

Dentre as principais ferramentas desenvolvidas nos últimos anos destacam-se: Tutor ICC (2010), The Huxley (PAES *et al*, 2013), TSTView (GAUDENCIO *et al*, 2013), AAPW (GOMES, 2014), Feeper (ALVES, 2014), PCodigo (DE OLIVEIRA, 2015), *Game Logic* (NETTO *et al*, 2017), Cosmo (RABÊLO *et al*, 2018), BrC (GUEDES, 2018) e Tupy Online (ROBERTO *et al*, 2018). As ferramentas aqui apresentadas abordam o estudo inicial de programação, utilizando para isso várias linguagens de programação, sendo algumas focadas na linguagem C, como a ferramenta Huxley. O problema que essas ferramentas que utilizam a linguagem C para auxiliar no ensino de programação não mostram ao aluno onde o erro está acontecendo, apenas retornam se o algoritmo está certo ou errado. O discente, na maioria das vezes, começa a utilizar o processo de tentativa e erro para corrigir o problema, sem saber ao certo o que está fazendo, e, quando consegue fazer funcionar o programa, não sabe explicar o que foi alterado para que o problema fosse resolvido, mas, como funcionou, não procura o professor para sanar as dúvidas encontradas, deixando lacunas que serão observadas no ensino e aprendizagem dos novos conteúdos a serem ministrados.

Entre as propostas que envolvem estratégias e ferramentas, a maioria dos estudos são específicos para os problemas encontrados e essa vertente necessita que seja bem analisada sua utilização, pois fatores importantes precisam ser considerados, tais como: o nível de instrução em que os alunos se encontram, já que todos precisam ser beneficiados independentemente dos conhecimentos que já possuem; e seu estado emocional, pois muitas vezes a simples utilização de uma ferramenta pode causar frustração, sendo um assunto difícil de ser tratado computacionalmente. Assim sendo, vários trabalhos têm sido realizados com base em alguma (s) ferramenta (s) e estratégia (s) bem definidas de ensino-aprendizagem (BELCHIOR; GOMES; MELO, 2013; GONZALES; TAMARIZ, 2014; BELCHIOR *et al.*, 2016; CASAROTTO *et al*., 2018; HEMING, 2018; FAÊDA; BAFFA; PEREIRA,2020).

Finalmente, os trabalhos que contemplam diferentes estratégias/metodologias bem definidas, são de suma importância para comparar com o presente trabalho. Abaixo, cita-se algumas pesquisas relevantes.

- Avaliação em duas fases na disciplina algoritmos (PANSANATO; SILVA, 2020)

A proposta desenvolvida é uma alternativa à avaliação tradicional e segue os seguintes passos: (a) o professor elabora duas provas que contenham o mesmo número de questões e cada questão de uma prova com grau de dificuldade semelhante à sua equivalente na outra prova; (b) na Fase I, os alunos resolvem uma das provas; (c) o professor avalia a resolução dos alunos e mostra a prova com as questões corrigidas e os respectivos apontamentos e questionamentos; (d) o professor discute a resolução das questões, analisa os erros mais comuns e esclarece as dúvidas dos alunos; (e) na Fase II, os alunos resolvem a outra prova, mas não precisam refazer as questões nas quais foi obtido o valor integral na Fase I; (f) repete o procedimento descrito nos itens c) e d); (g) o valor de cada questão para compor a nota da prova bimestral é o maior entre os valores obtidos para a respectiva questão na Fase I e Fase II. A avaliação em duas fases tem a vantagem de permitir a retroalimentação (*feedback*) do processo, isto é, de permitir a obtenção de informações que possam redirecionar a prática empregada na aprendizagem. Assim, o aluno pode reorientar o método de estudo, rever a estratégia adotada e as decisões que foram tomadas, assim como refletir sobre o seu desempenho. Essa característica pode tornar o aluno mais consciente do seu papel no processo de ensino e aprendizagem, uma vez que provoca uma participação mais ativa na adequação da sua aprendizagem. Dessa maneira, percebe-se que a proposta de Pansanato realiza a investigação com relação ao aprendizado do aluno apenas durante o processo de avaliação, o que na maioria das vezes acarreta ao aluno um acúmulo de dúvidas, não sendo possível resgatar as lacunas de seu aprendizado e desanimando e desestimulando o discente na primeira avaliação realizada.

- Método 300 aplicado em disciplinas de algoritmo e programação (JUNIOR *et al.*, 2021)

O método consiste na prática de aprender ensinando, permitindo que os alunos com melhor desempenho sejam mentores de seus colegas de turma, incentivando a empatia e a cooperação entre estudantes que estão, momentaneamente, em estágios de aprendizagem distintos. Para identificar os estudantes e seus respectivos estágios de aprendizagem, todos participantes de uma avaliação individual, assim como os grupos de colaboração, são construídos com base no rendimento de cada um na avaliação. Assim, garante-se que todos os grupos serão compostos por alunos em diferentes estágios de aprendizagem e/ou conhecimento

prévio, garantindo a heterogeneidade. O objetivo de cada grupo é o de dar apoio aos estudantes com baixo rendimento por meio de encontros presenciais ou outros meios de comunicação e interação. Por outro lado, a proposta de Cheung exige que os alunos possuam uma maturidade e comprometimento no processo de desenvolvimento de sua metodologia o que muitas vezes não é observado nos alunos que estão iniciando o ensino médio.

- *Gamification technique for teaching programming* (CARREÑO *et al.*, 2018)

Essa proposta metodológica é baseada na gamificação, elementos e técnicas de *design* de jogos. A técnica de gamificação foi utilizada no ensino de programação em disciplinas introdutórias nos cursos de graduação, como suporte para aumentar a capacidade na resolução de problemas. Conforme passavam as semanas, o nível de dificuldade aumentava e percebeu-se que as resoluções não se mantiveram constantes, sendo que de quatro grupos apenas um conseguiu resolver todos os desafios. O progresso era quantificado pelo professor, como se os alunos estivessem em uma competição. Os autores consideraram que a técnica é válida para avaliar o comportamento humano durante o processo aprendizagem. Além disso, o método desenvolvido busca a interatividade entre os componentes dos grupos, estimula a participação, mas não avalia a aprendizagem do aluno individualmente, podendo o grupo ter resolvido satisfatoriamente o problema, mas ainda ter alunos com lacunas em seu aprendizado, o que cria uma falsa perspectiva de que se o grupo resolveu o problema todos os alunos estão conseguindo entender o conteúdo.

- *Gradually Learning Programming* (CAZZOLA; OLIVEIRA, 2016)

Proposta de aprendizagem gradual, em que um programa é escrito a partir de um modelo incremental onde o aluno vai modificando a complexidade do código conforme aprende novas estruturas de linguagem de programação. A partir de uma linguagem de programação escolhida, o professor modela um problema de forma que a solução possa ser incrementada gradualmente em nível de complexidade crescente. O modelo de ciclo de aprendizagem gradual é composto pelas fases de crescimento da linguagem, da exposição de novos conceitos e da prática dos alunos com incrementação do código. Contudo, a aprendizagem gradual se destaca por utilizar um princípio intrínseco da área de desenvolvimento de sistemas para o ensino de algoritmos. Ao implementar modificações evolutivas no código à medida que novos conceitos são aprendidos pelos alunos, instaura uma perspectiva real do que é feito na prática. Além de facilitar a aprendizagem em estágios, já familiariza os alunos indiretamente com o que acontece no cotidiano dos desenvolvedores. Vale ressaltar que não ocorre um acompanhamento

individual dos alunos, nem um retorno das dificuldades que possam o auxiliar em suas lacunas de aprendizagem.

- Metodologia *Peer Instruction* (DIEMER *et al.*, 2019)

Esse estudo avalia a aplicação da metodologia ativa *Peer Instruction* no aprendizado de algoritmos a fim de auxiliar os discentes a compreenderem o processo de construção de programas de computadores. A metodologia preconiza que os estudantes tenham contato prévio com o conteúdo, ou seja, antes da aula, assim como durante e depois da aula, enquanto o docente faz explanações curtas complementares ao estudo realizado. Após a explanação, o professor submete os discentes a um problema real, questionando-os ou os desafiando a resolver um caso prático em que aquele conteúdo se faz necessário. Valendo-se de algum mecanismo de questionamento, o professor obtém *feedback* sobre o grau de compreensão da explanação realizada. Se o índice de êxito é muito baixo (inferior a 30%), o professor deve complementar a sua explicação e refazer o processo de questionamento. Quando o índice de acertos é superior a 70% a metodologia preconiza que houve compreensão dos conceitos pela maioria dos estudantes e que é possível seguir em frente. A troca de experiência entre os pares ocorre justamente quando o índice de acertos fica entre 30% e 70%, indicando que uma parcela dos estudantes já compreendeu o conteúdo e outra parcela ainda carece de auxílio e mais tempo. Portanto, os alunos devem formar grupos de estudo com colegas que têm respostas divergentes, discutir o assunto, argumentar e defender seus pontos de vista a fim de chegar a uma resposta comum. Esse é o ponto máximo da metodologia, em que há intensa reflexão e postura ativa dos estudantes no processo de aprendizagem. Depois desse período de discussão, um novo *feedback* é colhido pelo professor e, conforme o índice de acertos, o professor decide complementar ou não com novo momento de explanação ou apenas com uma reflexão de consolidação do tema. Vale a pena ressaltar que o método *Peer Instruction* é um método satisfatório para ser aplicado em cursos que não possuem um rol de disciplinas tão grande em sua grade curricular, pois muitas atividades fazem com que os alunos não deem o retorno esperado.

- Metodologia 7Cs (LIMA; DINIZ; ELIASQUEVICI, 2019)

A proposta de Lima *et al.*, aborda uma sequência lógica de conteúdos e utiliza dimensões significativas para o ensino de programação. Essa metodologia, então, traz dimensões importantes a serem avaliadas, além de um *feedback* que é dado aos alunos pelos monitores, auxiliando os discentes em seus estudos. A metodologia proposta pretende facilitar a compreensão dos conteúdos básicos da disciplina de algoritmos, buscando aprimorar

habilidades e competências a partir do desenvolvimento de uma sequência lógica de aprendizagem. A Metodologia 7Cs contém, em sua estrutura, sete dimensões (Compreender, Conceber, Completar, Compatibilizar, Corrigir, Construir e Contribuir), cada uma delas possui um único objetivo que traça um caminho para construção do conhecimento sobre os conteúdos trabalhados na disciplina. A metodologia é inserida para os alunos por meio de um jogo que é estruturado em forma de cartas com situações problemas e desafios, os quais contém seis territórios. Cada território representa uma dimensão da Metodologia 7Cs. A dimensão "Contribuir" está presente em todo o jogo pelo fato das atividades serem realizadas de forma colaborativa, não constituindo, portanto, um dos territórios. Além de apresentar a Metodologia 7Cs, monitores também inserem junto conceitos iniciais de programação. A cada desafio desenvolvido os monitores realizavam o *feedback* para os alunos sobre a dimensão correspondente. Isso tudo era feito em um único dia, sendo que no segundo dia os alunos já eram colocados em contato com pequenos exemplos da linguagem *Python.* Dessa forma, as dimensões foram sendo trabalhadas por meio das tarefas dadas a eles. A diferença da proposta atual é que não utiliza fluxogramas para introduzir o conteúdo. As dimensões estabelecidas são importantes, mas teria que aplicá-las mais adiante e não em um primeiro momento da disciplina, pois os alunos, vindos do ensino fundamental, têm muitas lacunas básicas em sua formação, como a matemática e interpretação de texto, além do raciocínio lógico que nunca foi trabalhado. Fazê-los entender as dimensões juntamente com o conteúdo inicial não seria uma tarefa nada fácil.

Após apresentar parte da base teórica que irá embasar a discussão do trabalho proposto, o próximo capítulo será destinado para descrever a essência da concepção de pesquisa utilizada nessa pesquisa – *Educational Design Research* (EDR). Essa concepção de pesquisa é utilizada em nosso grupo de trabalho Métodos e Processos de Ensino e Aprendizagem (MPEAC)[9], por abordar investigações educacionais que tem como compromisso desenvolver, simultaneamente, percepções teóricas e soluções práticas. Portanto a EDR não apenas propõe elaborar uma solução para que depois seja testada, mas também cria a proposta, sendo que o teste tem que ser feito ao mesmo tempo, tendo sua validação realizada dentro de sua própria experiência. A validação só é promovida quando ocorre uma transformação no contexto em análise, caso contrário, a proposta não foi bem-sucedida.

[9] Registro no CNPQ: http://dgp.cnpq.br/dgp/faces/consulta/consulta_parametrizada.jsf

A EDR também traz em sua proposta solucionar problemas que serão conduzidos no contexto em que a problemática educacional é encontrada. Em outras palavras, a investigação empírica não pode ser realizada em circunstâncias nas quais existe a possibilidade de controle de variáveis, como o ambiente de laboratório (MCKENNEY; REEVES, 2012). Dessa forma, compreender as características e sua utilização passa a ser de extrema importância, pois as etapas pertencentes a essa concepção de pesquisa serão utilizadas na elaboração deste trabalho.

3 EDUCATIONAL DESIGN RESEARCH: UMA NOVA PERSPECTIVA DE INVESTIGAÇÃO

Neste capítulo, abordaremos o conceito da *Educational Design Research* (EDR), desde o surgimento até a importância dentro da pesquisa educacional, assim como as caraterísticas e o ciclo geral que a compõe.

3.1 EDUCATIONAL DESIGN RESEARCH

A *Educational Design Research*, traduzida como Pesquisa em Design Educacional (PDE), nasce na década de 1990 para desenvolver uma nova perspectiva que busca aliar aspectos teóricos da pesquisa com a prática. Essa nova linha de pesquisa foi introduzida na educação por Brown (1992) e Collins (1992) e, a partir deles, outros autores passaram a usar termos semelhantes para definir seus próprios tipos de pesquisa, tais como *Design Experiments* (BROW, 1992), *Development Research* (RICHEY *et al*., 2004), *Creating context: Design-Based Research in creating and understanding CSCL* (HOADLEY, 2002), *Design Basead Research Collective* (DBRC-COLLECTIVE, 2003), *Formative Research* (REINKIN; BRADLEY, 2008) e a já citada *Educacional Design Research* (VAN DEN AKKER *et al.*, 2006). Um precursor mais remoto da EDR foi John Dewey (1990), que indicava ser a educação um conhecimento prático, com estudos e pesquisas voltados para o desenvolvimento de soluções aplicáveis à prática concreta dos ambientes de ensino e aprendizagem (MATTA; SILVA; BOAVENTURA, 2014).

Dentre essas variadas abordagens, a escolhida para ser utilizada nesse trabalho foi a abordagem *Educational Design Research*, considerando o argumento de Mckenney e Reeves (2012). Os autores defendem que seria o nome mais conveniente para essa perspectiva de investigação porque, contém explicitamente o termo Educação, e isso evitaria confusões desnecessárias acerca do escopo de atuação dessa perspectiva de pesquisa.

3.1.1 EDR: uma nova visão na pesquisa educacional

Nos últimos anos, percebeu-se um crescimento de investigações em educação e, ao mesmo tempo, que elas eram pouco voltadas para melhorias efetivas dos processos educacionais, ou seja, à pesquisa aplicada (MATTA; SILVA; BOAVENTURA, 2014). Collins (1992) também destaca esse problema inerente a muitas pesquisas na área da educação, pois uma inovação projetada em laboratório e a inovação implementada em salas de aula reais são

muito diferentes. Isso pode ser visto como um problema, já que as teorias educacionais são raramente confirmadas por evidências inequívocas e/ou completas (MCKENNEY; REEVES, 2019). Autores como Van Den Akker (1999) e De Vries e Pieters (2007) argumentam que a pesquisa educacional não se concentrou nos problemas e nas questões confrontadas na prática cotidiana e lamentam a falta de conhecimento útil produzido pela pesquisa, o que poderia ajudar a informar o desenvolvimento de inovações e reformas. Broekkamp e Hout-Wolters (2007) agruparam essas questões em quatro temas principais: a) a investigação educacional produz poucos resultados conclusivos; b) a pesquisa educacional produz poucos resultados práticos; c) os profissionais acreditam que a investigação não é conclusiva ou prática; e d) os praticantes fazem pouco uso da pesquisa educacional.

Diante desse cenário, nas últimas décadas, houve uma preocupação em torno da lacuna apresentada entre pesquisa e prática na investigação educacional. A EDR, por suas ideias e características, passa a ser uma forma de investigação educacional, tendo como compromisso desenvolver, simultaneamente, percepções teóricas e soluções práticas. Para entender melhor essa investigação, Bakker (2018, p. 23-24) define a EDR como

> [...] o estudo sistemático de concepção, desenvolvimento e avaliação de intervenções educacionais, – como programas, estratégias de ensino-aprendizagem e materiais, produtos e sistemas – como soluções para tais problemas, que visa também avançar o nosso conhecimento sobre as características dessas intervenções e os processos para as conceber e desenvolver.

Ao utilizar a abordagem teórico-prática, a EDR também possibilita que os pesquisadores unam as vantagens das metodologias baseadas em dados qualitativos e quantitativos em suas investigações. Conforme Terry e Shattuck (2012), enquanto o método quantitativo revela padrões amplos de discurso, com base em *design*, o método qualitativo facilita o esclarecimento local por meio da observação, descrição e interpretação das características das interações e do papel do corpo docente, dos alunos e das tarefas. Não há mais sentido em separar essas duas formas nem em investir demasiadamente nessa diferença. Basta, apenas aplicá-las na medida do necessário, na direção do foco da pesquisa.

Portanto, aproxima-se de Mckenney e Reeves (2012), os quais argumentam que a EDR sozinha não resolverá os problemas descritos aqui. No entanto, essa forma de investigação é um caminho promissor para melhorar a robustez e a relevância da pesquisa educacional, já somando resultados importantes e capazes de prover pesquisas de desenvolvimento e inovação, aplicadas em educação.

3.1.2 Características da pesquisa em *design* educacional

Para Mckenney e Reeves (2012), Matta, Silva e Boaventura (2014) e Bakker (2018), a EDR, em geral, apresenta cinco características-chave que compõem a pesquisa em *design* educacional. Elas não precisam, necessariamente, serem todas cumpridas, pois são vistas como características familiares que, muitas vezes (mas nem sempre) precisam andar juntas. As características estão citadas abaixo e serão descritas em detalhes, na sequência.

- ✓ Teoricamente Orientada
- ✓ Intervencionista
- ✓ Colaborativa
- ✓ Fundamentalmente Responsiva
- ✓ Iterativa

3.1.2.1 Teoricamente Orientada

Essa característica utiliza a teoria existente para estruturar a investigação, sendo usada não apenas para enquadrar a pesquisa, mas também para moldar o *design*[10] de uma solução que busque resolver um problema real (MCKENNEY; REEVES, 2012). Além de estruturar a proposta prática, a base teórica também é estudada e potencialmente melhorada e compreendida, na medida dos resultados.

3.1.2.2 Intervencionista

Para Mckenney e Reeves (2012), o termo "intervencionista" é amplamente usado para abranger as diferentes soluções projetadas, podendo ser considerados como intervenções: a) produtos educacionais, tais como: materiais didáticos de toda natureza e suporte; b) processos pedagógicos, como: recomendações de atitude docente, novas propostas didáticas; c) programas educacionais, como: currículos, cursos, organização de temas e didáticas; d) políticas educacionais, como: protocolos de avaliação docente ou discente, procedimentos e recomendações de investimento, opções para relação entre a escola e a comunidade.

[10] Conforme Bakker (2018), *design* é projeto de intervenção que consiste em uma sequência de atividades que, juntas ou em combinação, intervêm nos conhecimentos existentes a fim de estimular um novo aprendizado que leva a novas práticas.

As ações intervencionistas, por meio de suas implementações, procuram solucionar o problema que originou a busca pelos processos/produtos, a fim de melhorar o cenário real em que a pesquisa está sendo realizada.

3.1.2.3 Colaborativa

A pesquisa em *design* educacional é sempre conduzida em meio a vários graus de colaboração, pois, ao buscar soluções para problemas concretos, faz com que todos os envolvidos (pesquisadores e profissionais da área) colaborem no processo de investigação. A ideia é considerar todos como parte da equipe de pesquisa, desde a definição do problema até o resultado final. Durante o processo de investigação, é necessário que nenhum conhecimento seja negado, nem o universitário nem o comunitário, e que nenhum deles seja posto em situação de dominância, pois o que realmente vai legitimar os resultados é a validação colaborativa de todos no processo. Conforme Matta, Silva e Boaventura (2014), o grau de colaboração entre comunidade de prática e pesquisadores pode ser vista de três maneiras: extração de dados, parceria de investigação e acordo de aprendizagem.

Na extração de dados, o processo é conduzido por um pesquisador externo à comunidade, o qual elabora, organiza e relata a investigação. A comunidade está empenhada na prática estudada, participando do processo de implementação da solução proposta. Já na colaboração por parceria de investigação, o pesquisador também é externo ao contexto de aplicação e está engajado exclusivamente na reflexão, ao passo que a comunidade envolvida está engajada tanto na reflexão quanto na ação. E para finalizar, há o acordo de aprendizagem, em que a elaboração e a execução reflexiva são compartilhadas entre o pesquisador e a comunidade. Em todas as formas de colaboração, o objetivo é sempre conduzir da melhor forma as soluções propostas para melhorar a prática educacional dentro do contexto pesquisado.

3.1.2.4 Fundamentalmente Responsiva

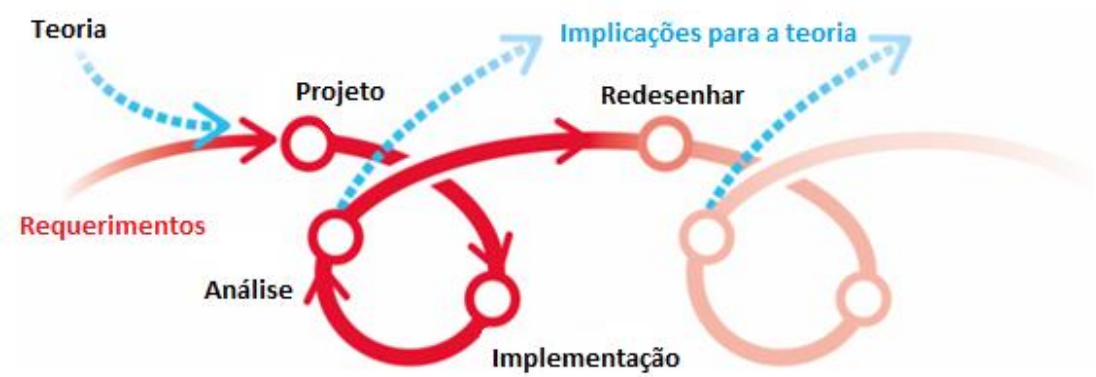
Conforme Alves (2018), a PDE é considerada responsiva porque suas pesquisas são conduzidas em ambientes complexos e mutáveis no tempo. Logo, elas devem ser capazes de se ajustar a essas mudanças, isto é, devem ser responsivas ao contexto de implementação. As mudanças vão ocorrendo no momento em que cada iteração é concluída, em que o modelo é refinado e, em seguida, implementado numa próxima etapa. Em seguida, são examinados os efeitos dos refinamentos na aprendizagem dos alunos, procurando tanto os resultados que

confirmam a resolução dos desafios quanto os desafios de aprendizagem adicionais. Por meio desse processo responsavelmente fundamentado, os professores foram capazes de refinar sua compreensão dos processos de aprendizagem dos alunos e, ao mesmo tempo, ajustar seu modelo de ensino (MCKENNEY; REEVES, 2012).

3.1.2.5 Iterativa

Para finalizar, de acordo com Mckenney e Reeves (2012), a PDE evolui com o tempo por meio de suas várias iterações de investigação, desenvolvimento, teste e refinamento. Essas etapas podem ser compreendidas como subciclos que compõem um ciclo de estudo completo que, ao final, ao ser aperfeiçoado, dará início a um novo ciclo, que contemplará as mesmas etapas e assim sucessivamente. Isso ocorre porque a PDE é voltada para construções de soluções práticas, raramente um único ciclo é suficiente para responder aos problemas e estabelecer melhorias. A Figura 3 ilustra essas etapas.

Figura 3 – Pesquisa baseada em design como um processo contínuo de inovação



Fonte: Traduzida de Fraefel (2012, p. 9).

Por meio das etapas ilustradas na Figura 3, percebe-se que o processo de pesquisa é estimulado pela melhoria no design. A cada ciclo, subciclos são implementados e avaliados e, conforme as análises vão sendo realizadas, um refinamento na solução pode ou não ser desenvolvido. No intuito de melhor compreender as fases pertencentes a um ciclo da PDE, uma descrição será realizada a seguir.

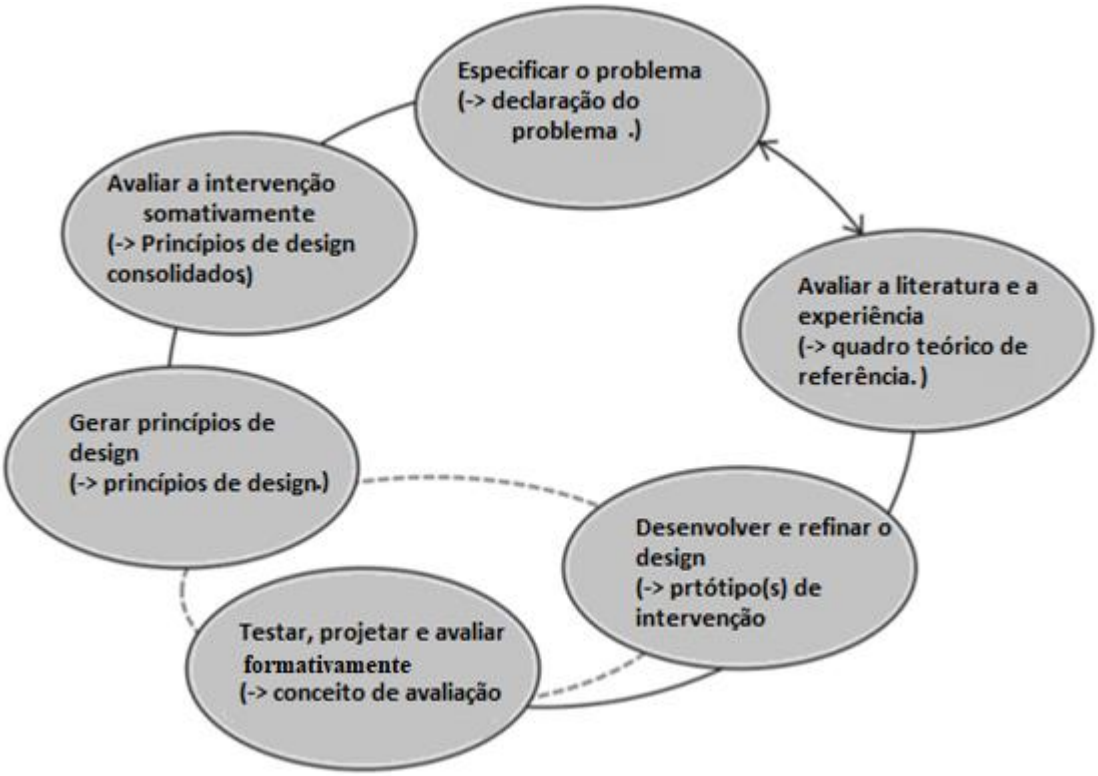
3.2 CICLO GERAL DA PESQUISA EM *DESIGN* EDUCACIONAL

A pesquisa em *design* educacional é conduzida por meio de ciclos e cada um deles é dividido em subciclos (ou fases). Fazem parte dessas fases:

- ✓ Análise
- ✓ Design e desenvolvimento
- ✓ Avaliação
- ✓ Revisão

Essas quatro fases, estão ilustradas na Figura 4, as quais serão detalhadas na sequência.

Figura 4 – Ciclos de pesquisa e desenvolvimento no contexto da pesquisa em design.



Fonte: Traduzida de Euler (2014, p. 20).

Analisando a Figura 4, percebe-se que a primeira fase tem como objetivo definir especificamente o problema. Para isso, leva em conta a literatura disponível e o conhecimento prático do pesquisador, estabelecendo um diálogo importante para a definição do problema a ser estudado. Em seguida, na segunda etapa, o investigador, apoiado inicialmente na literatura

e em sua prática, vai desenvolver os princípios de *design*, os quais guiarão o processo de construção da primeira versão do produto/processo educacional.

Seguindo o ciclo elaborado por Euler (2014), o próximo estágio realiza os testes, avalia a solução implementada e refina a proposta desenvolvida anteriormente, modificando, se necessário, os princípios de *design* para que um novo teste seja realizado, bem como uma nova avaliação e um refinamento da proposta sejam construídos. Esse processo de replicação constante forma os subciclos da pesquisa em *design* educacional, o qual pode ser desenvolvido quantas vezes o pesquisador achar necessário, até que uma solução efetiva seja encontrada.

A última fase, a avaliação geral da pesquisa, só é realizada após um sucessivo refinamento dos princípios de *design* com o qual se pretende encontrar a melhor solução prática para o problema a ser solucionado. Isto é, busca-se uma melhoria, na teoria, por meio dos princípios de *design* e, na prática, por meio dos produtos implementados. Nessa fase, resultados são gerados e, para Alves (2018, p. 44), as possíveis análises encontradas podem ser classificadas como:

> A explicitação de princípios de *design* passíveis de serem generalizados para outros contextos; A obtenção de produto/processo educacional efetivo — ou seja, de uma solução prática viável — para o seu contexto original de aplicação; O desenvolvimento profissional dos sujeitos envolvidos no processo de pesquisa, capacitando-os a resolver outros problemas de forma autônoma.

Entretanto, apenas analisar os resultados gerados pela PDE não é suficiente: é preciso focar atenção na questão da generalização, pois para Matta, Silva e Boaventura (2014), uma das premissas mais importantes do método científico é a de reconhecer o conhecimento científico com a possibilidade de ser generalizado e aplicável em outros problemas que não sejam aqueles que o originou. Na PDE, a generalização está sempre a cargo da capacidade e da possibilidade de fazer migrar uma efetiva intervenção de nossa situação de aplicação para outras, isto é, permitir que o conhecimento gerado em uma condição particular possa, posteriormente, ser aplicado a outras situações similares enfrentadas em outros contextos, respeitando as particularidades de cada situação.

Conforme Mckenney e Reeves (2012), existem duas formas úteis de se realizar a generalização na PDE:

- ✓ A primeira procura generalizar um conjunto particular de dados para uma teoria mais ampla, aumentando o alcance ao utilizar os princípios de *design*, que também são replicados e refinados, conforme o novo contexto de aplicação.

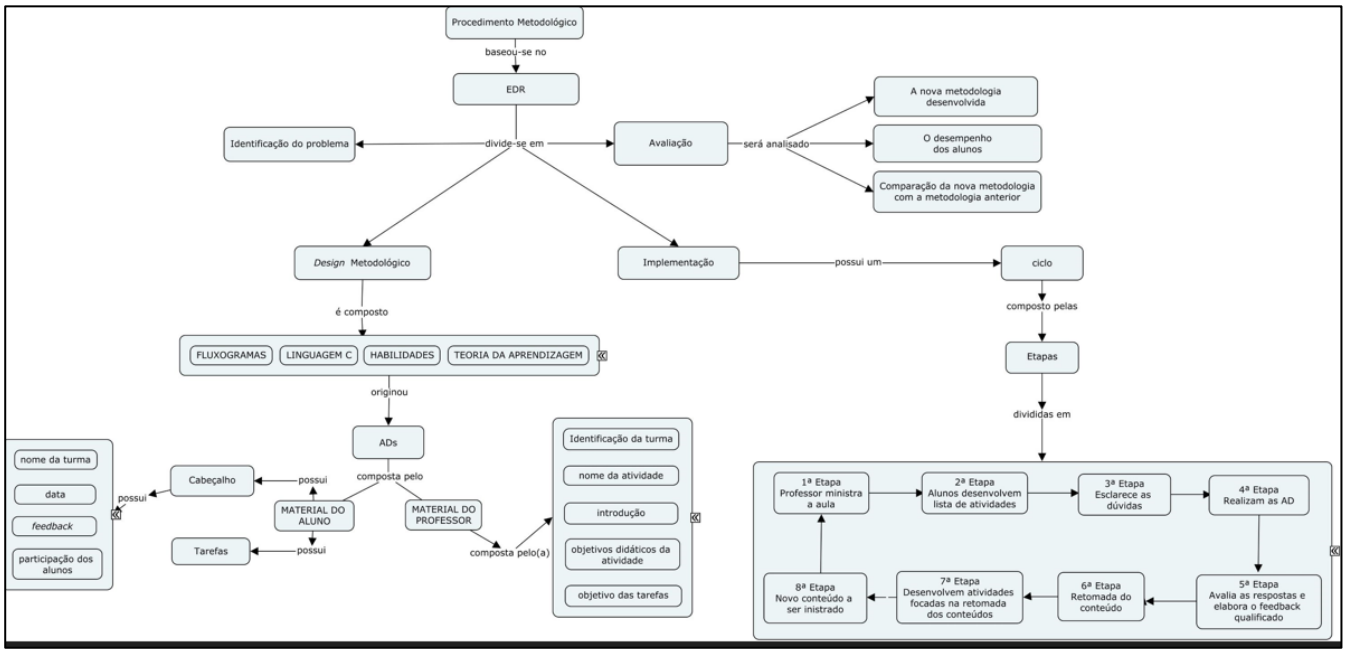
- ✓ A segunda considera a generalização caso-a-caso, em que a replicação é realizada uma vez em cada contexto diferente do original, não em vários contextos ao mesmo tempo.

Portanto, é importante perceber que o que pode ser generalizado é a forma de interpretar e entender as soluções implementadas para que sua replicação possa ser realizada em outros contextos, mesmo que isso implique em modificações nas soluções já desenvolvidas.

4 PROCEDIMENTOS METODOLÓGICOS

Neste capítulo, será detalhado de que forma a pesquisa foi desenvolvida. Para isso utilizou-se as fases da metodologia EDR: 1) identificação do problema; 2) *design* Metodológico; 3) implementação; e 4) avaliação. A Figura 5, elucida a forma de como essas fases se desenvolveram, realizando após a descrição metodológica de cada etapa.

Figura 5 – Desenvolvimento da pesquisa.



Fonte: Da Autora (2022).

4.1 IDENTIFICAÇÃO DO PROBLEMA

O problema a ser especificado nasce da dificuldade dos alunos ingressantes do curso Técnico em Informática Integrado ao ensino médio em compreender o conteúdo relacionado à disciplina de Programação I, situado em uma Instituição Federal de ensino básico, técnico e tecnológico do Estado do Rio Grande do Sul. As dificuldades encontradas no desenvolvimento da disciplina se refletem desde quando o curso começou a ser ofertado, no ano de 2011, cujo percentual de alunos reprovados variou de 29 a 62%, além do baixo número de alunos com bom desempenho (Quadro 2).

Quadro 2 – Índice de desempenho dos alunos do primeiro ano da disciplina de Programação I.

Ano/Total de alunos	Percentual de Alunos Aprovados	Percentual de Alunos Reprovados	Percentual de Alunos Transferidos	Percentual de alunos Aprovados sem os transferidos	Percentual de alunos Reprovados sem os transferidos
2011 / 30	36%	60%	4%	38%	62%
2012 / 39	59%	41%	0%	59%	41%
2013 / 39	41%	38%	21%	51,6%	48,3%
2014 / 31	71%	29%	0%	71%	29%
2015 / 36	56%	38%	6%	58,8%	41,2%
2016 / 31	42%	58%	0%	42%	58%
2017 / 39	69%	31%	0%	69%	31%
2018 / 60	52%	45%	2%	52,5%	47,5%
2019 / 64	64%	34%	2%	65%	35%
2020/ 78	55,2%	28,2%	16,6%	66%	34%
2021/64	73%	16%	11%	82,4%	17,6%

Fonte: Da Autora (2020).

Diante do exposto, percebe-se uma alta taxa de reprovação dos alunos na disciplina de Programação I, o que, muitas vezes, leva o aluno a desistir do curso, pois, quando reprova em uma disciplina, ele precisa repetir o ano e fazer, novamente, as 13 disciplinas previstas no currículo. Essa situação não é exclusiva do *campus* onde a pesquisa está sendo realizada, já que se estende a todos os *campi* que possuem o curso Técnico em Informática Integrado ao ensino médio pertencentes a Instituição.

Observa-se que os dados referentes ao Quadro 2 de aprovação/reprovação da disciplina de Programação I oscilam a cada ano, pois há anos em que os alunos entram no ensino médio mais preparados para frequentar um curso técnico na área de informática, com uma sólida base de conhecimentos nas disciplinas de Matemática e Português. Em outros anos, essa lacuna é mais acentuada, aumentando a dificuldade dos alunos no Curso. Diante dessas informações, percebe-se a dificuldade dos estudantes em cursar a disciplina de Programação I, sendo os altos índices de reprovação um problema a ser superado.

Todos os dados apresentados até o momento sempre foram muito discutidos e, a cada ano, uma nova proposta pedagógica foi sendo implementada por meio da oferta de monitorias[11], atendimento individual aos alunos, utilização de ferramentas como VisualG[12], Scratch[13] entre outras. No entanto, a oscilação nos números permaneceu alta.

A fim de alterar essa situação, a presente pesquisa se propõe em desenvolver, implementar e analisar um conjunto de ADs que possam contribuir com o ensino e aprendizagem dos alunos do curso técnico em informática da respectiva Instituição Federal de ensino básico, técnico e tecnológico.

4.2 *DESIGN* METODOLÓGICO

Nesta subseção, será realizada uma descrição dos princípios de *design* que irão guiar o desenvolvimento dessa pesquisa. Ao pensar em uma solução que atendesse as necessidades e especificidades dos alunos do curso técnico em informática do *campus* em análise, elaborou-se uma proposta que tivesse como base a utilização: de fluxogramas, da Linguagem C, das habilidades essenciais a serem analisadas nos alunos ingressantes no curso de informática e da Teoria de Aprendizagem.

Esses princípios foram utilizados como base para a construção das ADs, em que suas respectivas importâncias foram descritas no Capítulo 2, intitulado como Referencial Teórico. Para compor as ADs, desenvolveu-se o material do professor e do aluno, os quais terão seus aspectos de *design* descritos abaixo.

4.2.1 Material do Professor

Inicialmente, para elaborar o material do professor, considerou-se os aspectos desenvolvidos na construção das ADs. No intuito de desenvolver frequentemente as ADs, levou-se em consideração o conteúdo programático da disciplina, em que constavam sete tópicos a serem desenvolvidos durante o ano letivo de 2021. Assim, os tópicos que compõem este cronograma são: introdução aos conceitos básicos de programação, dois comandos de

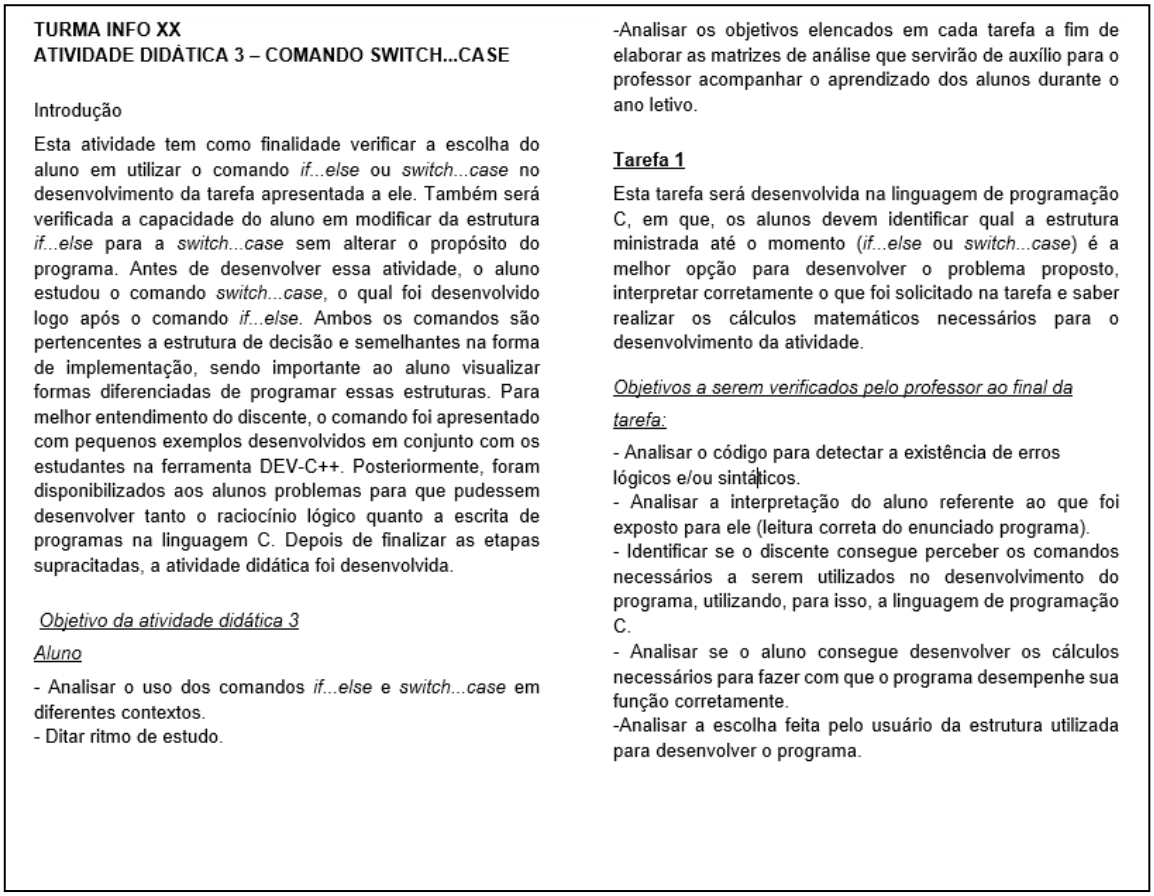
[11] Reforço disponibilizado pela Instituição duas vezes por semana por alunos que já cursaram a disciplina. O material das aulas de reforço é preparado conjuntamente pelo professor e pelo monitor, conforme as necessidades dos discentes.
[12] Visualg é um programa gratuito de edição, interpretação e execução de algoritmos, com uma linguagem próxima ao portugol.
[13] O Scratch é uma linguagem de programação criada pelo grupo Lifelong Kindergarten da universidade americana MIT. É um tipo de programação “visual”, mais simplificada para o aluno iniciante em programação.

seleção (*if...else* e *switch...case*), três laços de repetição (*while, do...while* e *for*) e, por último, estruturas homogêneas (*vetor*). Na construção dessas atividades, foram elaboradas tarefas que pudessem verificar especificamente o aprendizado do aluno referente a cada estrutura. Essa abordagem permitirá ao professor monitorar as lacunas observadas no decorrer do ano letivo e, se necessário, realizar uma retomada nos conteúdos (individualmente ou em grupo) a fim de não deixar nenhum aluno com defasagem, evitando que se desmotive, reprove ou abandone o curso.

Após, para elaborar o material do professor, foi necessário pensar em uma estrutura que pudesse auxiliar o docente a conduzir o desenvolvimento das tarefas específicas em cada AD. A estrutura elaborada é composta por: identificação da turma, nome da atividade a ser desenvolvida, introdução, objetivos didáticos da AD e tarefas. A introdução tem como objetivo descrever um breve resumo do que será ministrado pelo professor e a forma que será desenvolvido em aula. Os objetivos didáticos têm como finalidade mostrar o que se busca atingir em relação ao aluno e ao professor. Por fim, as tarefas descrevem os objetivos que terão que ser alcançados pelos alunos e os objetivos a serem verificados pelo professor ao seu final.

A Figura 6 ilustra a estrutura desenvolvida que irá compor o referido material. Essa estrutura será apresentada por meio de um modelo reduzido referente a AD3, a qual mostra a tarefa 1 (as tarefas 2 e 3 seguem essa mesma estrutura).

Figura 6 – Modelo reduzido, contendo a tarefa 1, do material do professor referente a AD3, comando switch...case.



TURMA INFO XX
ATIVIDADE DIDÁTICA 3 – COMANDO SWITCH...CASE

Introdução

Esta atividade tem como finalidade verificar a escolha do aluno em utilizar o comando *if...else* ou *switch...case* no desenvolvimento da tarefa apresentada a ele. Também será verificada a capacidade do aluno em modificar da estrutura *if...else* para a *switch...case* sem alterar o propósito do programa. Antes de desenvolver essa atividade, o aluno estudou o comando *switch...case*, o qual foi desenvolvido logo após o comando *if...else*. Ambos os comandos são pertencentes a estrutura de decisão e semelhantes na forma de implementação, sendo importante ao aluno visualizar formas diferenciadas de programar essas estruturas. Para melhor entendimento do discente, o comando foi apresentado com pequenos exemplos desenvolvidos em conjunto com os estudantes na ferramenta DEV-C++. Posteriormente, foram disponibilizados aos alunos problemas para que pudessem desenvolver tanto o raciocínio lógico quanto a escrita de programas na linguagem C. Depois de finalizar as etapas supracitadas, a atividade didática foi desenvolvida.

Objetivo da atividade didática 3

Aluno

- Analisar o uso dos comandos *if...else* e *switch...case* em diferentes contextos.
- Ditar ritmo de estudo.

-Analisar os objetivos elencados em cada tarefa a fim de elaborar as matrizes de análise que servirão de auxílio para o professor acompanhar o aprendizado dos alunos durante o ano letivo.

Tarefa 1

Esta tarefa será desenvolvida na linguagem de programação C, em que, os alunos devem identificar qual a estrutura ministrada até o momento (*if...else* ou *switch...case*) é a melhor opção para desenvolver o problema proposto, interpretar corretamente o que foi solicitado na tarefa e saber realizar os cálculos matemáticos necessários para o desenvolvimento da atividade.

Objetivos a serem verificados pelo professor ao final da tarefa:

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.
- Analisar a interpretação do aluno referente ao que foi exposto para ele (leitura correta do enunciado programa).
- Identificar se o discente consegue perceber os comandos necessários a serem utilizados no desenvolvimento do programa, utilizando, para isso, a linguagem de programação C.
- Analisar se o aluno consegue desenvolver os cálculos necessários para fazer com que o programa desempenhe sua função corretamente.
-Analisar a escolha feita pelo usuário da estrutura utilizada para desenvolver o programa.

Fonte: Da Autora (2021).

4.2.2 Material do Aluno

Após desenvolver o material do professor, elaborou-se a estrutura do material do aluno, a qual foi utilizada no desenvolvimento de todas as ADs, alterando somente as tarefas propostas em cada atividade. Dessa forma, cada AD desenvolvida tem como objetivo investigar os conceitos básicos de cada conteúdo ministrado, analisando-o individualmente e de forma integrada aos conteúdos já trabalhados, fazendo com que os alunos consigam desenvolver, a cada conteúdo ministrado, algoritmos mais elaborados por meio de uma base sólida.

A fim de compor as ADs, foram considerados os objetivos a serem alcançados pelos alunos descritos no material do professor, bem como as habilidades a serem avaliadas: interpretação de texto, abstração, raciocínio lógico e noções matemáticas. Ao final de cada AD, foi inserido um questionário, que servirá de auxílio para que o professor construa a próxima atividade didática a ser trabalhada e verifique se a percepção dos alunos corrobora com as ADs desenvolvidas, levando em consideração o empenho, a aprendizagem, o foco nos estudos, e o

trabalho despendido, conforme a Figura 7. Esse questionário é uma prática do grupo[14] quando se desenvolve ADs em sequência, onde é necessário ser curto para que o aluno responda e que possa avaliar a carga de trabalho e dificuldade da atividade proposta.

Figura 7 – Questionário 1 a ser respondido pelos alunos no final de cada AD.

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
1 (Não contribuiu) 2 (Contribuiu pouco) 3 (Contribuiu) 4 (Contribuiu muito) 5 (Contribuiu por completo)

Aprendizagem:
1 (Não contribuiu) 2(Contribuiu pouco) 3 (Contribuiu) 4 (Contribuiu muito) 5 (Contribuiu por completo)

Foco nos estudos:
1 (Não contribuiu) 2 (Contribuiu pouco) 3 (Contribuiu) 4 (Contribuiu muito) 5 (Contribuiu por completo)

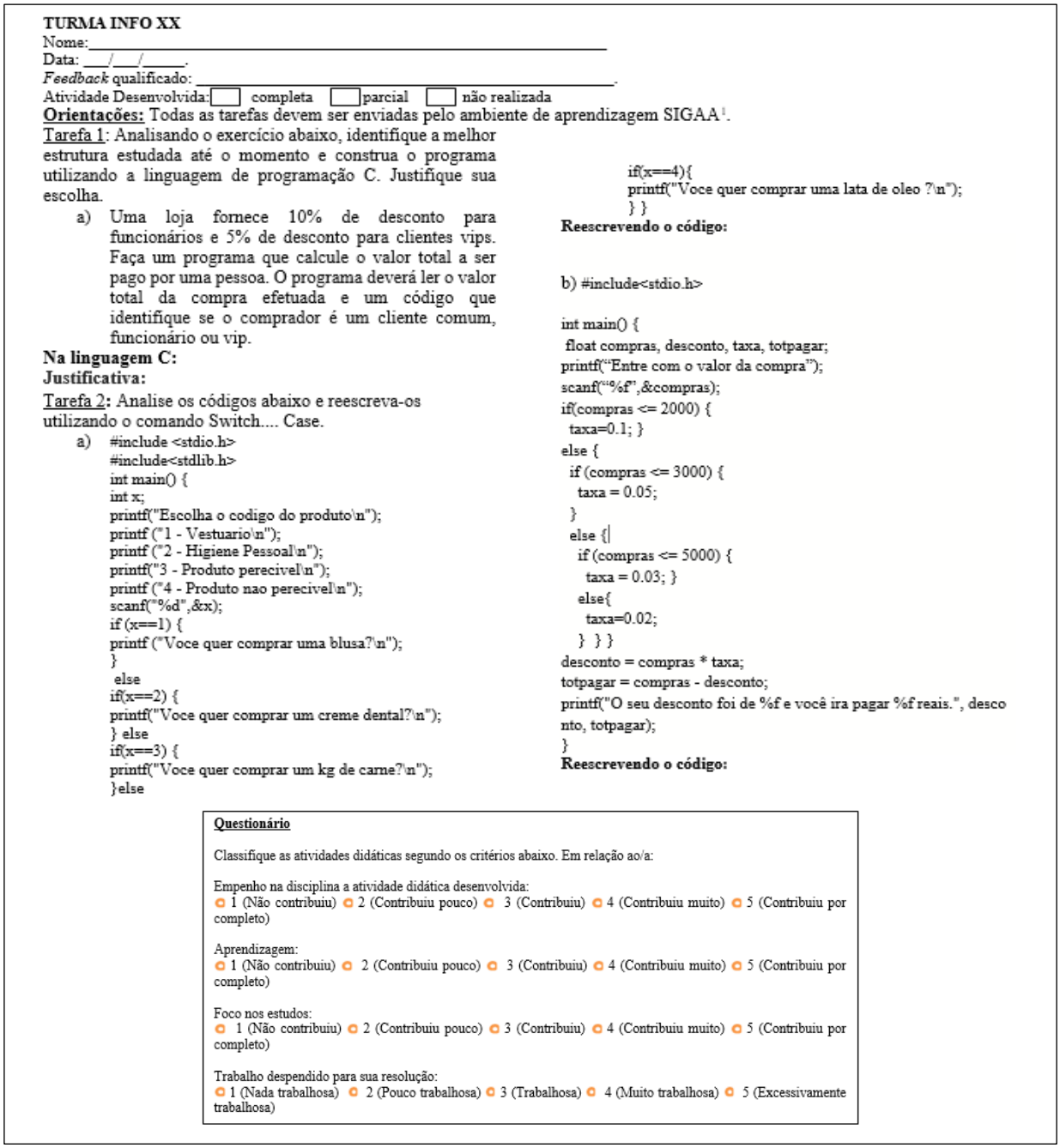
Trabalho despendido para sua resolução:
1 (Nada trabalhosa) 2 (Pouco trabalhosa) 3 (Trabalhosa) 4 (Muito trabalhosa) 5 (Excessivamente trabalhosa)

Fonte: Da Autora (2021).

Após definir os aspectos que farão parte do material do aluno, sua construção começa a ser desenvolvida. O material é composto pelo cabeçalho, o qual é constituído de: nome da turma que participará da pesquisa, um espaço para ser inserido o nome do aluno, a data em que a atividade será realizada, um *feedback* qualificado a ser preenchido pelo professor, o nível de participação do aluno nas ADs, as tarefas a serem desenvolvidas e, para finalizar, um questionário geral. Vale ressaltar que sete ADs foram produzidas com base nos conteúdos trabalhados na disciplina de Programação I (AD1 - introdução aos conceitos básicos de programação, AD2 - comandos de seleção (*if...else*), AD3 - comandos de seleção (*switch...case*), AD4 - laços de repetição (*while)*, AD5 - laços de repetição (*do...while*), AD6 - laços de repetição (*for*) e, por último, AD7 - estruturas homogêneas (vetor)). Como todas as ADs possuem a mesma estrutura, abaixo será apresentada a AD3 desenvolvida na forma resumida (Figura 8), enquanto todas as ADs (AD1 – AD7) estarão nos apêndices B, C, D, E, F, G e H.

[14] Métodos e Processos de Ensino e Aprendizagem de Ciências (MPEAC). Registro no CNPQ: http://dgp.cnpq.br/dgp/faces/consulta/consulta_parametrizada.jsf

Figura 8 – Modelo resumido referente ao material apresentado ao aluno da AD3 - Comando switch...case.



TURMA INFO XX
Nome:_____
Data: ___/___/_____.
Feedback qualificado: _____.
Atividade Desenvolvida: ☐ completa ☐ parcial ☐ não realizada

Orientações: Todas as tarefas devem ser enviadas pelo ambiente de aprendizagem SIGAA[1].

Tarefa 1: Analisando o exercício abaixo, identifique a melhor estrutura estudada até o momento e construa o programa utilizando a linguagem de programação C. Justifique sua escolha.

a) Uma loja fornece 10% de desconto para funcionários e 5% de desconto para clientes vips. Faça um programa que calcule o valor total a ser pago por uma pessoa. O programa deverá ler o valor total da compra efetuada e um código que identifique se o comprador é um cliente comum, funcionário ou vip.

Na linguagem C:
Justificativa:

Tarefa 2: Analise os códigos abaixo e reescreva-os utilizando o comando Switch.... Case.

a)
```
#include <stdio.h>
#include<stdlib.h>
int main() {
int x;
printf("Escolha o codigo do produto\n");
printf ("1 - Vestuario\n");
printf ("2 - Higiene Pessoal\n");
printf("3 - Produto perecivel\n");
printf ("4 - Produto nao perecivel\n");
scanf("%d",&x);
if (x==1) {
printf ("Voce quer comprar uma blusa?\n");
}
 else
if(x==2) {
printf("Voce quer comprar um creme dental?\n");
} else
if(x==3) {
printf("Voce quer comprar um kg de carne?\n");
}else
if(x==4){
printf("Voce quer comprar uma lata de oleo ?\n");
} }
```

Reescrevendo o código:

b)
```
#include<stdio.h>

int main() {
 float compras, desconto, taxa, totpagar;
printf("Entre com o valor da compra");
scanf("%f",&compras);
if(compras <= 2000) {
 taxa=0.1; }
else {
 if (compras <= 3000) {
  taxa = 0.05;
 }
 else {
  if (compras <= 5000) {
   taxa = 0.03; }
  else{
   taxa=0.02;
 } } }
desconto = compras * taxa;
totpagar = compras - desconto;
printf("O seu desconto foi de %f e você ira pagar %f reais.", desco
nto, totpagar);
}
```

Reescrevendo o código:

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Aprendizagem:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Foco nos estudos:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
○ 1 (Nada trabalhosa) ○ 2 (Pouco trabalhosa) ○ 3 (Trabalhosa) ○ 4 (Muito trabalhosa) ○ 5 (Excessivamente trabalhosa)

Fonte: Da Autora (2021).

Além disso, é importante destacar que as ADs elaboradas são compostas por duas (AD1 e AD7) e três tarefas (AD2, AD3, AD4, AD5, AD6) que seguem determinados níveis de dificuldade: fácil, médio e difícil. Essa abordagem foi utilizada para que os alunos não fiquem desmotivados, desanimem ou deixem de realizar as próximas atividades, o que pode resultar em desistência ou abandono da disciplina, ou até mesmo do curso. Segundo Dorneles e Casa

(2004), é preciso pensar em metodologias que motive os alunos, fazendo-os tomar gosto pelo aprendizado e procurando superar suas dificuldades.

O número diferenciado nessas duas atividades se justifica pelo fato de a AD1 ser a primeira a ser desenvolvida junto aos alunos, mostrando a eles a importância de seu desenvolvimento e os estimulando a prosseguir mesmo diante das dificuldades iniciais encontradas no ensino-aprendizagem da disciplina de Programação I. A AD7 foi a última atividade a ser desenvolvida durante o ano letivo e um cansaço já era percebido nos alunos. Mesmo sendo a última atividade, era importante que eles a desenvolvessem em sua integridade.

4.3 IMPLEMENTAÇÃO DAS ADS

As ADs foram aplicadas no ano letivo de 2021, no Curso Técnico em Informática Integrado ao Ensino Médio de uma Instituição Federal do Estado do Rio Grande do Sul. Nesse ano, ingressaram 64 alunos pertencentes ao primeiro ano do curso.

Em 2021, o mundo se encontrava em um momento difícil da pandemia do Covid-19 e, no Brasil, as instituições de ensino foram mantidas fechadas, de modo que as aulas passaram a ser ministradas de forma remota. Conforme decidido no ano de 2020, enquanto a instituição estivesse fechada para a realização das aulas presenciais, as disciplinas seriam organizadas em módulos, cuja carga horária foi calculada de acordo com o número de horas de cada matéria. Os módulos eram ofertados de forma simultânea para os alunos e, conforme iam sendo finalizados, outros se iniciavam. Essa estratégia foi organizada pela instituição para que não houvesse um acúmulo de trabalhos a serem desenvolvidos pelos alunos, pois a presença era computada pela entrega das atividades semanais. Na disciplina de Programação I, por ter um elevado número de horas-aula (120h anuais), ficou destinado um tempo correspondente a dois meses e meio de aula em cada semestre.

Dentro dos módulos que estariam sendo ministrados, a instituição organizou um horário e um tempo para que as aulas síncronas fossem ministradas semanalmente. Na disciplina de Programação I, as aulas eram ministradas de forma síncrona duas vezes por semana, com um tempo de duração de 1h e 15 min, sendo que, no restante do tempo, o aluno tinha acesso, por meio do Sistema Integrado de Gestão de Atividades Acadêmicas (SIGAA[15]), aos materiais necessários para compor o seu aprendizado — como gravação das aulas dadas, exercícios, livros, entre outros —, complementando o conhecimento de forma assíncrona. A disciplina foi

[15] SIGAA – Sistema Integrado de Gestão de Atividades Acadêmicas utilizado na Instituição Federal onde a pesquisa foi desenvolvida.

organizada e o conteúdo dividido da seguinte forma: no primeiro semestre, trabalhou-se, inicialmente, com fluxogramas, comandos de decisão *(if...else e switch...case*) e a linguagem de programação C; no segundo semestre, foram abordados os conteúdos dos laços de repetição (*while, do...while* e *for*) e estrutura homogênea, vetor.

Como os momentos síncronos ocorriam duas vezes por semana, decidiu-se que, no primeiro encontro, o professor ministraria o conteúdo referente ao tópico a ser abordado naquele momento e, no final da aula, disponibilizaria uma lista de exercícios para os alunos resolverem. No encontro seguinte, o professor esclareceria as dúvidas que os alunos trouxessem e analisaria a necessidade de dispor mais tarefas antes de aplicar a AD referente ao conteúdo abordado. As ADs eram aplicadas no momento em que os alunos não relatavam mais dúvidas e conforme iam compreendendo o conteúdo. Isso oscilava de uma a duas semanas, dependendo do conteúdo a ser abordado, bem como sua complexidade.

Após essas etapas serem cumpridas, a AD era aplicada, sendo disponibilizada no horário da aula síncrona, e o aluno tinha até o final do dia para retornar sua avaliação pelo SIGAA. Essa alternativa foi implementada junto aos alunos porque muitos não podiam estar presentes no momento em que as aulas online estavam ocorrendo e, pelas normativas, os discentes não eram obrigados a comparecer nos encontros previstos no horário das aulas síncronas. Sendo assim, o professor deveria proporcionar outros momentos para os alunos cumprirem com suas obrigações escolares.

Posteriormente, o professor avaliava as respostas e elaborava o *feedback* qualificado para cada aluno, para que na aula seguinte um retorno fosse dado aos discentes, realizando uma retomada nas principais lacunas observadas. Após as dúvidas serem esclarecidas, outras atividades eram disponibilizadas aos alunos, sendo que o foco principal dessas atividades estavam justamente em proporcionar aos alunos uma nova oportunidade de verificar seu aprendizado.

Após realizar essas etapas, um novo conteúdo era ministrado pelo professor e o ciclo de implementação descrito voltava para seu início.

4.4 AVALIAÇÃO

Concorda-se com os pesquisadores Wang e Hannafin (2005) ao afirmarem que a avaliação de uma metodologia que utiliza EDR deve ser constante, ou seja, durante todo o processo de desenvolvimento e aplicação. Dessa forma os dados obtidos foram sendo

continuamente analisados durante o desenvolvimento dos ciclos iterativos e ao final do trabalho.

O *Design* Metodológico proposto nesta tese possui várias etapas, já discutidas anteriormente, dessa forma cada etapa possui objetivos diferentes e, portanto, análises de dados e resultados diferentes também. Nessa proposta foi utilizada a análise quantitativa, que conforme Michel (2008) é um método de pesquisa social que utiliza a quantificação nas modalidades de coleta de informações e no seu tratamento, mediante técnicas estatísticas, tais como percentuais, média, desvio-padrão, coeficiente de correlação, análise de regressão, entre outros. Normalmente esse método implica na construção de inquéritos por questionário.

Assim, ao final do ano letivo, um questionário geral de avaliação (apêndice A) foi aplicado a fim de verificar o nível de satisfação dos alunos com a proposta desenvolvida e se corroboraram com os dados extraídos das ADs desenvolvidas pelos alunos. Sendo assim, no intuito de demonstrar os diferentes pontos de avaliações, este trabalho contempla as avaliações dos seguintes segmentos: da nova metodologia, do desempenho dos alunos e uma comparação do método proposta com a utilizada nos anos anteriores: 2019 e 2020.

4.4.1 Avaliação da nova metodologia

Para analisar a nova metodologia desenvolvida, foi verificada a percepção dos alunos quanto às ADs desenvolvidas por meio do questionário elaborado e aplicado a cada AD desenvolvida. O questionário considera os seguintes itens: empenho na disciplina, aprendizagem, foco nos estudos e trabalho despendido na AD. Esses itens permitem analisar, respectivamente: o interesse do aluno em relação ao desenvolvimento das ADs; o entendimento dos alunos frente aos assuntos abordados; a evolução no ritmo de estudo; e o tempo despendido para realizá-las, analisando se as ADs estão adequadas, pois a qualidade das ADs é importante para o processo ser bem-sucedido. Além de realizar as respectivas análises, os itens considerados permitem auxiliar o professor a construir a próxima atividade e verificar se a percepção dos alunos corrobora com as ADs desenvolvidas. Além disso, torna-se possível investigar a viabilidade didática das atividades construídas segundo a perspectiva do professor por meio da evolução apresentada pelos alunos.

Para finalizar o estudo, uma análise qualitativa foi realizada através da aplicação de um questionário geral (apêndice A). Nele os alunos avaliavam a qualidade das atividades, seu desempenho, o número de tarefas dadas em cada AD e o número de ADs desenvolvidas, a contribuição em relação ao aproveitamento, ao ritmo de estudo, ao *feedback* qualificado, ao

ensino remoto e a destreza do aluno para desenvolver os programas propostos. Depois de todo o processo ser concluído, temos um ciclo completo implementado e, assim, é possível, por meio dos dados coletados, analisar a viabilidade da proposta.

4.4.2 Desempenho dos alunos com a nova metodologia proposta

Após a construção e implementação das ADs, foi desenvolvido o material de análise, denominado matrizes de análise (Tabela 1). Esse material auxilia o docente a realizar o *feedback* e analisar o desempenho e o desenvolvimento dos seus alunos de acordo com os critérios estabelecidos. Com o propósito de construir a matriz de análise, a Tabela 1 foi desenvolvida para as turmas cujos dados coletados foram analisados a partir dos objetivos elencados dentro de cada AD no momento de sua elaboração, levando em consideração as 4 habilidades avaliadas nessa tese. Essas habilidades serão identificadas por escalas qualitativas e quantitativas, utilizando os conceitos "satisfatório" (maior que 70%), "parcialmente satisfatório" (50 – 69%) e "insatisfatório" (abaixo de 50%). Também será inserido, em todas as análises, um critério que sinaliza a existência da devolutiva da atividade pelo aluno, que poderá ser parcial, completa ou não realizada.

Ainda é possível perceber que, na Tabela 1, o professor avalia o nível de compreensão e entendimento dos alunos de acordo com cada habilidade: (I) interpretação de texto, (A) abstração, (RL) raciocínio lógico e (NM) noções matemáticas. Dessa maneira, é possível acompanhar cada aluno nos diferentes graus de percepção necessários à aprendizagem de programação.

Tabela 1 – Matriz de análise das ADs.

	Interpretação de texto - I			Abstração - A			Raciocínio lógico - RL			Noções matemáticas - NM		
Alunos	**S**	**PS**	**I**	**S**	**PS**	**I**	**S**	**PS**	**I**	**S**	**PS**	**I**
Aluno 001												
Aluno 002												
Aluno 003												
Aluno 004												
Aluno 005												
Resultado por critério num total de XX alunos	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**
Devolutiva dos alunos	**C** **0%**	**P** **0%**	**NR** **0%**	**C** **0%**	**P** **0%**	**NR** **0%**	**C** **0%**	**P** **0%**	**NR** **0%**	**C** **0%**	**P** **0%**	**NR** **0%**

S – Satisfatório; PS – Parcialmente Satisfatório; I – Insatisfatório
Fonte: Da Autora (2021).

Ao extrair as informações da Tabela 1, o professor irá realizar uma média das tarefas 1, 2 e 3 em todas as ADs levando em consideração as quatro habilidades analisadas. Após, uma comparação entre os resultados das respectivas tarefas será realizada, analisando os índices alcançados pelos alunos.

4.4.3 Comparação da nova metodologia com a metodologia anterior

Conforme Matta, Silva e Boaventura (2014), a EDR propõe que cada vez que um ambiente de aprendizagem ou intervenção tenha sido projetado e desenvolvido, o passo seguinte será a implementação e avaliação da intervenção em ação. Dessa forma, a pesquisa desenvolvida visa, por meio de uma comparação de metodologias utilizadas nos anos de 2019 e 2020 no contexto de aplicação, realizar uma análise quantitativa referente aos índices de aprovação, reprovação, evasão e média escolar alcançadas nos últimos três anos.

É importante ressaltar que, para comparar as médias das avaliações, foi necessário realizar uma equalização nas avaliações realizadas, já que o número de avaliações desenvolvidas em 2021 durante o desenvolvimento da pesquisa foi maior do que nos anos anteriores. Após, foi realizada uma comparação entre as avaliações analisadas (AV1, AV2, AV3 e TF), entre as médias finais dos alunos nessas mesmas avaliações, na frequência com que as avaliações foram desenvolvidas, no número de alunos aprovados com exame, aprovados sem exame e reprovados.

Assim, por meio desses índices, será possível analisar se os fatores ritmo de estudo e *feedback* qualificado foram importantes no processo de ensino-aprendizagem desta pesquisa, analisando sua eficácia perante o contexto analisado.

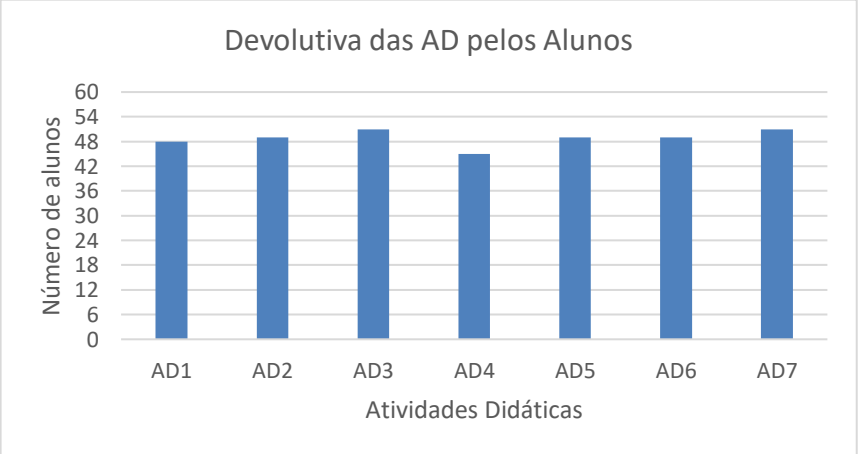
5. RESULTADOS E DISCUSSÕES

No intuito de melhor dispor os dados coletados durante a pesquisa, bem como agrupá-los e discuti-los, os resultados a serem apresentados estão organizados em três partes: i) avaliação da metodologia, em que serão analisada: a implementação, o índice de participação dos alunos e o índice de resolução das atividades enviadas; ii) avaliação de cada AD e uma avaliação geral do conjunto das ADs desenvolvidas; e iii) desempenho dos alunos com a metodologia proposta e comparação da metodologia desenvolvida com as metodologias usadas em anos anteriores (com e sem pandemia).

5.1 AVALIAÇÃO DA METODOLOGIA

Os procedimentos metodológicos propostos foram implementados conforme descrito anteriormente. Além disso, uma conversa explicando detalhadamente como os procedimentos seriam abordados foi fundamental para o engajamento dos alunos, pois no primeiro dia de aula surgiram diversas dúvidas sobre como seria o andamento do processo ensino-aprendizagem, principalmente em relação ao número de avaliações, a forma avaliada e os *feedbacks* qualificados. Assim, aconteceu um debate entre o professor e os alunos com a finalidade de demonstrar as dificuldades verificadas em anos anteriores, culminando com a ênfase de como seria esperado a evolução dos alunos. Com o propósito de analisar o índice de participação dos estudantes nas ADs, foi considerado o envio das atividades pelo ambiente SIGAA. A Figura 9 ilustra o número de alunos que desenvolveram as ADs no decorrer da pesquisa.

Figura 9 – Participação dos alunos em cada atividade didática no ambiente SIGAA.

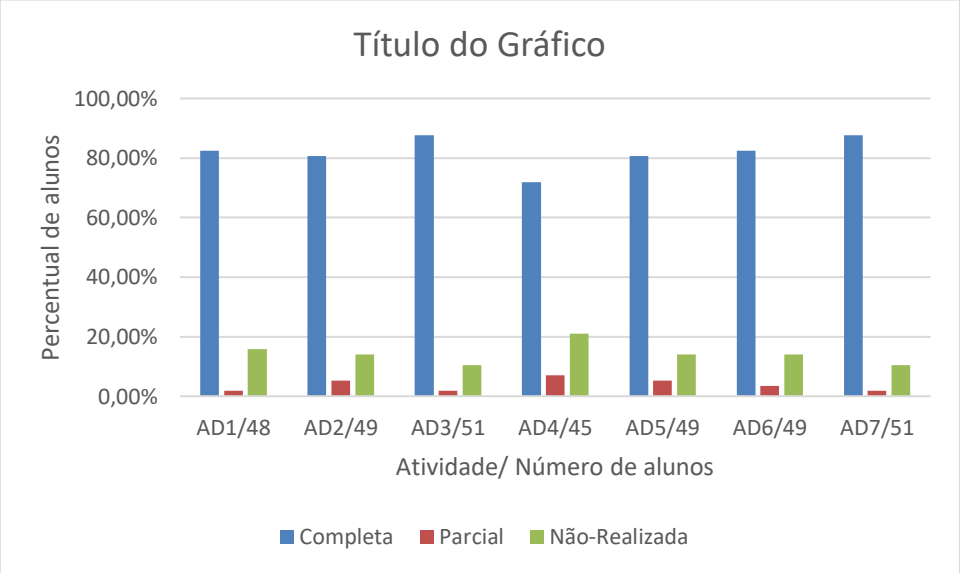


Fonte: Da Autora (2022).

A fim de analisar os dados, calculou-se a média de participação dos alunos em cada AD, a qual ficou em 48,86 discentes de um total de 57 participantes. Também se computou a moda (M_0 = 49) e verificou-se que o número de participantes em cada AD é o mesmo da média, sinalizando uma homogeneidade entre os dados. Por último, analisou-se o desvio padrão, no intuito de verificar o grau de variação das devolutivas e evidenciou-se que houve uma pequena variação (DP = 1,88). Analisando a Figura 9 e calculando a média, moda e desvio padrão das devolutivas dos alunos, observa-se que somente na AD4 ocorre uma pequena variação, pois o número de alunos participantes dessa atividade está um pouco abaixo do desvio padrão calculado. Isso ocorreu quando os alunos se depararam com uma estrutura mais complexa — laço de repetição (*while*) — e junto a ela foi necessário utilizar comandos condicionais, aumentando a complexidade dos programas a serem desenvolvidos. Contudo, no geral, ocorreu um alto índice de participação nas ADs (média ~ 87%), permitindo que o professor tivesse uma percepção mais eficiente quanto ao rendimento do aluno e identificasse as lacunas peculiares para retomar o(s) diferente(s) conteúdo(s) específicos, individualmente ou em grupo.

Após, foi verificado o índice de participação dos alunos em relação ao desenvolvimento das ADs, onde a devolutiva de cada AD foi classificada como completa, parcial e não-realizada. A Figura 10, mostra os índices de alunos nas respectivas classificações.

Figura 10 – Devolutivas de cada AD.

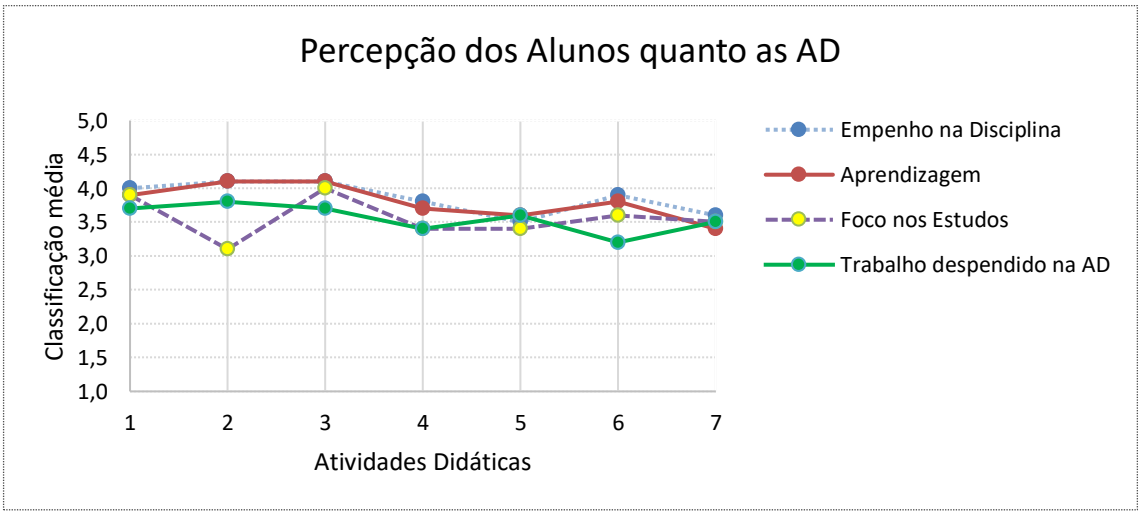


Fonte: Da Autora (2022).

Ao analisar a Figura 10, percebe-se que o índice de alunos que desenvolveram cada AD em sua totalidade não baixou de 80% (média = 81,93%), com um DP = 4,82%, sendo a AD4 a única exceção, a qual teve o índice de devolutiva menor (71,90%). Além disso, a devolutiva parcial por parte dos estudantes possui uma média muito baixa (3,75%), indicando que os alunos conseguem compreender e responder quase todas as ADs, proporcionando ao estudante verificar seu aprendizado em todas as habilidades que estão sendo analisadas.

A fim de avaliar a percepção dos alunos quanto às ADs desenvolvidas, foi elaborado e aplicado um questionário (Figura 7) considerando os seguintes itens: empenho na disciplina, aprendizagem, foco nos estudos e trabalho despendido na AD. Esses itens permitem analisar, respectivamente, o interesse do aluno em relação ao desenvolvimento das ADs, o entendimento dos alunos frente aos assuntos abordados, a evolução no ritmo de estudo e o tempo despendido para realizá-las. Além das respectivas análises, os itens considerados permitem auxiliar o professor a construir a próxima atividade e verificar se a percepção dos alunos corrobora com as ADs desenvolvidas.

Figura 11 – Gráfico que demonstra a variação de Empenho, Aprendizagem, Foco nos Estudos e Trabalho Despendido nas ADs.



Fonte: Da Autora (2022).

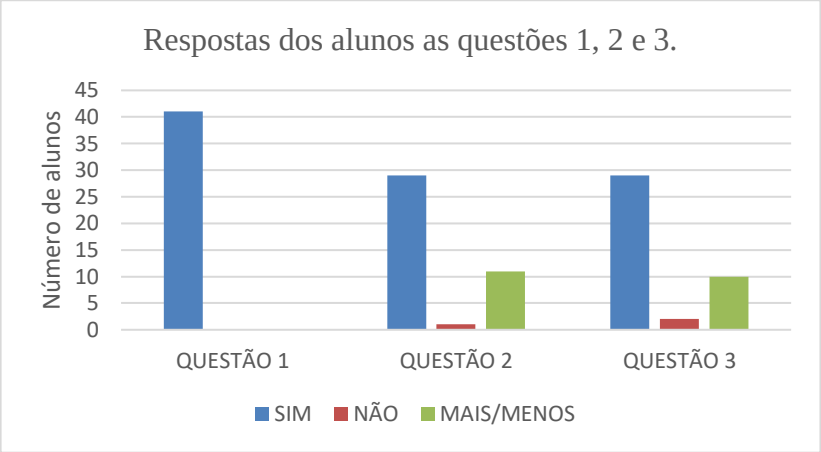
Observando as respostas dos alunos, conforme a Figura 11, distribuídas em 342 questionários, verifica-se que o trabalho despendido nas ADs possui uma correlação com as outras percepções dos alunos da AD1 até a AD4, mas, a partir da AD5, evidencia-se uma oscilação em seu comportamento. Isso corrobora em duas visões que podem ser observadas pelo professor: o tempo despendido para realizar a tarefa e o nível de dificuldade do aluno

perante cada AD. Com isso, da AD1 à AD4, sugere-se que a variação do trabalho foi influenciada pelo tempo despendido, pois, quanto maior o tempo, maior foi o empenho e a aprendizagem. No entanto, observa-se uma queda na variação quando se compara os resultados obtidos da AD4 até a AD7 com os resultados obtidos nas três primeiras avaliações (AD1 – AD3), já que, a partir da AD4, os estudantes tiveram o primeiro contato com a nova estrutura (laços de repetição), em que o nível de dificuldade foi maior porque, nesse momento, problemas mais complexos começam a ser desenvolvidos. Além disso, se considerarmos estatisticamente uma variação de 5% (para mais ou para menos) entre a AD4 e a AD7, somente observa-se uma pequena queda na variação da AD6 referente ao trabalho despendido. Já na AD7, nova estrutura, novamente o trabalho despendido aumenta e o empenho e aprendizagem voltam a cair.

Quanto ao foco, sua percepção é muito parecida com as percepções de empenho e aprendizagem e trabalho despendido, mostrando que o aluno mais focado tende a ter uma melhor aprendizagem e um melhor empenho nas ADs. A exceção é a atividade AD2, pois nela os comandos referentes à linguagem de programação C foram apresentados ao aluno pela primeira vez. Ao sentir uma maior dificuldade no conteúdo trabalhado, o aluno teve um menor foco em realizar a AD, mas ainda assim conseguiu manter o empenho e a aprendizagem. Pode-se concluir que a diminuição dos índices nesse item, possivelmente, está correlacionada com a dificuldade dos discentes quanto aos tópicos trabalhados. Assim, verifica-se que, nas próximas vezes, quando esses assuntos forem discutidos, deve-se usar metodologias complementares de aprendizagem. Desse modo, todos esses quesitos nos possibilitaram fazer ajustes na forma do ensino-aprendizagem das temáticas abordadas.

Após verificar a percepção dos alunos no desenvolvimento de cada AD, foi aplicado um questionário geral onde o conjunto de atividades foi analisado pelos discentes. O questionário geral mantinha as mesmas percepções dos questionários referentes a cada AD individual, mas agora com um maior espectro de percepções dos alunos sendo avaliadas. Contudo, foi necessário que os alunos descrevessem suas respostas, justificando-as. O questionário é composto por nove perguntas abertas (apêndice A). Responderam ao questionário 41 alunos, em que as questões foram relacionadas conforme os questionamentos realizados. As questões 1, 2 e 3 tinham como propósito investigar os alunos em relação ao ritmo e foco nos estudos, desempenho e elucidação de suas dúvidas. Essas três questões foram relacionadas entre si, conforme ilustra a Figura 12.

Figura 12 – Resposta dos alunos referente as questões trabalhadas em cada AD.

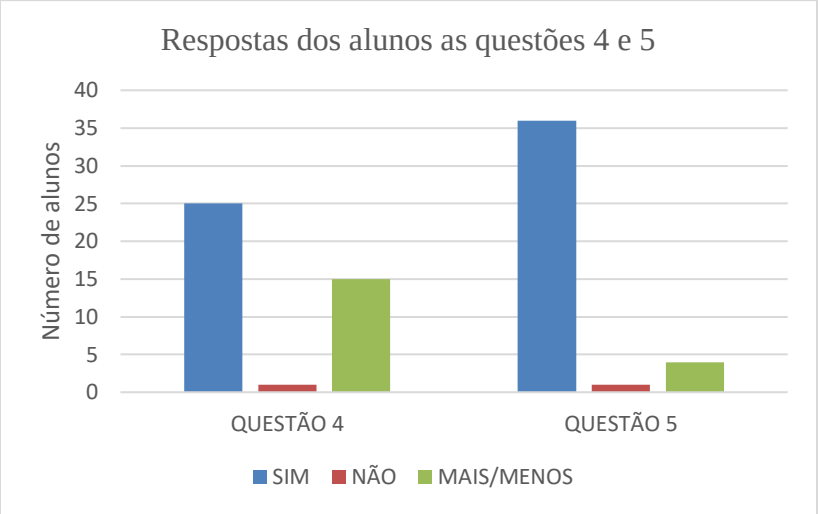


Fonte: Da Autora (2022)

Ao analisar as respostas encontradas, observou-se que todos os alunos que responderam ao questionário disseram que o conjunto de ADs ajudou a manter o foco e ritmo nos estudos e, com isso, melhorou o desempenho na disciplina, pois suas dúvidas eram esclarecidas e isso colaborava para um aprendizado contínuo. Entre os comentários dos alunos destaca-se as respostas referente ao seguintes aspectos: i) ritmo e foco nos estudos: “Sim, acredito que sem elas não teríamos buscado entender melhor o conteúdo a fim de resolver as atividades”; ii) ao desempenho: “Melhorou bastante, com as atividades adquirimos mais foco e o ritmo melhorou muito, acredito que não só o meu, mas de toda a turma”; e iii) ao retorno qualificado e a revisão realizada: “Sim, é como citei sobre ser algo que se aprende na prática, resolvendo, errando e corrigindo, isso faz com que vejamos onde está o erro e compreendamos melhor a disciplina”.

Já as perguntas 4 e 5 indagavam os alunos em relação ao número de tarefas de cada AD e o número de ADs desenvolvidas durante o ano letivo (Figura 13).

Figura 13 – Percepção dos alunos ao número de tarefas por AD e ao total de ADs desenvolvidas durante o ano letivo.

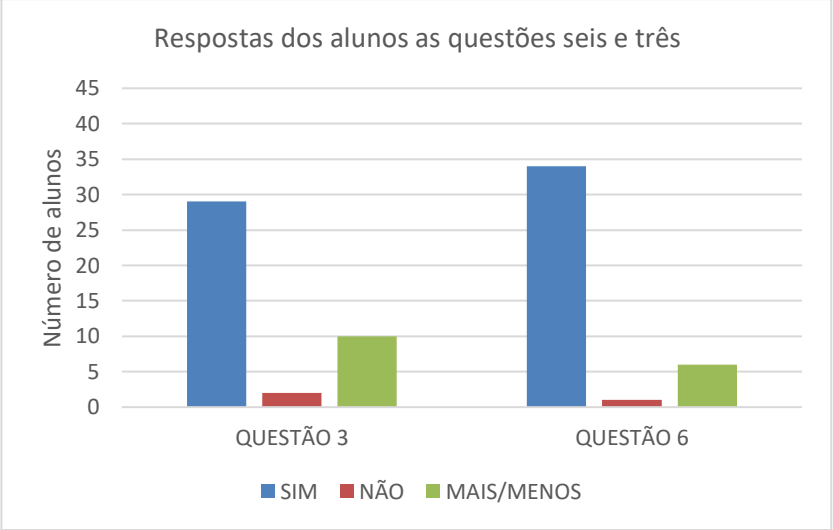


Fonte: Da Autora (2022).

Por meio dos dados coletados, percebeu-se que a maioria dos alunos entendeu que o número de tarefas em cada AD estava adequado, mas uma parcela deles considerou que podiam ser menos atividades, pois tinham outros trabalhos para serem desenvolvidos em outras disciplinas, e o tempo ficava curto para realizá-los. Já com relação ao número de ADs desenvolvidas durante o ano letivo, os discentes compreenderam que estava adequado. Entre os comentários dos alunos referentes ao número de tarefas por AD, cita-se: “Sim, isso é importante, mas às vezes seria melhor tentar auxiliar as dificuldades em mais encontros via *meet*, pois o grande número de atividades, mesmo sendo bom para esclarecer bem a matéria, acaba pesando para os alunos que tem várias atividades em todas as matérias”; e, com relação ao conjunto de ADs desenvolvidas, relata-se: “Sim. Ao incluir uma avaliação por conteúdo de programação, fez com que eu desenvolvesse melhor minha habilidade com aquele conteúdo e não uma coisa “meia boca” sobre cada um deles”.

A questão 6 enfatiza o retorno qualificado e a revisão desenvolvida em cima das dificuldades apresentadas, podendo relacionar essa pergunta com a questão 3, referente às dúvidas encontradas na disciplina. A Figura 14 traz os dados referente a essa relação.

Figura 14 – Respostas as questões 6 e 3.

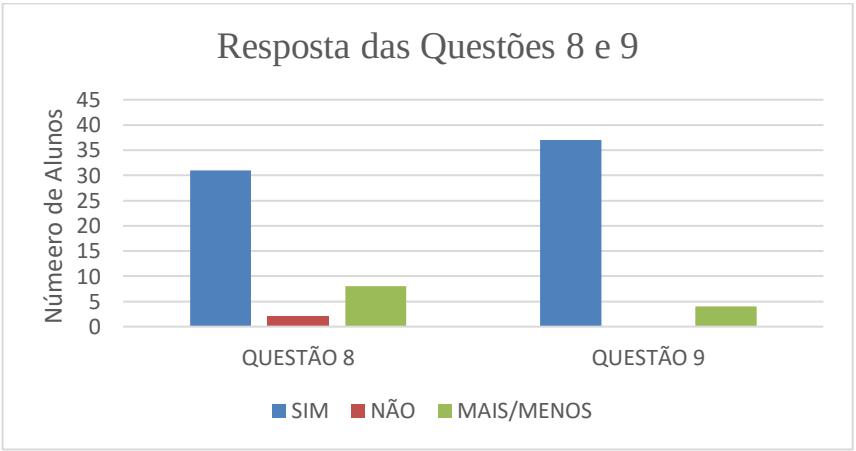


Fonte: Da Autora (2022).

Essas questões relacionam-se entre si, uma vez que ao mesmo tempo, a maioria dos alunos relata que o retorno qualificado, a revisão desenvolvida e o esclarecimento das dúvidas encontradas na disciplina foram pontos importantes que auxiliaram no processo de ensino-aprendizagem da disciplina. Os alunos, ao comentarem as questões 3 e 6, relataram: “Sim, foram muito boas para treinar os estudos sobre os conteúdos”; “Sim, pois era durante as correções das atividades que eu via onde eu estava encontrando dificuldades no conteúdo”.

Já nas questões 8 e 9, os discentes tinham que responder se a metodologia proposta contribuiu para o ensino-aprendizagem e sua importância durante o ensino remoto (Figura 15).

Figura 15 – Número de alunos que responderam as questões 8 e 9.

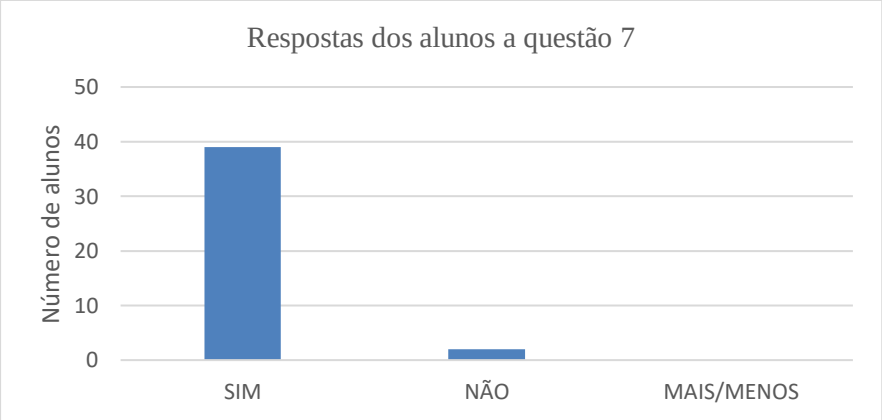


Fonte: Da Autora (2022).

A maioria dos alunos na questão 8 e 9, 30 e 37 alunos, respectivamente, entenderam que tanto a metodologia proposta foi considerada importante como sua aplicação durante o ensino remoto. Entre os relatos dos alunos destaca-se: “Sim, por ser um conteúdo que nunca havia visto, no começo era difícil, mas com a metodologia que foi adotada, com todas as atividades, provas e aulas, isso mudou e passou a ser algo mais fácil.”; “Sim, me desenvolveu muito minha interpretação na hora de ler os programas.”

Por fim, a última questão a ser analisada, questão 7, refere-se à evolução dos discentes em relação a sua destreza para programar. A Figura 16, ilustra que a maioria dos alunos (39) consideraram que sim, as ADs contribuíram muito para que o desenvolvimento de seus códigos passasse a ser realizados de forma mais rápida, pois, com as dúvidas sendo sanadas, tiveram uma maior agilidade em suas resoluções.

Figura 16 - Número de alunos que responderam à questão 7.



Fonte: Da Autora (2022).

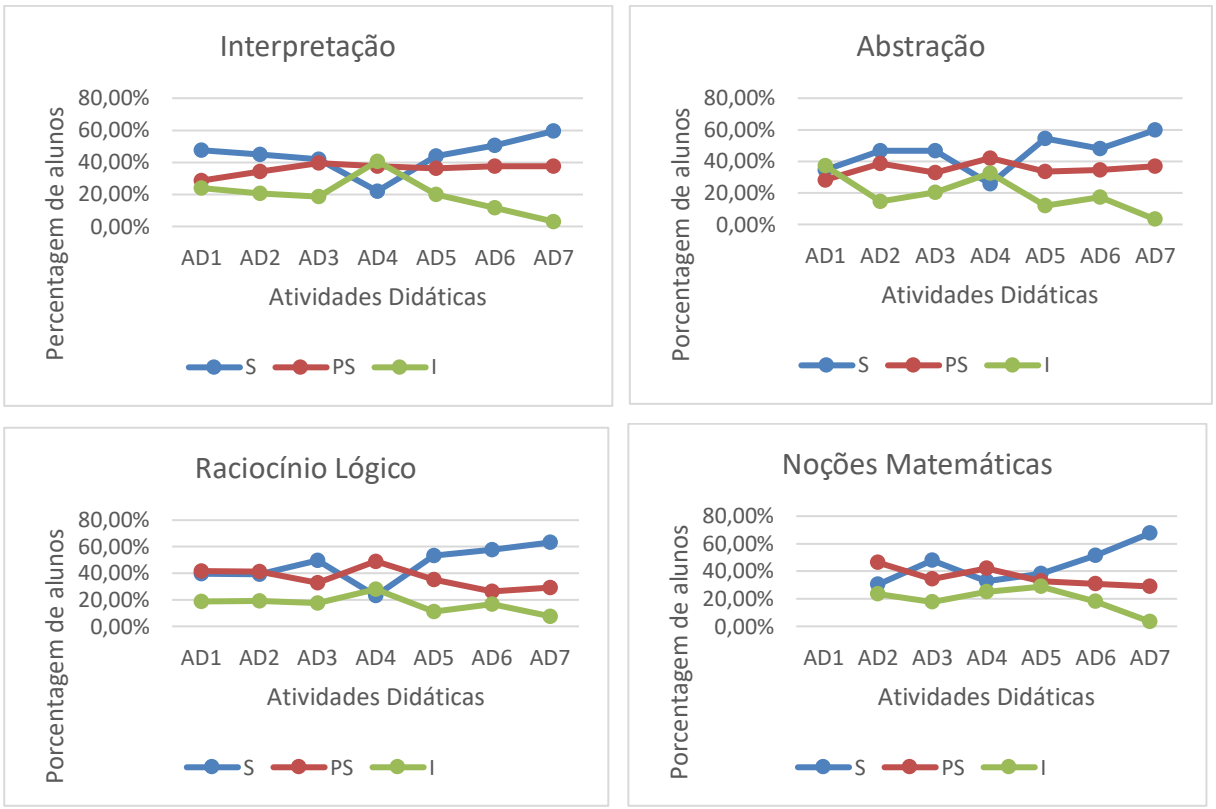
Para finalizar a análise em relação a metodologia desenvolvida, é necessário recordar que uma das propostas a serem inseridas na construção das ADs era o uso imediato de uma linguagem de alto nível: nesse contexto, a Linguagem C. A Linguagem C foi utilizada sem que os alunos tivessem contato com o pseudocódigo, sendo desenvolvidas tarefas específicas para cada estrutura estudada – condicional (*switch...case*), laço de repetição (*do...while*) e estrutura homogênea (vetor). Nessas tarefas, os alunos recebiam um programa em pseudocódigo e tinham que transcrevê-lo para a Linguagem C, sem ter sido trabalhado o pseudocódigo em sala de aula. Ao verificar o índice de acertos dessas tarefas, que ficou em 91% na tarefa *switch...case*, 94% na tarefa *do...while* e 98% na tarefa vetor, constatou-se que os alunos podem sim iniciar o

desenvolvimento de seus problemas utilizando uma Linguagem de alto nível sem prejudicar seu aprendizado na disciplina.

5.2 DESEMPENHO DOS ALUNOS COM A METODOLOGIA PROPOSTA

Para analisar a capacidade das ADs na aprendizagem dos alunos segundo as habilidades interpretação de texto, abstração, raciocínio lógico e noções matemáticas, foi realizada uma média geral das tarefas abordadas em cada AD, em que esses critérios foram sendo trabalhados e desenvolvidos por meio dos *feedbacks* qualificados. Além disso, os dados serão discutidos de acordo com as habilidades, com base na faixa de desempenho dos alunos: satisfatório (S), parcialmente satisfatório (PS) e insatisfatório (I). Primeiramente, informa-se que na AD1, referente à competência denominada interpretação, a tarefa 1 foi desconsiderada porque não havia texto para ser interpretado. Além disso, pelos mesmos motivos da tarefa 1 na AD1 da habilidade Interpretação, a AD1 (tarefa 1 e 2) referente a competência denominada Noções de Matemática, não foi avaliada. Também é importante salientar que o nível de dificuldade entre as tarefas desenvolvidas era diferente, sendo considerado: fácil, médio e difícil, respectivamente. É importante que, em um primeiro contato com a AD, o aluno perceba uma certa facilidade em sua resolução que o estimule a continuar a desenvolver as tarefas seguintes. A Figura 17 ilustra os resultados obtidos nos indicadores “satisfatório”, “parcialmente satisfatório” e “insatisfatório”, referente à média das tarefas em todas as ADs em relação às quatro habilidades analisadas.

Figura 17 – Gráfico da média das tarefas abordadas em todas as ADs levando em consideração as quatro habilidades analisadas.



Fonte: Da Autora (2022).

Em relação ao indicador satisfatório (S), as habilidades interpretação, abstração, e raciocínio lógico, mostram tendências parecidas, com a habilidade caindo da AD3 para AD4 e, posteriormente, na AD4, observa-se um decréscimo dos alunos nessas habilidades, voltando a aumentar o índice nas próximas ADs. No entanto, as variações, a partir da AD1 até a AD3, são diferentes nessas habilidades, pois, na abstração, ocorre um aumento dessa percepção pelos alunos da AD1 – AD2, ocorrendo uma posterior estabilização da AD2 – AD3; enquanto na interpretação, ocorre uma pequena diminuição gradual da AD1 – AD3 e, por fim, no raciocínio lógico, observa-se uma estabilização na variação da AD1 – AD2, com posterior aumento da AD2 – AD3. Dessa maneira, consegue-se perceber que as variações das abstrações, interpretações e raciocínios lógicos estão correlacionadas em várias das ADs avaliadas.

Em relação ao indicador parcialmente satisfatório (PS), as habilidades de abstração e interpretação possuem uma mesma tendência, com pouca variação quando comparados entre si. Essas habilidades possuem os valores de 28, 25 e 28,69%, referentes a AD1 e as respectivas variações 29,38 – 42,10 e 32,00 – 39,25%, referentes aos intervalos da AD2 – AD7. Outra

correlação que pode se observar do indicador PS, refere-se as habilidades noções matemáticas e raciocínio lógico, já que nessa variação de desempenho dos alunos correspondentes a essa faixa de aprendizado, observa-se uma diminuição da AD2 – AD3, aumentando da AD3 – AD4 e, posteriormente, tendo uma tendência de diminuição da AD4 – AD7.

Quanto ao indicador insatisfatório (I), as habilidades abstração, interpretação, raciocínio lógico e noções matemáticas possuem tendência de diminuição acentuada a partir das AD4, observando-se uma baixa porcentagem de alunos nas ADs seguintes e obtendo um número baixíssimo na AD7. No entanto, a única exceção é a habilidade de noções matemáticas, na qual essa diminuição ocorre somente a partir da AD5. De uma maneira geral, em todas as ADs, o indicador "satisfatório" foi o mais observado entre os alunos e, consequentemente, esses alunos atingiram maiores níveis de aprendizado, com exceção da AD4, cujo indicador PS foi o mais observado. Foi apresentado aos discentes, pela primeira vez, a estrutura de laço de repetição (*while*) e eles tiveram dificuldade em compreender o modo de funcionamento desse comando.

5.3 COMPARAÇÃO DA METODOLOGIA PROPOSTA COM METODOLOGIAS USADAS EM ANOS ANTERIORES

A metodologia proposta apresentou resultados satisfatórios referentes aos desempenhos dos alunos, mas, no intuito de comprovar a sua eficácia, precisa-se comparar com outras metodologias aplicadas no mesmo contexto acadêmico. Para isso, mostra-se a comparação da metodologia proposta com as metodologias aplicadas em anos anteriores, como em 2019 (sem pandemia Covid-19) e em 2020 (com pandemia Covid-19). Inicialmente, as metodologias aplicadas em 2019 e 2020 serão discutidas abaixo para posterior comparação.

A metodologia aplicada em 2019 se baseou no "método tradicional", em que o professor ministrava os encontros através de aulas expositivas-dialogadas, em seguida realizava exercícios para verificar a aprendizagem e, de tempos em tempos, realizava avaliações práticas individuais (tempo de execução: 90min) e trabalhos em dupla (igualmente, tempo de execução: 90min). Acreditava-se que, ao disponibilizar várias listas de atividades, os alunos conseguiriam desenvolvê-las e, com isso, melhorar seu desempenho, mas o que acontecia era um acúmulo de exercícios não realizados e, ao desenvolver as avaliações, o aluno confundia estruturas e comandos, tendo um rendimento muito abaixo do esperado.

A metodologia utilizada em 2020 se baseou na metodologia aplicada em 2019, mas como o mundo se encontrava numa pandemia, as instituições tiveram que adaptar aulas e avaliações. As aulas síncronas e assíncronas foram trabalhadas como descritas na metodologia proposta em

2021, mas com diferenças muito consideráveis na condução do ensino-aprendizagem, pois os alunos mantiveram suas rotinas diárias de aulas e estudos em suas próprias residências. O processo de avaliação também foi modificado, naquele momento o aluno estava em casa e tinha o material no ambiente SIGAA, além da liberdade de fazer consultas na internet. Isso levou o professor a desenvolver questões mais elaboradas nas avaliações, nas quais os discentes precisam analisar situações-problemas mais complexos ao desenvolver suas próprias soluções. Além disso, é importante ressaltar que, além de realizar suas provas em suas casas, os alunos possuíam um prazo maior para devolvê-la ao professor (24h). Quanto aos trabalhos, eles continuaram sendo realizados em duplas, em que os alunos se reuniam fora do horário de aula para desenvolvê-los, pois, os trabalhos também tinham o propósito de integrar os alunos, visto que não tinham contato presencial com os colegas.

Em 2021, o mundo ainda se encontrava numa pandemia e a evasão e a reprovação tinham aumentado, com isso, a metodologia descrita para ser implementada no ano de 2021 foi aplicada, trazendo muitas diferenças em relação as metodologias desenvolvidas em 2019 e 2020. Dentre as principais diferenças, destacam-se: a maior frequência de avaliações, a presença de um *feedback* qualificado (baseado nas habilidades) e, consequentemente, um maior ritmo de estudos dos alunos. Além de ajudar no processo de ensino, o modelo desenvolvido tem como propósito melhorar a comunicação com os alunos, pois, com as avaliações sendo realizadas mais frequentes, os *feedbacks* qualificados e a retomada dos conteúdos, estimulam os alunos a participarem mais dos momentos síncronos realizados, interagindo mais com o professor.

No intuito de demonstrar as diferenças entre as metodologias, o Quadro 3 abrange os pontos relevantes de cada uma delas.

Quadro 3 - Pontos relevantes nas metodologias usadas em 2019, 2020 e 2021.

Diferenças	**Procedimentos Metodológicos 2019**	**Procedimentos Metodológicos 2020**	**Procedimentos Metodológicos 2021**
Aulas Presenciais	Sim	Não	Não
Aulas ER	Não	Síncronas e Assíncronas	Síncronas e Assíncronas
Número de Avaliações Práticas	3	3	7
***Feedback* do Professor**	Não	Não	Sim

Fonte: Da Autora (2022).

No intuito de comparar as três metodologias, realizou-se uma equalização referente ao desempenho dos alunos em cada proposta, pois nos anos de 2019 e 2020 foram realizadas três avaliações envolvendo atividades práticas (individuais) e dois e três trabalhos (duplas), respectivamente. No ano de 2019, as avaliações da aprendizagem e os trabalhos foram realizadas presencialmente em sala de aula. Já no ano de 2020, devido a pandemia do Covid-19, as mesmas avaliações foram realizadas fora do ambiente escolar. Devido a esse desafio de melhorar o ensino-aprendizagem, a metodologia aplicada em 2021 foi proposta com nove avaliações (sete avaliações individuais e duas avaliações em duplas). Dessa maneira, a equalização da avaliação procedeu-se de acordo com a Quadro 4.

Quadro 4 - Equalização das avaliações realizadas em 2019, 2020 e 2021

2019	**2020**	**2021**	**Conteúdos**
AV1	AV1	AD1	Introdução aos conceitos básicos de programação
		AD2	Comando de seleção - *if...else*
		AD3	Comando de seleção - *switch...case*
AV2	AV2	AD4	Laço de repetição - *while*
		AD5	Laço de repetição – *do...While*
		AD6	Laço de repetição - *for*
AV3	AV3	AD7	Estrutura homogênea - vetor
Trabalho Final	Trabalho Final	Trabalho Final	Todo o conteúdo

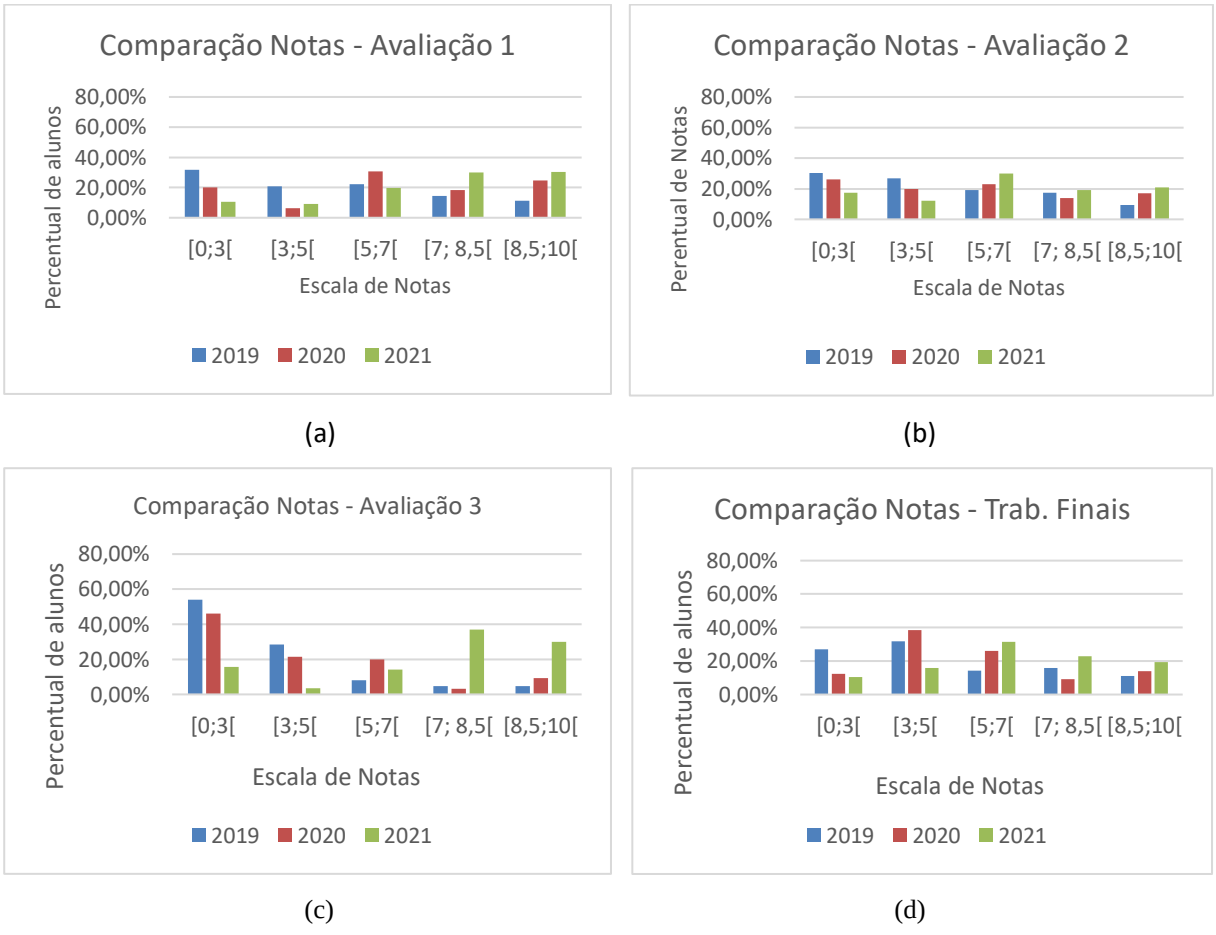
Fonte: Da Autora (2022).

Sendo assim, a média das avaliações práticas denominadas de AD1, AD2 e AD3 equivalem à avaliação denominada de AV1 nos outros anos, enquanto as avaliações práticas denominadas AD4, AD5 e AD6 equivalem a avaliação denominada AV2. A última avaliação prática, denominada AD7, equivale a avalição denominada AV3. Vale ressaltar que as AV's e as ADs sempre abordaram a totalidade do conteúdo trabalhado até o momento de cada avaliação, pois não se pode esquecer que cada conteúdo abordado servirá de conhecimento prévio a ser usado na continuação do aprendizado dos conteúdos posteriores.

Na equalização das notas referente aos trabalhos, somente foi possível equalizar os trabalhos que abrangeram a totalidade dos conteúdos, sendo denominados nos respectivos anos de T2 (2019), T3 (2020) e T2 (2021), mas, nesta tese, chamamos de Trabalho Final. Em 2019, o T1 abrangeu os seguintes conteúdos: introdução aos conceitos básicos de programação, comandos de seleção (*if...else* e *switch...case*) e laços de repetição (*while*, *do...while* e *for*); já

em 2020, o T1 abrangeu os seguintes conteúdos: introdução aos conceitos básicos de programação e comandos de seleção (*if...else* e *switch...case*), o T2 abrangeu os laços de repetição (*while*, *do...while* e *for*); e, em 2021, o T1 a abrangeu apenas a introdução aos conceitos básicos de programação e os comandos de seleção (*if...else* e *switch...case*), não sendo possível equalizar e comparar o desempenho dos alunos nesses trabalhos, conforme a Figura 18.

Figura 18 – Desempenhos equalizados das ADs e do trabalho final, referentes aos anos de 2019, 2020 e 2021.



Fonte: Da Autora (2022).

Primeiramente, serão analisados os resultados de cada avaliação separadamente e depois, o desempenho ao longo do ano. Assim, comparando as notas das avaliações 1, consegue-se perceber que em 2019, aproximadamente 52% dos alunos, concentraram as faixas de notas [0;3[e [3;5[, enquanto as faixas [7;8,5[e [8,5;10[, apresentaram aproximadamente 25%, sendo o restante correspondente à faixa [5;7[. Já em 2020, houve uma inversão nos resultados, pois encontra-se nas faixas [0;3[e [3;5[, 26,15% dos alunos, enquanto nas faixas [7;8,5[e [8,5;10[, encontram-se 33,07%, além disso, observa-se a maior parte dos alunos na faixa [5;7[(30,78%).

Dessa maneira, consegue-se observar que as avaliações não realizadas no ambiente escolar interferiram nos valores, sendo um item muito importante a ser discutido. Além disso, em 2021, encontram-se os melhores resultados, pois apenas 19,82% dos estudantes estão nas faixas [0;3[e [3;5[, enquanto 60,17% estão nas faixas [7;8,5[e [8,5;10[e ainda se observa 19,82% na faixa [5;7[. No entanto, em 2021, mesmo os alunos realizando as avaliações em casa, os novos procedimentos metodológicos apresentaram um maior índice de aprendizagem quando comparados ao ano de 2020 (procedimentos metodológicos "tradicionais") (Figura 18-a).

Comparando as notas das avaliações 2, consegue-se perceber que em 2019 e 2020, respectivamente, 57,14 e 46,15% dos alunos, concentraram as faixas de notas [0;3[e [3;5[, enquanto as faixas [7;8,5[e [8,5;10[, apresentaram aproximadamente 26,81 e 30,77%, sendo que 19,05 e 23,08% correspondendo a faixa [5;7[. Já em 2021, encontram-se os melhores resultados, pois 29,83% dos estudantes estão nas faixas [0;3[e [3;5[, enquanto 40,35% estão nas faixas [7;8,5[e [8,5;10[e, ainda se observa 29,82% na faixa [5;7[. No entanto, nessa avaliação, consegue-se observar que só houve melhora nos aproveitamentos dos alunos quando os procedimentos metodológicos criados nesta pesquisa foram implementados (Figura 18-b).

Finalmente, analisando as notas das avaliações 3 nos respectivos anos, consegue-se observar uma variação parecida com a variação correspondente às avaliações 2, só que ainda melhor, pois apenas 19,30% estão na faixa [0;3[e [3;5[, enquanto 66,17% encontram-se nas faixas [7;8,5[e [8,5;10[e o restante dos alunos encontram-se na faixa 03 (14,03%). Nesse sentido, os dados mostram que com os procedimentos metodológicos propostos ocorreu uma diferença no processo de ensino-aprendizagem no decorrer do ano letivo, demonstrando que maiores números de avaliações durante o ano e um *feedback* mais específico, corroboram para os alunos obterem maior ritmo de estudos e foco e, consequentemente, um maior interesse, proporcionando assim um melhor rendimento (Figura 18-c).

Outro ponto que prova a eficácia desse procedimento metodológico é o trabalho final (Figura 18-d) que os alunos fizeram em dupla, o qual demonstra que somente no ano de 2021 ocorre uma melhora considerável quanto ao entendimento total da disciplina, pois obteve-se um nível de conhecimento de 31,58% na faixa [3[e de 42,11% nas faixas [7;8,5[e [8,5;10[. No intuito de comparar estatisticamente o desempenho dos alunos, atribuiu-se uma variação de 5% (para mais ou para menos), como um limite de igualdade e, dessa maneira, constata-se que houve variação significativa nas AVs 1 e 3 referentes aos três anos e, também nas comparações das AVs 2, nos anos de 2019 e 2021. Além disso, só se percebe uma variação estatística maior no trabalho final do ano de 2021, demonstrando que os alunos apresentaram melhor entendimento na disciplina apenas na metodologia proposta em 2021. Enquanto isso, os

resultados das AVs 2 e dos trabalhos finais, nos anos de 2019 e 2020, não apresentam variação significativa estatisticamente (Quadro 5).

Quadro 5 – Desempenhos equalizados das avaliações realizadas entre os anos de 2019 a 2021 e suas respectivas médias finais.

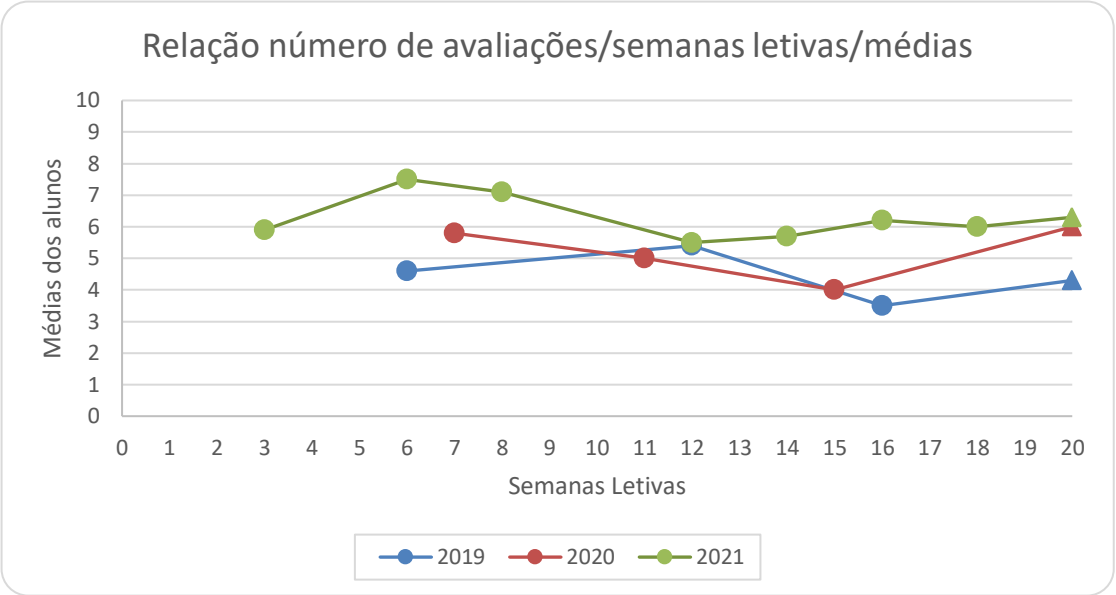
2019	2020	2021	2019	2020	2021
AV1	AV1	AD1 AD2 AD3	4,6	5,8	6,8
AV2	AV2	AD4 AD5 AD6	5,4	5	5,8
AV3	AV3	AD7	3,5	4	6
Trabalho	Trabalho	Trabalho	4,3	4,5	6,3

Fonte: Da Autora (2022).

Na AV1 e na AV3, quando comparamos os anos 2019 e 2020, percebe-se um aumento de 26,1% e 14,3%, respectivamente, na média dos alunos. Esse fato, possivelmente, é devido ao aluno realizar a avaliação podendo consultar qualquer material, além dos colegas, já que, no de 2020, o mundo estava em pandemia e as avaliações eram realizadas *online*. No entanto, quando se compara a AV2 e TF, nos mesmos anos, observa-se uma pequena alteração, dentro da variação estatística de 5% (para mais ou para menos), concluindo-se que não houve modificação. Quando se compara os resultados obtidos pela nova metodologia implementada em 2021, com os resultados obtidos em 2019 e 2020, todos apresentam variação significativamente maior. Na AV1, esse aumento foi de 47,8% (2019/2021) e 17,24% (2020/2021); na AV2, o aumento foi de 16% (2020/2021); enquanto na AV3, os aumentos constatados foram de 71,4% (2019/2021) e 50% (2020/2021) e no TF, os aumentos analisados foram de 46,5% (2019/2021) e 40% (2020/2021). Além disso, a única exceção ocorre na comparação da AV2 entre 2019/2021, onde a pequena variação (7,40%) sugere uma estabilidade de rendimento dos alunos. Dessa maneira, certifica-se que independentemente das avaliações serem online ou presencial, a metodologia comprovou ser eficaz na sua forma de utilização.

Um outro ponto relevante a ser analisado é o intervalo em que as avaliações são desenvolvidas durante o ano letivo e seu impacto nas médias dos alunos. A Figura 19, ilustra essa relação entre os anos de 2019 e 2021.

Figura 19 – Semanas letivas em que as avaliações são aplicadas e médias alcançadas nas avaliações e no trabalho final.

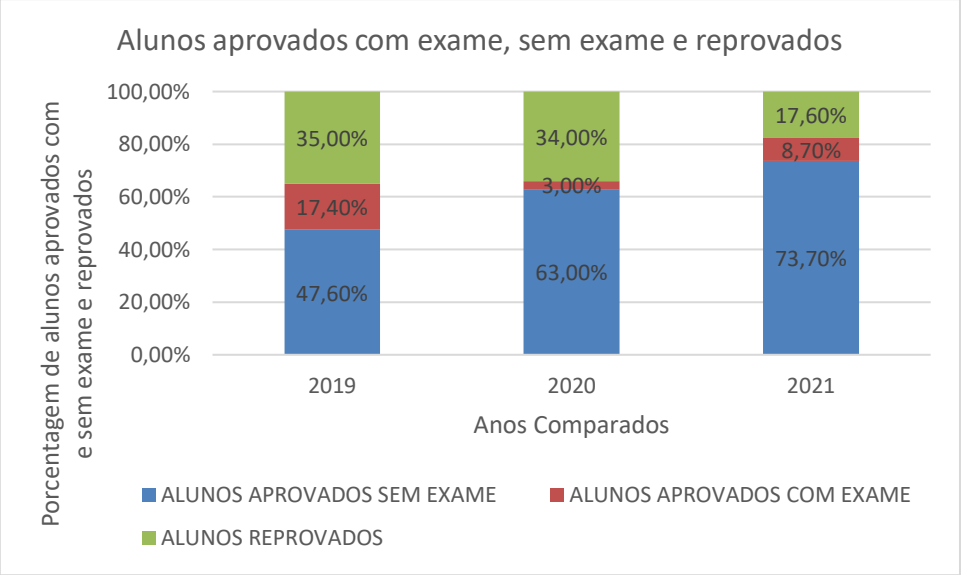


Fonte: Da Autora (2022).

Ao observar a Figura 19, percebe-se que, no ano de 2019, foram realizadas quatro avaliações, identificadas por meio de pequenos círculos no gráfico, dentro de um intervalo médio de cinco semanas entre a ocorrência de uma avaliação e outra. A média das notas dos alunos ficou em 4,45. Em 2020, mesmo com as aulas sendo ministradas de forma remota, o número de avalições e o intervalo de aplicação das avaliações se mantiveram tendo uma leve melhora nas médias dos alunos, as quais ficaram em 5,2. Já em 2021, ainda com as aulas sendo ministradas remotamente, mas com a nova proposta de ensino-aprendizagem sendo implementada, o número de avalições a serem realizadas aumentou e, com isso, a frequência com que eram desenvolvidas também. A média entre os intervalos das avaliações ficou em 2,5 semanas e as médias das notas dos alunos ficou em 6,2. Do mesmo modo, nos trabalhos finais identificados no gráfico por meio de triângulos, observou-se um acréscimo nas médias finais dos alunos, demostrando uma melhora em seus resultados na comparação de 2021 e 2019 e em comparação a 2021 e 2020. Também é importante ressaltar que além de melhorar a média anual, o intervalo das notas dos alunos se manteve em uma escala maior, não tendo quedas acentuadas conforme observa-se em 2019 e 2020.

A análise realizada reflete no índice de alunos aprovados sem exame, com exame e reprovados nesse mesmo período, conforme a Figura 20.

Figura 20 – Percentual de alunos aprovados com exame, sem exame e reprovados no período de 2019 a 2021.



Fonte: Da Autora (2022).

Na análise realizada, foi possível verificar o número de alunos que não atingiram a média (7,0) durante os anos letivos e tiveram que fazer o exame de recuperação. Percebe-se que, em 2019, quando as aulas foram realizadas presencialmente, os alunos tiveram um percentual acima dos anos seguintes, isso pode ser explicado pelo fato do aluno não ter a possibilidade de consultar nenhum material no momento da avaliação. No entanto, se compararmos os anos de 2020 e 2021, momento em que as aulas foram ministradas remotamente, observa-se que, em 2021, ocorre um decréscimo de 11 pontos percentuais no número de alunos que ficaram em exame de 37% para 26,3%. Essa melhora é considerada satisfatória, mostrando que a implementação da metodologia desenvolvida auxiliou os alunos não somente a não serem reprovados, mas a conseguirem ser aprovados sem precisar do exame final. Esses resultados confirmam que a metodologia proposta teve êxito, tanto na aceitação, quanto no desempenho dos alunos, pois o número daqueles que não atingiram a média para ser aprovado caiu de 52,4%, em 2019, para 26,3%, em 2021, mostrando uma redução de 50%.

Também foi realizada uma comparação nos índices de alunos aprovados sem exame e alunos aprovados com exame (Figura 20). Ao analisar os anos de 2019 e 2020, mesmo com contextos diferenciados, aula presencial e remota, os dados referentes ao índice de alunos aprovados, 65 e 66%, e alunos reprovados, 35 e 35%, praticamente não se alteraram. Já em 2021 ocorre uma melhora significativa no percentual de aprovação, sendo possível observar que os resultados alcançados com o desenvolvimento das ADs fizeram com que os alunos

tivessem uma evolução em seu aprendizado, em que o ritmo de estudo, por meio das avaliações contínuas e o *feedback* qualificado, proporcionou aos discentes construir o conhecimento de forma sólida, minimizando as lacunas em seu aprendizado. Além disso, a taxa de alunos aprovados subiu de 66 para 82,40%, mostrando que o aumento observado foi devido a metodologia implementada, pois em 2020, a taxa de alunos aprovados (66%) foi praticamente a mesmo de 2019.

6. CONCLUSÃO

Esta pesquisa nasceu da dificuldade encontrada no processo de ensino-aprendizagem da disciplina de Programação I do curso técnico em informática integrado ao ensino médio de uma Instituição Federal. Alunos ingressantes na instituição oriundos do ensino fundamental demostram dificuldade em compreender o conteúdo, pois a área da Computação exige que os alunos dominem habilidades importantes, como a interpretação, a abstração, o raciocínio lógico e noções matemáticas. Ao enfrentar dificuldades em dominar essas habilidades, os alunos ingressantes no curso técnico em informática passam a ter muitas dificuldades no processo de ensino-aprendizagem, apresentando um baixo rendimento na disciplina e acarretando altas taxas de reprovação.

Perante esse cenário, surge o problema de pesquisa: *Como melhorar o desempenho e diminuir o índice de reprovação dos alunos na disciplina Programação I em um curso técnico integrado ao ensino médio?* Diante do problema exposto, AD's foram construídas, introduzindo comandos básicos de programação por meio de fluxogramas e a linguagem C, com base nas habilidades estabelecidas por meio de tarefas que demonstram as habilidades dos alunos e propiciam ao professor disponibilizar um *feedback* que ajude os estudantes a entenderem quais as lacunas em sua formação necessitam ser melhoradas.

Para elaborar as ADs, foram considerados os estudos de Ausubel referente à Teoria da Aprendizagem, pois, na disciplina de Programação I, os conteúdos a serem ministrados possuem um inter-relacionamento e a importância de aprender de forma satisfatória é um tópico é crucial para o bom desenvolvimento da disciplina. Assim, estipulou-se que sete ADs seriam elaboradas, uma para cada tópico a ser desenvolvido durante o ano letivo. Também foi importante pensar em como apresentar o conteúdo inicialmente aos alunos, que chegam ao curso técnico em informática sem o mínimo de conhecimento da disciplina. Desse modo, a fim de não impactar os discentes com algo tão desconhecido, definiu-se pelo uso de fluxogramas em um primeiro momento, para após iniciar a programação com o uso da Linguagem C.

A estratégia estabelecida apresentou um bom resultado perante as avaliações dos alunos, que por meio de um questionário inserido no final de cada AD, permitiu investigar o empenho, a aprendizagem, o foco nos estudos e o trabalho despendido. Os itens avaliados mostraram que os discentes ao desenvolverem as tarefas referente a cada AD, tiveram resultados positivos, melhorando seu aprendizado e mantendo o empenho e o foco nas tarefas a serem desenvolvidas. Com relação ao trabalho despendido, os alunos concluíram que o número de tarefas podia ser menor, por causa do tempo para desenvolver os trabalhos das outras disciplinas. Ao analisar

que essa percepção foi relatada em mais de uma AD, uma adequação foi sendo realizada nas atividades seguintes e, até o final da pesquisa, um equilíbrio foi alcançado.

Além disso, esta pesquisa se mostrou eficiente quando os alunos analisaram a viabilidade do conjunto de ADs por meio da aplicação do questionário geral. Nesse questionário, os alunos foram indagados em relação ao ensino-aprendizagem, ao ritmo e foco nos estudos, ao número de tarefas em cada AD e de ADs desenvolvidas, a participação dos alunos, ao índice de resolução das ADs, a retomada do conteúdo, ao *feedback* qualificado e a contribuição no ensino remoto. Os alunos trouxeram, em suas descrições, que a metodologia desenvolvida contribuiu para que adquirissem um ritmo de estudo, pois avaliar sua aprendizagem a cada conteúdo e realizar uma retomada nas lacunas encontradas foram pontos importantes que contribuíram para que suas dúvidas não acumulassem, não deixando que conteúdos já ministrados se tornasse obstáculo para o aprendizado de um próximo tópico. Também foi salientado que a metodologia proposta auxiliou no processo de ensino remoto, pois com o *feedback* qualificado e a retomada do conteúdo os possibilitou a ter um melhor rendimento, acompanhando melhor a disciplina. Ainda em relação ao ensino remoto os alunos relataram que a metodologia propiciou uma comunicação entre alunos e o professor, motivando-os a frequentar os encontros síncronos, tão importantes para o seu aprendizado.

Além de analisar a estratégia desenvolvida, também foi verificada a evolução dos dissentes perante o desenvolvimento das ADs propostas. Para realizar o acompanhamento dos alunos foi elaborada uma matriz de análise que por meio dos critérios satisfatório, parcialmente satisfatório e insatisfatório, permitiu ao professor identificar dentro das habilidades analisadas o desempenho dos alunos.

Ao analisar as habilidades investigadas, foi verificado que a interpretação, a abstração, o raciocínio lógico e as noções matemáticas possuíram comportamentos parecidos nos critérios analisados em várias ADs, indicando que são habilidades que se complementam e, suas análises em conjunto, são um importante indicativo para os alunos, pois percebeu-se que quando os discentes conseguem esclarecer suas dúvidas em relação a uma dessas habilidades, o desempenho em todas as outras crescem simultaneamente, refletindo em um melhor aproveitamento e em um aumento das notas, as quais tiveram uma melhora significativa durante todo o ano letivo.

A fim de verificar a eficácia da metodologia desenvolvida, foi realizada uma comparação com as metodologias utilizadas nos anos de 2019 e 2020, considerando o mesmo contexto de aplicação. Para realizar essa verificação, foi comparado o percentual de alunos aprovados sem e com exame e reprovados na disciplina, o número de avaliações desenvolvidas, a escala de

notas e as médias alcançadas em cada avaliação. Os resultados encontrados nessas análises mostraram que com a metodologia proposta, os discentes obtiveram melhores resultados em suas avaliações, concentrando um maior número de alunos com notas acima da média, melhorando, consequentemente, suas notas finais. Com médias iguais ou acima de 7, menos alunos ficaram em exame, tendo uma melhor taxa de aprovação na disciplina. Esses resultados, demostram que o número maior de avaliações comparadas com os anos anteriores e um *feedback* qualificado levou os alunos a obterem melhores resultados ao serem comparados com os anos de 2019 e 2020.

Em síntese, conclui-se que as ADs apresentadas neste trabalho se mostraram uma alternativa viável para o ensino introdutório de programação. Isso porque o acompanhamento contínuo por meio das sete ADs e do *feedback* qualificado fez com que os alunos adquirissem um melhor ritmo de estudo e, consequentemente, um bom nível de entendimento dos conteúdos trabalhados. Além disso, as ADs se mostraram importantes durante a pandemia de Covid-19, pois a baixa interação e comunicação com os alunos foram importantes fatores a serem superados. Futuramente, essa metodologia poderá ser aplicada em outras instituições de ensino, a fim de analisar sua viabilidade em contextos diferentes, onde os mesmos julgamentos possam ser realizados, além disso, pode ser testada em outras áreas do conhecimento, onde o ritmo de estudo e o *feedback* qualificado sejam fatores a serem considerados no desenvolvimento da disciplina.

REFERÊNCIAS

AGUILAR, Luis Joyanes. **Fundamentos de Programação: Algoritmos, estruturas de dados e objetos**. 3. ed. São Paulo: AMGH, 2011.

ALVES, Fabio; JAQUES, Patricia. **Um ambiente virtual com feedback personalizado para apoio a disciplina de programação**. Trabalho de Conclusão de Curso (Análise de desenvolvimento de Sistemas) – Universidade do Vale do Sinos – UNISINOS – São Leopoldo, 2014.

ALVES, Josemar. **Desenvolvimento de um Sistema Integrado para Implementação de Tarefas Avaliativas Reflexivas e Formativas Contínuas.** 2018. 42 p. Tese (Doutorado em Educação em Ciências: Química da Vida e Saúde) – Universidade Federal de Santa Maria, Santa Maria, RS, 2018. Disponível em: https://repositorio.ufsm.br/handle/1/14732. Acesso em: 20 janeiro 2020.

AMBRÓSIO, Ana Paula *et al*. Programação de Computadores: compreender as dificuldades de aprendizagem dos alunos. **Revista Galego-Portuguesa de Psicoloxía e Educación**, v. 19, n. 1, ano 16, p. 185–197, 2011. Disponível em: <http://repositorium.sdum.uminho.pt/handle/1822/15554>. Acesso em: 20 fevereiro 2021.

ARIMOTO, Maurício Massaru; OLIVEIRA, Weldrey Toneto. Dificuldades no Processo de Aprendizagem de Programação de Computadores: um Survey com Estudantes de Cursos da Área de Computação. *In*: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 27, 2019, Belém. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2019. p. 244-254. ISSN 2595-6175. DOI: https://doi.org/10.5753/wei.2019.6633.

AURELIANO, Viviane Cristina Oliveira; TEDESCO, Patricia Cabral Azevedo Restelli. Ensino-aprendizagem de programação para iniciantes: uma revisão sistemática da literatura focada no SBIE e WIE. *In*: **Anais[...].** SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, volume 23, 2012a. DOI: http://dx.doi.org/10.5753/cbie.sbie.2012.%25p.

AUSUBEL, David Paul. **Aquisição e Retenção de Conhecimentos: Uma perspectiva Cognitiva.** Coimbra: Paralelo, 2000.

BAKKER, Arthur**. Design Research in Education: A Practical Guide for Early Career Researchers.** Editora: Routledge, 2018.

BAEZA-YATES, Ricardo. **Teaching Algorithms. SIGACT News**, v. 26, 4. ed, p. 51-59, 1995. Disponível em: https://doi.org/10.1145/219817.219828.Acesso em: 18 março 2020.

BARBOSA, Leônidas da Silva. **Aprendizado Significativo Aplicado ao Ensino de Algoritmos**. 2011. 71 p. Dissertação (Mestrado em Sistemas e Computação) – Universidade Federal do Rio Grande do Norte, Natal, RN, 2011.

BELCHIOR, Jéssica Hannah *el al.* MEIRELES, M. A. C.; FERREIRA, R. DOS S. **Avaliando Aspectos Motivacionais do Uso da Ferramenta Scratch para Ensino de Programação: Um Relato de Experiência.** Nuevas Ideas en Informática Educativa, v. 12, p. 618–623, 2016.

BERSSANETTE, João Henrique. **Ensino de Programação de Computadores: Uma Proposta de Abordagem Prática Baseada em Ausubel**. 2016. 145 p. Dissertação (Mestrado Profissional em Ensino de Ciência e Tecnologia) – Universidade Tecnológica Federal do Paraná, Ponta Grossa, PR, 2016.

BRANCO, Neto Wilson Castello; SCHUVARTZ Aguinaldo Antônio. Ferramenta Computacional de Apoio ao Processo de Ensino-Aprendizagem dos Fundamentos de Programação de Computadores. *In*: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO (SBIE), 2007, v. 28, São Paulo, SP. **Anais[...]** São Paulo/SP: Mackenzie, 2007, p. 520-528.

BRASIL, Ministério da Educação. **Diretrizes Curriculares Nacionais para os Cursos de Graduação em Ciência da Computação**. Resolução CNE/CES 5/2016. Diário Oficial da União, Brasília, 17 de novembro de 2016, Seção 1, págs. 22-24. Dispnível em: http://portal.mec.gov.br/component/content/article?id=12991.

BRASIL. Ministério da Educação. **Base Nacional Comum Curricular.** Brasília: 2018. Disponível em: http://basenacionalcomum.mec.gov.br/images/BNCC_EI_EF_110518_versaofinal_site.pdf.

BREZOLIN, Warner. **Mapas Conceituais e os Conceitos da Disciplina de Algoritmos: Uma Aplicação para a Aprendizagem Significativa.** 2015. 117 p. Dissertação (Mestrado em Tecnologias da Inteligência e Design Digital) – Pontifícia Universidade Católica de São Paulo, São Paulo, SP, 2015. Disponível em: https://leto.pucsp.br/bitstream/handle/18196/1/Warner%20Brezolin.pdf Acesso em: 24 outubro 2019.

BROEKKAMP, Hein; VAN HOUT-WOLTERS, Bernadette. **The gap between educational research and practice: A literature review, symposium, and questionnaire**. Educational Research and Evaluation, v. 13, n. 3, p. 203-220, 2007. Disponível em: https://www.researchgate.net/publication/46697041_The_gap_between_educational_research_and_practice_A_literature_review_symposium_and_questionnaire Acesso em: 23 outubro 2021.

BROWN, A. L. Design experiments: Theoretical and methodological challenges in creating complex interventions in classroom settings. **Journal of the Learning Sciences**, v. 2, n. 2 (1992), pp. 141–178, 1992. Dispnível em: https://www.jstor.org/stable/1466837?origin=JSTOR-pdf. Acesso em: 21 novembro 2020.

CARREÑO-LEON, M *et al*. Gamification technique for teaching programming. 2018 IEEE Global Engineering Education Conference (EDUCON). **Anais[...].** *In*: 2018 IEEE GLOBAL ENGINEERING EDUCATION CONFERENCE (EDUCON). Tenerife: IEEE, 2018.

CASAROTTO, Romeu Isaac *et al*. Logirunner: um Jogo de Tabuleiro como Ferramenta para o Auxílio do Ensino e Aprendizagem de Algoritmos e Lógica de Programação. **Revista: Renote**, v. 16, n. 1, 21 ago. 2018. Disponível em: https://seer.ufrgs.br/renote/article/view/85998: Acesso em: 24 novembro 2021.

CAZZOLA, Walter; OLIVARES, Diego Mathias. **Gradually Learning Programming Supported by a Growable Programming Language.** IEEE Transactions on Emerging Topics in Computing, v. 4, n. 3, p. 404–415, jul. 2016. Disponível em:

https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7273711. Acesso em: 12 janeiro 2022.

COLL, César *et al*. **O construtivismo na sala de aula.** 6. ed. São Paulo: Ática, 1999.

COLLINS, Allan. Towards a design Science of education**.** In: SCANLON, E; O'SHEA, T. (Orgs.). **New Directions in educational technology**. Berlin: Springer, 1992. p. 15-22. Disponível em: http://dx.doi.org/10.1007/978-3-642-77750-9_2. Acesso em: 4 maio 2020.

COSTA, Eli Banks Liberato. **O Invento de Jacquard e os Computadores: alguns aspectos das origens da programação no século XIX.** 2008. 94 p. Dissertação (Mestrado em História da Ciência), Pontifícia Universidade Católica de São Paulo, São Paulo, SP, 2008. Disponível em: https://sapientia.pucsp.br/bitstream/handle/13377/1/Eli%20Banks%20Liberato%20da%20Costa.pdf. Acesso em: 10 fevevereiro 2021.

CREWS, Thad; ZIEGLER, Uta. **The Flowchart Interpreter for Introductory Programming Courses**. Department of Computer Science Western Kentuchy. University Bowling Green, 1998. Disponível em: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=736854. Acesso em: 20 janeiro 2021.

DANTAS, Marcos. **O Crime de Prometeu: como o Brasil Obteve a Tecnologia da Informática.** Rio de Janeiro: ABICOMP – Associação Brasileira da Indústria de Computadores e Periféricos. Setembro, 1989. Disponível em: http://marcosdantas.com.br/conteudos/wp-content/uploads/2018/04/7-LIVRO_O_Crime_de_Prometeu.pdf. Acesso em: 13 março 2021.

DA SILVA, Marcos Antônio Vieira. **MODI – A Proposal of a visual tool to simulate and synthesize software applied to embedded systems**. 2002. 140 f. Tese (Doutorado em Engenharia Elétrica) – Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas, Campinas, SP, 2001.

DBRC (DESIGN-BASED RESEARCH COLLECTIVE). Design-Based Research: an emerging paradigm for educational inquiry. **Educational Researcher**, v. 32, n. 1, p. 5-8, 2003. Disponível em: https://journals.sagepub.com/doi/10.3102/0013189X032001005. Acesso em: 20 maio de 2020.

DE OLIVEIRA, Márcia; NOGUEIRA, Matheus; OLIVEIRA, Elias. Sistema de Apoio à Prática Assistida de Programação por Execução em Massa e Análise de Programas. *In*: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 23, 2015, Recife. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2015. p. 90-99. ISSN 2595-6175. DOI: https://doi.org/10.5753/wei.2015.10225.

DE VRIES, B.; PIETERS, J. M. Bridging the gap between research and practice: Exploring the role of knowledge communities in educational change**.** *In***: European educational research journal.** 2007 ; Vol. 6, No. 4. pp. 382-392. Disponível em: https://journals.sagepub.com/doi/10.1177/0002716216633404. Acesso em: 21setembro 2021.

DIEMER, Mouric Halen *et al.* Metodologias Ativas no Ensino de Algoritmos e Programação: Um relato de Aplicação da Metodologia Peer Instruction. **Revista Destaques Acadêmicos,** Lajeado, v. 11, n. 4, 2019. ISSN 2176-3070. Disponível em: http://univates.br/revistas/index.php/destaques/article/view/2400. Acesso em: 28 setembro 2020.

EULER, Dieter. Design-Research – a paradigm under development. In: EULER, D; SLOANE, P.F.E. (Orgs.) **Design-Based Research**. Stuttgart: Franz Steiner Verlag, 2014. p. 15-44. Disponível em: https://www.alexandria.unisg.ch/publications/232672. Acceso em: 4 abril 2020.

FAÊDA, Leonardo; BAFFA, Matheus; PEREIRA, Julie. AI(3P)A: Uma Metodologia para o Ensino de Lógica de Programação Utilizando Jogos Eletrônicos. SBC – Proceedings of SBGames 2020 — ISSN: 2179-2259. Disopnível em: https://www.sbgames.org/proceedings2020/EducacaoFull/209584.pdf. Acesso em: 14 março 2021.

FALKEMBACH, Gilse Antoninha *et al*. Aprendizagem de Algoritmos: Uso da Estratégia Ascendente de Resolução de Problemas. *In*: 8º TALLER INTERNACIONAL DE SOFTWARE EDUCATIVO. **Anais[...].** Santiago, Chile, 2003.

FRAEFEL, Urban. **Professionalization of pre-service teachers through univer-sity-school partnerships Partner schools for Professional Development: development, implementation and evaluation of cooperative learning in schools and classes.** WERA Focal Meeting: Edinburgh, 2014. Disponível em: https://www.researchgate.net/publication/275040746. Acesso em: janeiro de 2021.

GAUDENCIO, Matheus *et al.* Eu sei o que Vocês Fizeram (Agora e) na Aula Passada: o TSTView no Acompanhamento de Exercícios de Programação. *In*: II CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – CBIE 2013. XXIV SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE 2013. **Anais[...]**. Porto Alegre – RS, Brasil.

GOMES, Mouglas; D'EMERY, Richarlyson; CYSNEIROS, Gilberto. AAPW: Uma ferramenta para facilitar o aprendizado de programação Web. *In*: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 2014, Brasília. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2014. p. 269-278. ISSN 2595-6175.

GOMES, Tancicleide; MELO, Jeane Cecília Bezerra. App Inventor for Android: Uma Nova Possibilidade para o Ensino de Lógica de Programação.*In:* **Anais[...].** II CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – CBIE; WORKSHOPS (WCBIE) 2013, v. 2, n. 1. DOI:10.5753/CBIE.WCBIE.2013.620.

GONZALEZ, Sahudy Montenegro; TAMARIZ, Annabell Del Real. Integração de uma Metodologia de Ensino Presencial de Programação com um Sistema Tutor Inteligente. **Revista Brasileira de Informática na Educação (RBIE).** v. 22, n. 02 2014. Disponível em: https://www.researchgate.net/publication/271371128_Integracao_de_uma_Metodologia_de_E

nsino_Presencial_de_Programacao_com_um_Sistema_Tutor_Inteligente_Integration_of_a_Methodology_for_Teaching_Programming_with_an_Intelligent_Tutoring_System. Acesso em: 20 fevereiro 2021.

GOODRICH, Michael; TAMASSIA, Roberto. **Projeto de Algoritmos: fundamentos, análise e exemplos da Internet.** Porto Alegre: Bookman, 2008.

GOSSMANN, Tiago. **A utilização de algoritmos nos processos de ensino e de aprendizagem de programação de computadores em cursos técnicos.** 2017. Especialização – Curso de Docência na Educação Profissional, Universidade do Vale do Taquari - Univates, Lajeado, 2017. Disponível em: http://hdl.handle.net/10737/2047. Acesso em: 13 junho de 2021.

GUEDES, Luciana Rita; PATERNO, Aleksander Sade. BrC: Proposta de uma Biblioteca em Português para Ensino de Programação em Linguagem C. *In*: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 26, 2018, Natal. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2018. ISSN 2595-6175.
DOI: https://doi.org/10.5753/wei.2018.3523.

HALLEY, Wesley; GONDIM, Ana Paula Ambrósio. Esboço de Fluxogramas no Ensino de Algoritmos. *In:* **Anais[...].** XXVIII CONGRESSO DA SBC; WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO. Belém/PA, 2008.

HEMING, Cléverton **Ferramenta de apoio ao ensino e aprendizagem de algoritmos e programação.** 82p. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) – Universidade do Vale do Taquari – UNIVATES - Lajeado 2018. Disponível em: http://hdl.handle.net/10737/2228. Acesso em: 12 novembro 2021.

HENDERSON, Peter. Anatomy of an Introductory Computer Science Course. In: **Proceedings of the Seventeenth SIGCSE Technical Symposium on Computer Science Education**, ACM Press, 1986. p. 257-264. Disopnível em: https://www.researchgate.net/publication/221537763_Anatomy_of_An_Introductory_Computer_Science_Course. Acesso em: 30 março 2021.

HOADLEY, Chistopher. Creating context: Design-based research in creating and understanding CSCL. **Computer Science Bibliography**, 2002. Disponível em: https://www.researchgate.net/publication/2495924_Creating_context_Design-based_research_in_creating_and_understanding_CSCL. Acesso em: 24 maio 2021.

JUNIOR, Amaury Antônio Castro *et al*; Uma Análise Preliminar da Aplicação do Método 300 em Turmas de Algoritmos e Programação. *In:* **Anais[...].** WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 2021.

JUNIOR, José Carlos Rrocha Pereira *et al.* Ensino de Algoritmos e Programação: Uma experiência no Nível Médio. *In*: **Anais [...].** CONFERENCE: XXV CONGRESSO DA SBC – X WEI (WORKSHOP DE EDUCAÇÃO EM INFORMÁTICA). São Leopoldo – RS, 2005.

KOLIVER, Cristian; DORNELES, Ricardo Vargas; CASA, Marcos Eduardo. Das (muitas) dúvidas e (poucas) certezas do ensino de algoritmos. *In:* **Anais**[...]. XII WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO – WEI, 24. Salvador, BA, 2004.

LIMA, Árllon; DINIZ, Marcos; ELIASQUEVICI, Marianne. Metodologia 7Cs: Uma Nova Proposta de Aprendizagem para a Disciplina Algoritmos. *In:* WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 27, 2019, Belém. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2019. p. 429-443.

MATTA, Alfredo Eurico Rodrigues; SILVA, Francisca de Paula Santos; BOAVENTURA, Edivaldo Machado. *Design-based research* ou Pesquisa de Desenvolvimento: metodologia para pesquisa aplicada de inovação e educação do século XXI. **Revista da FAEEBA – Educação e Contemporaneidade**, Salvador, v. 23, n. 42, p. 23-36, 2014. Disponível em: https://www.researchgate.net/publication/327169964_DESIGN-BASED_RESEARCHOU_PESQUISA_DE_DESENVOLVIMENTOMETODOLOGIA_PARA_PESQUISA_METODOLOGIA_PARA_PESQUISA_DE_DESENVOLVIMENTO_METODOLOGIA_PARA_PESQUISAAPLICADA_DE_INOVACAO_EM_EDUCACAO_DO_SECULO_XXI. Acesso em: 4 abril 2021.

MCKENNEY, Susan; REEVES, Thomas. ***Conducting Educational Desing Research***. 1. ed. Routledge: Londres e Nova York, 2012.

MCKENNEY, Susan; REEVES, Thomas. ***Conducting Educational Desing Research***. 2. ed. Routledge: Londres e Nova York, 2019.

MEDINA, Marcos; FERTIG, Cristina. Algoritmos e Programação – Teoria e Prática. Editora: Novatec, 2005.

MENZIES, Tim. Applications of Abduction: Knowledge-Level Modeling. **International Journal of Human Computer Studies**. v. 45, issue 3, páginas 305-335, 1996. Disponível em http://menzies.us/pdf/96abkl.pdf. Acesso em: 21 março 2019.

MICHEL, Maria Helena. Metodologia e Pesquisa Científica em Ciências Sociais. Editora: Atlas, 2008.

MIRANDA, Hamilton Jesus. **Estratégias de Leitura como Instrumento na Formação do Leitor Competente**. 2016. 206 p. Dissertação (Mestrado Profissional em Letras em Rede Nacional) – Universidade Federal do Pará, Belém/PA, 2016. Disponível em: http://repositorio.ufpa.br/jspui/handle/2011/8227. Acesso em: 18 outubro 2021.

MORAIS, Ceres Germanna Braga; NETO, Francisco Milton Mendes; OSÓRIO, Antônio José Meneses. Dificuldades e desafios do processo de aprendizagem de algoritmos e programação no ensino superior: Uma revisão sistemática de literatura. ***Research, Society and Development***, *9*(10), 2020. Disponível em: https://www.researchgate.net/publication/344943457_Dificuldades_e_desafios_do_processo_de_aprendizagem_de_algoritmos_e_programacao_no_ensino_superior_uma_revisao_sistematica_de_literatura. Acesso em: 25 abril 2021.

MOREIRA, Marco Antonio; MASINI, Elcie Salzano. **Aprendizagem significativa:** a teoria de David Ausubel. São Paulo: Editora Centauro, 2011.

MOREIRA, Marco Antônio. Aula Inaugural do Programa de Pós-Graduação em Ensino de Ciências Naturais, Instituto de Física, Universidade Federal do Mato Grosso, Cuiabá, MT, 23

de abril de 2010. Aceito para publicação, Qurriculum, La Laguna, Espanha, 2012. Disponível em: file:///C:/Users/llour/Downloads/coloquio2%20(2).pdf. Acesso em: 10 janeiro 2022.

MOREIRA, Marco Antônio. **Aprendizagem Significativa: a teoria e textos complementares.** 1. ed. São Paulo: Livraria Física, 2011.

MOREIRA, Marco Antônio. **Teorias de Aprendizagem**. 2. ed. ampl. – [Reimp]. – São Paulo: E.P.U., 2019.

MOURA, Manoel Oriosvaldo. **Educação Matemática na infância:** abordagens e desafios. Editora: Vila Nova de Gaia, USP, 2007. Disponível em: https://repositorio.usp.br/item/001608539. Acesso em: 21 abril 2020.

NETTO, Dorgival *et al*. Game Logic: Um jogo para auxiliar na aprendizagem de lógica de programação. *In*: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 25, 2017, São Paulo. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2017. ISSN 2595-6175. DOI: https://doi.org/10.5753/wei.2017.3546.

NOBRE, Isaura Aalcima; MENEZES, Crediné Silva. Suporte à Cooperação em um Ambiente de Aprendizagem para Programação (SAmbA). *In:* **Anais[...]**. XIII SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO – SBIE 2002. São Leopoldo, RS. São Leopoldo, RS, Brasil, 2002.

PAES, Rrodrigo Barros *et al*; Ferramenta para avaliação de aprendizado de alunos em programação de computadores. *In:* **Anais [...].** II CONGRESSO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, WORKSHOP, 2013.

PANSANATO, Luciano; DA SILVA, Adriane. Avaliação em Duas Fases na Disciplina Algoritmos. *In*: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 28, 2020, Cuiabá. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2020. p. 76-80. ISSN 2595-6175. DOI: https://doi.org/10.5753/wei.2020.11133.

PEREIRA JÚNIOR, José Carlos Rocha; RAPKIEWICZ, Cleci Elena. O Processo de Ensino-Aprendizagem de Programação: Uma Visão Crítica da Pesquisa no Brasil. *In:* **Anais[...].** WORKSHOPE DE INFORMÁTICA NA EDUCAÇÃO - WEI. Salvador- BA, 2004.

PEREIRA, João Pedro Lima. **Programação e Pensamento Computacional no 8º e 9º ano do Ensino Fundamental: Um Estudo de Caso**. Dissertação (Mestrado profissional em Matemática em Rede Nacional) – Universidade de Brasília, DF, 2019. Disponível em: https://repositorio.unb.br/bitstream/10482/37528/1/2019_Jo%C3%A3oPedrodeLimaPereira.pdf. Acesso em: 8 janeiro 2022.

PIRES, Fernanda *et al*. Desenvolvendo o Pensamento Computacional através da Máquina de Turing: o enigma do Curupira. *In*: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 27, 2019, Belém. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2019. p. 523-532. ISSN 2595-6175. DOI: https://doi.org/10.5753/wei.2019.6657.

RABÊLO, Júnior Dilson José Lins *et al*; Cosmo: Um ambiente virtual de aprendizado com foco no Ensino de Algoritmos. *In*: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 2018, Natal. **Anais [...]**. Porto Alegre: Sociedade Brasileira de

Computação, 2018. ISSN 2595-6175. DOI: https://doi.org/10.5753/wei.2018.3524. Disponível em: 04 fevereiro 2022.

RAPKIEWICZ, Clevi Elena *et al*. Estratégias Pedagógicas no Ensino de Algoritmos e Programação Associadas ao Uso de Jogos Educacionais. **Revista Renote:** Novas Tecnologias na Educação, Porto Alegre, v. 4, n. 2, p.1-6, dez. 2006. Disponível em:<http://www.seer.ufrgs.br/index.php/renote/article/view/14284/8203>. Acesso em: 17 março de 2019.

ROBERTO, Giancarlo *et al.* TuPy Online - Programação em Português com Visualização de Execução e Abstrações de Estruturas de Dados na Web. *In*: WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO (WEI), 26. 2018, Natal. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2018. ISSN 2595-6175. DOI: https://doi.org/10.5753/wei.2018.3498.

RODRIGUES, Miranda. Experiências positivas para o ensino de algoritmos. *In*: IV ESCOLA REGIONAL DE COMPUTAÇÃO. Bahia-Sergipe. Feira de Santana/BA. **Anais[...].** Feira de Santana/BA. 2004.

RODRIGUES, Methanias Colaço Junior. Como Ensinar Programação? **Jornal Computação Brasil da Sociedade Brasileira de Computação**, 2002. Disponível em: http://www.unit.br/methanias/artigos.htm. Acesso em: 20 abril 2021.

ROGERS, Carl. **Tornar-se pessoa.** São Paulo: Martins Fontes, 8. ed. 1998.

SCANLAN, David. **Structured Flowcharts Outperform Pseudocode**: An Experimental Comparisons. California States University at Sacramento, 1989. Disponível em: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=35587. Acesso em: 30 março 2020.

SCHULTZ, Max Rubem Oliveira. **Metodologias para Ensino de Lógica de Programação de Computadores**. 2003. Monografia (Especialização) – Universidade Federal de Santa Catarina, Florianópolis, SC, Brasil, 2003.Disponível em: http://repositorio.ufsc.br/xmlui/handle/123456789/89982. Acesso em: 18 julho 2021.

SEBESTA, Robert. **Conceitos de Linguagens de Programação.** 11. ed. ed. Bookman, 2018.

SILVA, Dirceu Jesus Lima da. Desenvolvimento de programa computacional para determinação de valores de concentração de tensões. 2016. 171 p. Dissertação (Mestrado em Projeto e Processos de Fabricação) - Universidade de Passo Fundo, Passo Fundo, RS, 2016. Disponível em: http://tede.upf.br/jspui/bitstream/tede/1626/2/2016DirceuJesusLimaSilva.pdf. Acesso em: 23 março 2022.

SILVA, Thiago Reis da Silva *el al*. **Ensino-aprendizagem de programação: uma revisão sistemática da literatura. Revista** Brasileira de Informática na Educação, Porto Alegre, v. 23, n 1, 2015. Disponível em: http://ojs.sector3.com.br/index.php/rbie/article/view/2838

SLACK, Nigel *et al*. **Administração da Produção.** São Paulo: Editora Atlas, 1. ed. 1997.

SOUZA, Eduardo Rramos Coimbra. **Schopenhauer e os conhecimentos intuitivo e abstrato: uma teoria sobre as representações empíricas e abstratas**. São Paulo: Editora UNESP; São Paulo: Cultura Acadêmica, 2015, pp. 97-146.

TAVARES, Romero. Construindo mapas conceituais. **Ciências & Cognição**, v. 12, p. 72-85, 2007. Disponível em: http://www.cienciasecognicao.org/pdf/v12/m347187.pdf. Acesso em: 1 março de 2018.

TERRY, Anderson, SHATTUCK, Julie. Design-Based Research: A Decade of Progress in Education Research? **Educational Researcher**, v. 41, n. 1, p. 16-25, 2012. Disponível em: https://www.researchgate.net/publication/254088681_Design-Based_Research. Acesso em: 20 fevereiro 2021.

VAN DEN AKKER, Jan. Principles and methods of development research. In: VAN DEN AKKER, J. *et al* (Ed.). **Design methodology and developmental research in education and training**. Norwell: Kluwer Academic Publishers,1999. p. 1-14.

VAN DEN AKKER, Jan *et al.* **Educational Design Research**. Editora: Routled. 1. ed, p 176, 2006.

VILLAÇA, Marco Valério Miorim; STEINBACH, Reginaldo. Brevíssima História do Computador e suas tecnologias – parte I – Do Osso de Lebombo aos computadores eletromecânicos. **Revista**: Ilha Digital, ISSN 2177-2649, volume 5, páginas 3 – 24, 2014. Disopnível em: https://ilhadigital.florianopolis.ifsc.edu.br/index.php/ilhadigital/article/view/72. Acesso em: 2 março 2022.

ZACARIAS, Rodrigo Oliveira; MELLO, Denise Ribeiro Barreto. Metodologias de Ensino de Lógica de Programação e Algoritmos em Cursos de Graduação. **Revista Interdisciplinar do Pensamento Científico**, 5(2), 29-44, 2019. Disponível em: https://www.researchgate.net/publication/339313254_METODOLOGIAS_DE_ENSINO_DE_LOGICA_DE_PROGRAMACAO_E_ALGORITMOS_EM_CURSOS_DE_GRADUACAO. Acesso em: 12 junho 2021.

APÊNDICE A – QUESTIONÁRIO GERAL DE AVALIAÇÃO DAS ATIVIDADES DIDÁTICAS

Responda o questionário abaixo afim de avaliar a metodologia utilizada no ano de 2021 no desenvolvimento da disciplina de Programação I.

Questão 1

As atividades didáticas foram importantes para manter o ritmo e o foco no estudo da disciplina? Justifique sua resposta.

Questão 2

Como ficou seu desempenho com o desenvolvimento das atividades didáticas?

Questão 3

Em relação as dúvidas encontradas na disciplina, as atividades didáticas ajudaram a esclarece-las? Justifique sua resposta.

Questão 4

O número de tarefas dadas em cada atividade didática foi satisfatório a fim de esclarecer suas dúvidas? Justifique sua resposta.

Questão 5

Em relação ao número de atividades didáticas dadas durante o ano letivo para dar ritmo ao estudo e auxiliar nas dificuldades encontradas no decorrer da disciplina, foram importantes? Justifique sua resposta.

Questão 6

Em relação ao retorno qualificado (correções das atividades didáticas) dado ao seu estudo e a revisão desenvolvida em cima das dificuldades apresentadas, as atividades didáticas auxiliaram em seu processo de aprendizagem? Justifique sua resposta.

Questão 7

Você considera que as atividades didáticas foram importantes para melhorar sua destreza (programar de maneira mais rápida) em programação?

Questão 8

Com relação a metodologia adotada durante o ano letivo de 2021, a mesma contribuiu para o ensino de programação? Justifique sua resposta.

Questão 9

As atividades didáticas trouxeram benefícios no desenvolvimento da disciplina durante o ensino remoto? Justifique sua resposta.

APÊNDICE B – ATIVIDADES DIDÁTICAS CONCEITOS INICIAIS DE PROGRAMAÇÃO - FLUXOGRAMAS

TURMA INFO 1A

Nome:_____

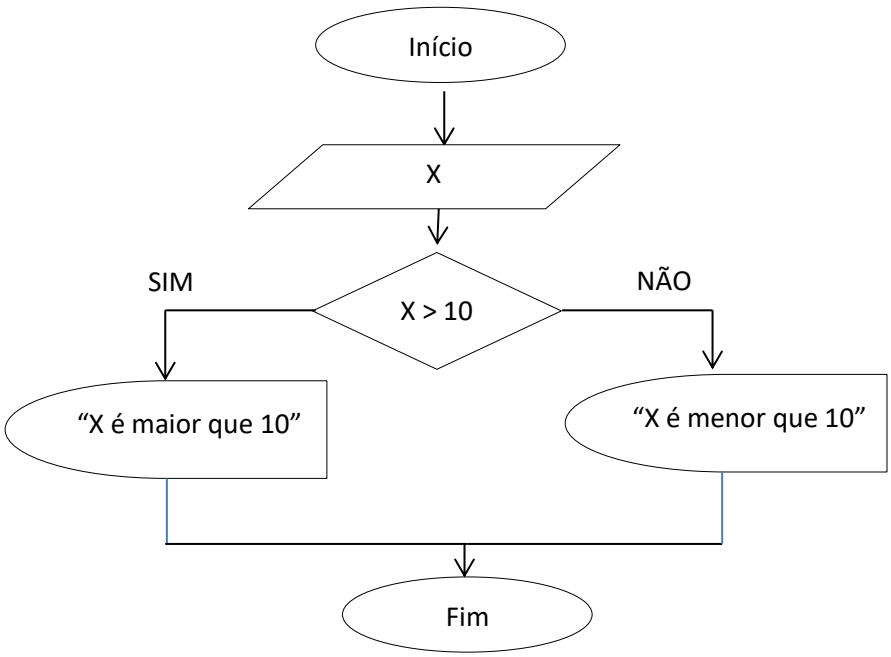
Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

1.Diga que valores serão impressos, após a execução do algoritmo, conforme as entradas dadas abaixo. Nas letras a, b, c e d ao lado de cada símbolo identifique-os com o seu nome.

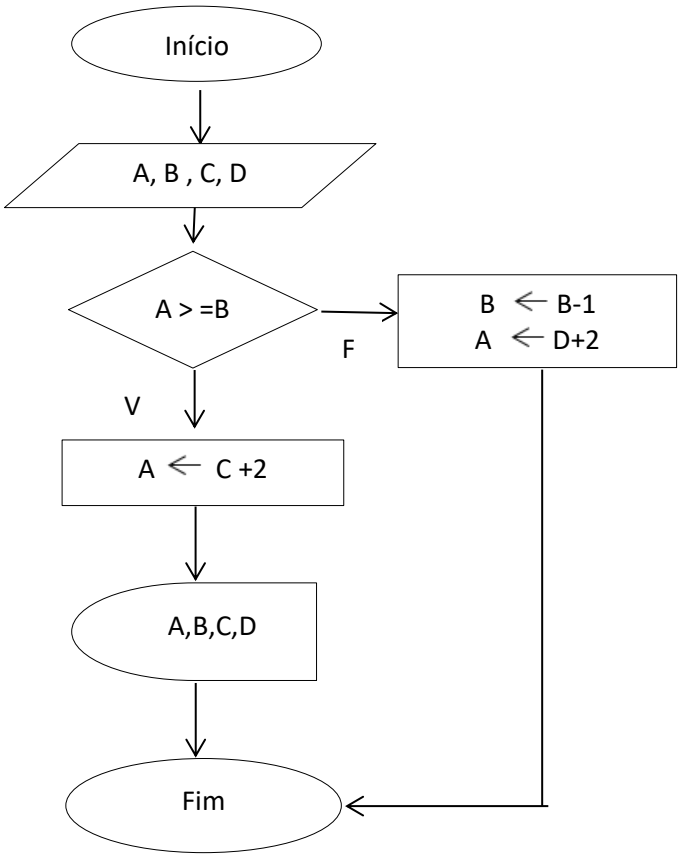
a)



Conforme as entradas abaixo escrevam ao lado a saída correspondente.

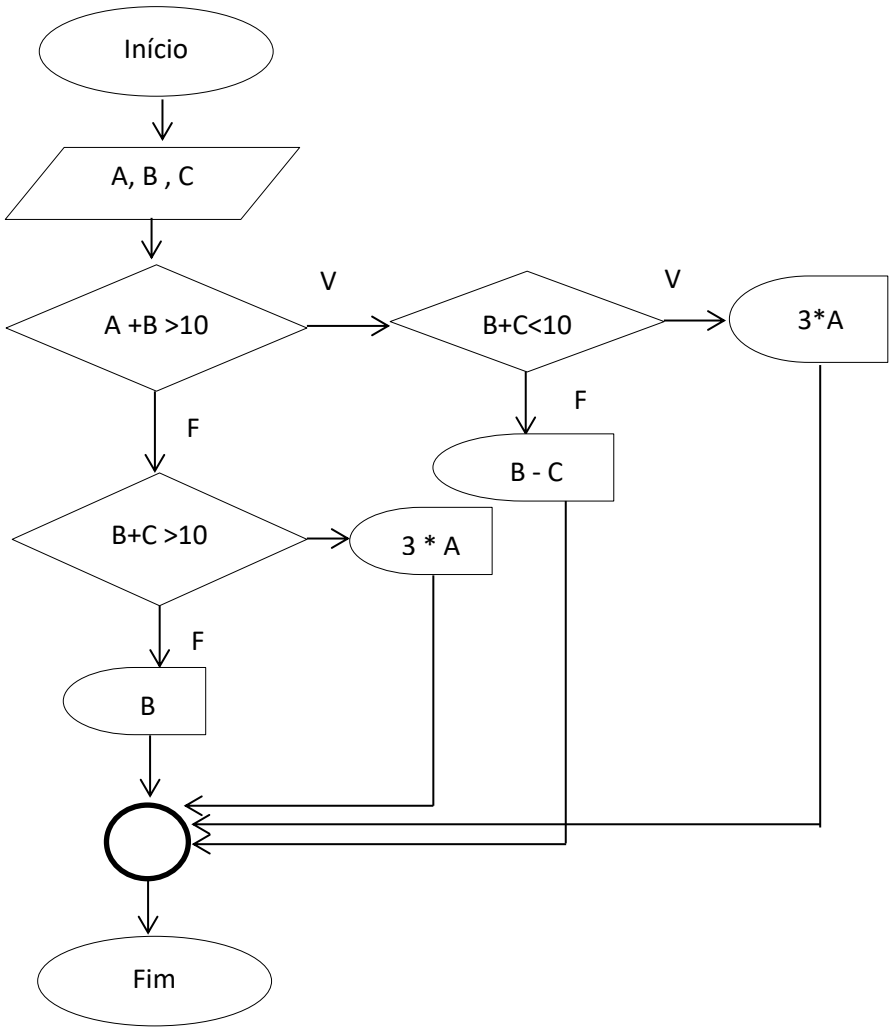
a) -8 - Saída: _____
b) 15 - Saída: _____
c) A - Saída: _____
d) 4 e 20 - Saída: _____
e) 10 - Saída: _____

b)



a) A=51, B=10, C=68 e D=22. Saída_____
b) A=-30, B=22, C=-2 e D=10. Saída_____
c) A=0, B=0, C=9 e D=60. Saída_____
d) A=20, B=35, C=-10 e D=0. Saída_____
e) A=-7, B=-3, C=0 e D=-20. Saída_____

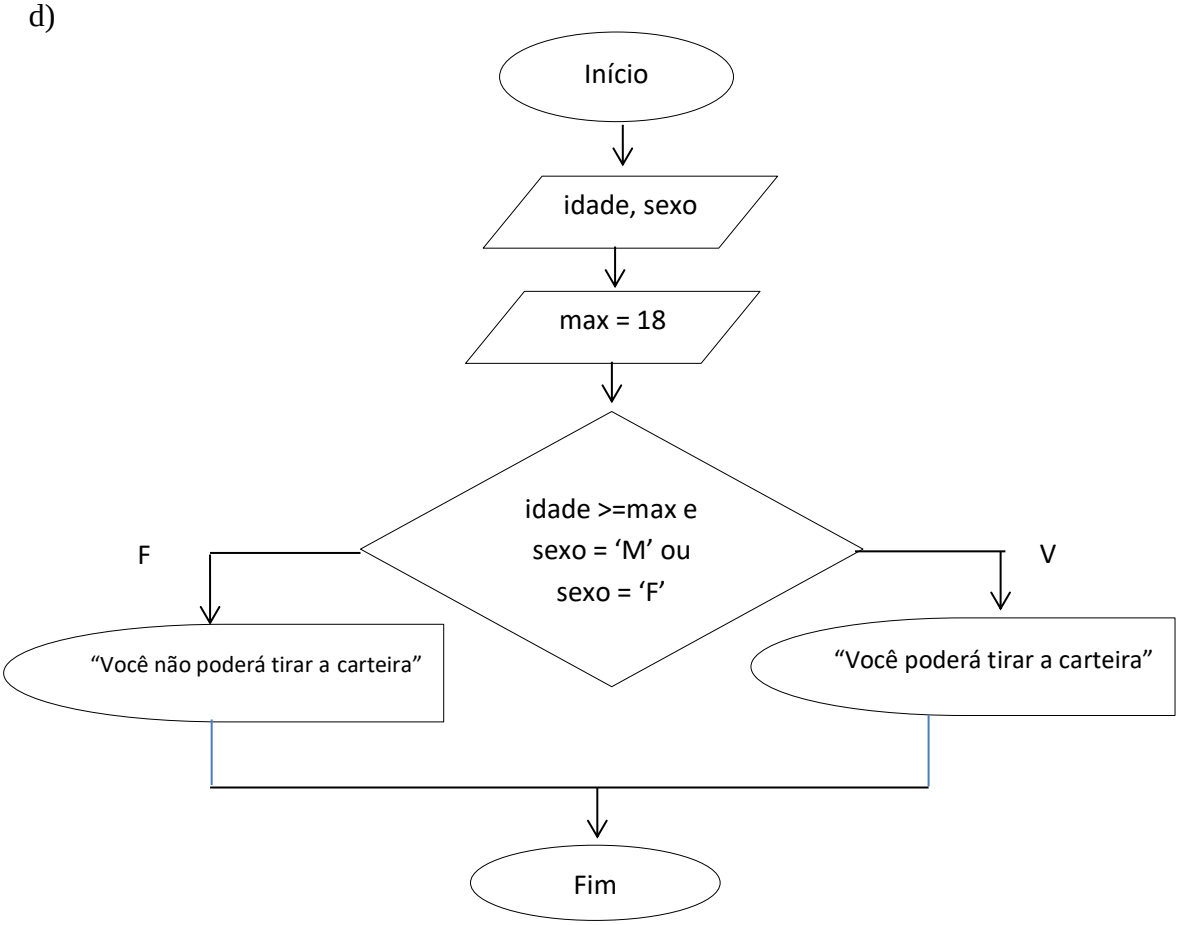
c)



Conforme as entradas abaixo identifiquem as respectivas saídas.

a) Se forem lidos 4, -2 e 20, o que será escrito pelo algoritmo?
b) Se forem lidos 8, 10 e 15, o que será impresso pelo algoritmo?
c) Se forem lidos os valores 12, 24 e -8 o que será impresso pelo algoritmo?
d) Se forem lidos 5, 5 e 0, o que será impresso pelo algoritmo?
d) Que valores deveriam ser lidos para que seja impresso na tela o valor de B?

d)



Conforme as entradas abaixo identifiquem as respectivas saídas.

a) idade= 12, sexo = 'M'. Saída:_____
b) idade= 19, sexo = 'F'. Saída:_____
c) idade= 19, sexo = 'M'. Saída:_____
d) idade= 21, sexo = 'M'. Saída:_____
e) idade =15 e sexo ='F'. Saída:_____

Tarefa 2: Desenvolva as questões abaixo utilizando fluxogramas.

a) Faça um fluxograma que leia o nome, o sexo (F ou M) e o estado civil de uma pessoa. Caso o sexo seja "F" e estado civil seja "casada", solicitar o tempo de casada (anos) e mostrar os dados no final do programa.

b) Para receber certo benefício, uma pessoa precisa ter ao menos 3 filhos, ou ter mais de 65 anos de idade. Crie um fluxograma que leia a idade e o número de filhos de uma pessoa, e verifique se essa pessoa tem direito ou não ao benefício.

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Aprendizagem:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Foco nos estudos:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
○ 1 (Nada trabalhosa) ○ 2 (Pouco trabalhosa) ○ 3 (Trabalhosa) ○ 4 (Muito trabalhosa) ○ 5 (Excessivamente trabalhosa)

TURMA INFO - 1B

Nome:_____

Data: ___/___/_____

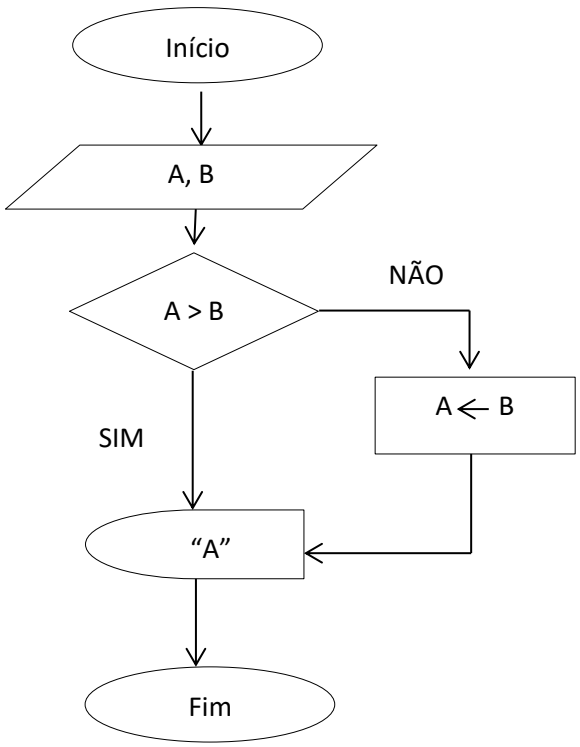
Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

1. Desenvolvam as tarefas conforme o solicitado.

Tarefa 1: Diga que valores serão impressos, após a execução do algoritmo, conforme as entradas dadas abaixo. Nas letras a, b, c e d ao lado de cada símbolo identifique-os com o seu nome.

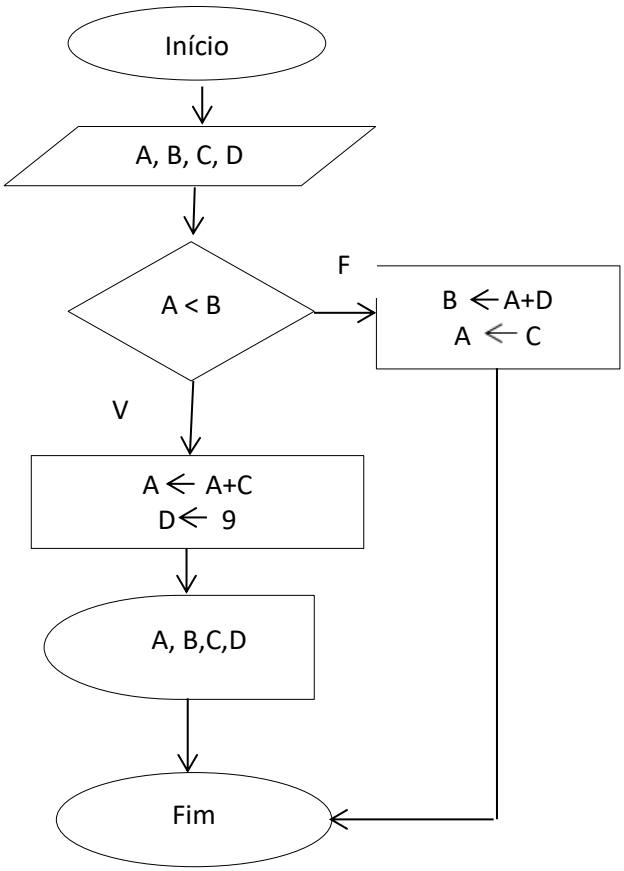
a)



Conforme as entradas abaixo escrevam ao lado a saída correspondente.

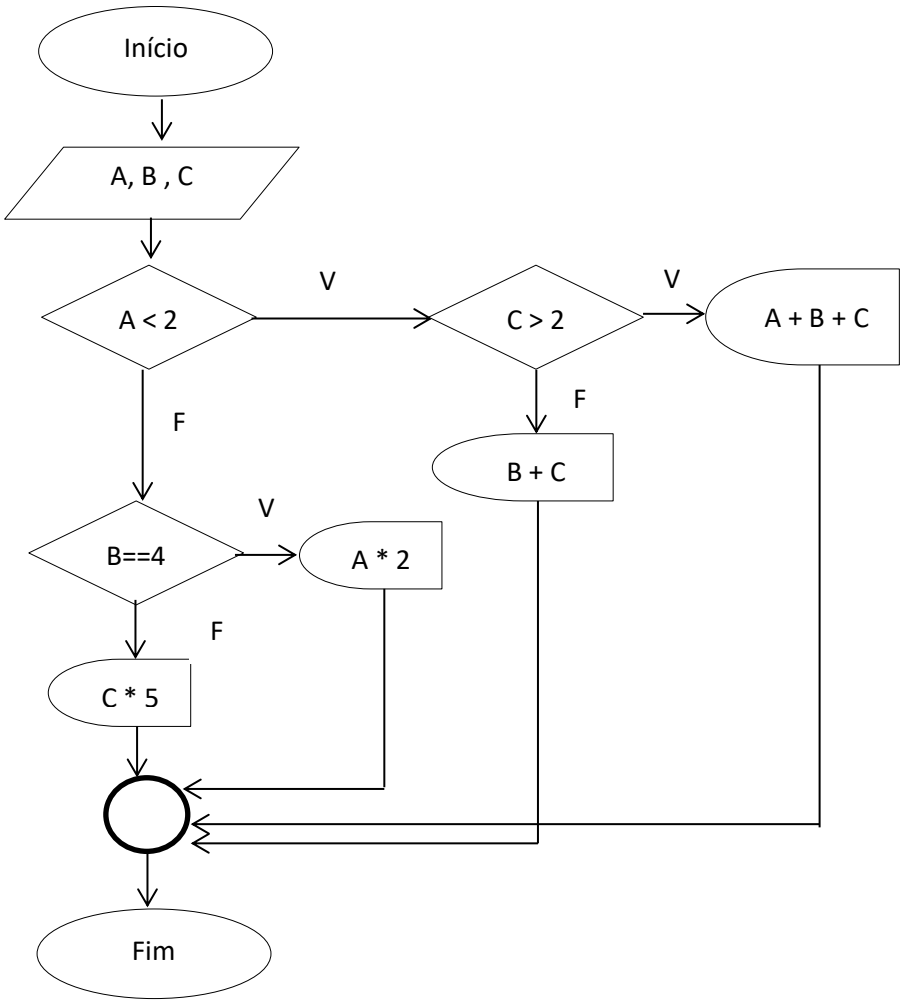
a) 8 e 7 - Saída: _____
b) 9 e -3 - Saída: _____
c) -7 e 2 - Saída: _____
d) 0 - Saída: _____
e) -22 3 -40 - Saída: _____

b)



a) A=50, B=10 e C=44 e D=75. Saída_____
b) A=-30, B=20 e C=-2 e D=-1. Saída_____
c) A=10, B=-20 e C=40 e D=-4. Saída_____
d) A=70, B=4 e C=-3 e D=40. Saída_____
e) A=-1, B=-10 e C=20 e D=10. Saída_____

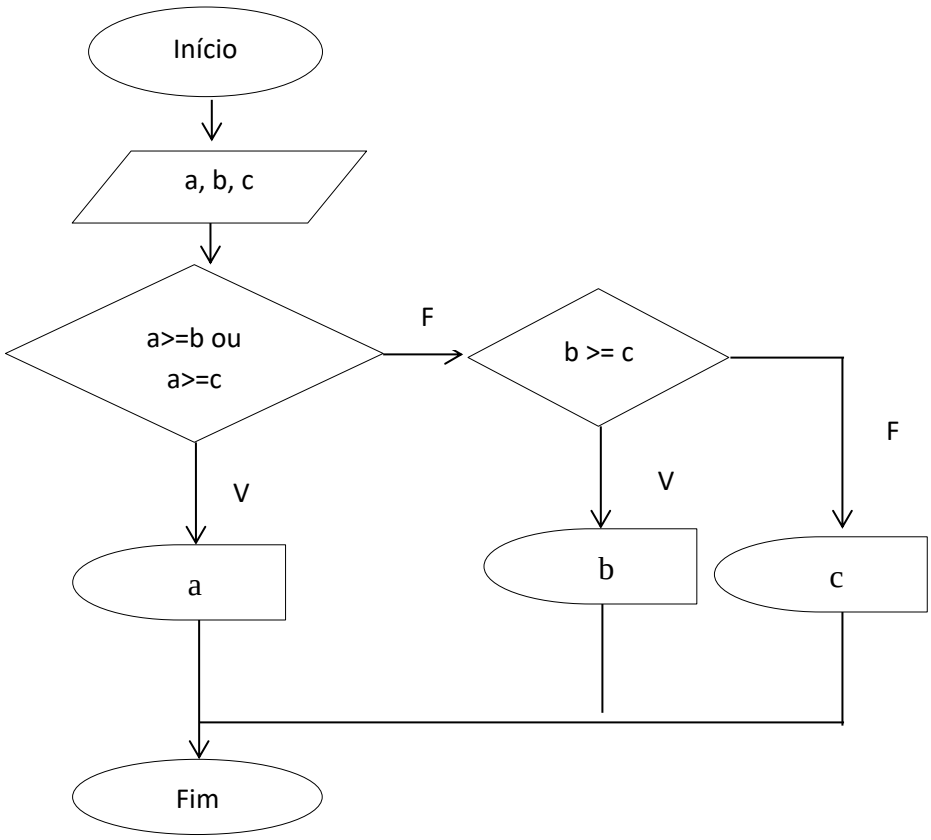
c)



Conforme as entradas abaixo identifiquem as respectivas saídas.

a) Se forem lidos 1, 8 e 7, o que será escrito pelo algoritmo?
b) Se forem lidos 5, 4 e -4, o que será escrito pelo algoritmo?
c) Se forem lidos -2, 1 e 2, o que será escrito pelo algoritmo?
d) Que valores deveriam ser lidos para que fosse impresso ao usuário apenas o valor de "C”?

d)



Conforme as entradas abaixo identifiquem as respectivas saídas.

a) a= 12, b = 6 , c=25. Saída:_____
b) a= 4, b = 10, c=8. Saída:_____
c) a= 24, b = 15, c=21. Saída:_____
d) a= E, b = F , c=0. Saída:_____
e) a=5 , b = 5, c=5. Saída:_____

Tarefa 2: Desenvolva as questões abaixo utilizando fluxogramas.

a) Leia três números reais e se o primeiro número for igual 1, escreva "A", senão verifique se o segundo número é igual a 1, se verdadeiro verifique se o terceiro número também possui o valor 1, se verdadeiro escreva "B". Caso o terceiro número não for igual a 1, escreva C e D. E se o segundo número for diferente de 1 escreva "E".

b) Faça um algoritmo para ler: a descrição do produto (nome), a quantidade adquirida e o preço unitário. De acordo com as condições abaixo mostre ao usuário o desconto por ele recebido:
   - Se quantidade <= 5 o desconto será de 2%
   - Se quantidade > 5 e quantidade <=10 o desconto será de 3%
   - Se quantidade > 10 ou quantidade < 20 o desconto será de 5%

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Aprendizagem:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Foco nos estudos:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
○ 1 (Nada trabalhosa) ○ 2 (Pouco trabalhosa) ○ 3 (Trabalhosa) ○ 4 (Muito trabalhosa) ○ 5 (Excessivamente trabalhosa)

TURMA INFO - 1C

Nome:_____

Data: ___/___/_____

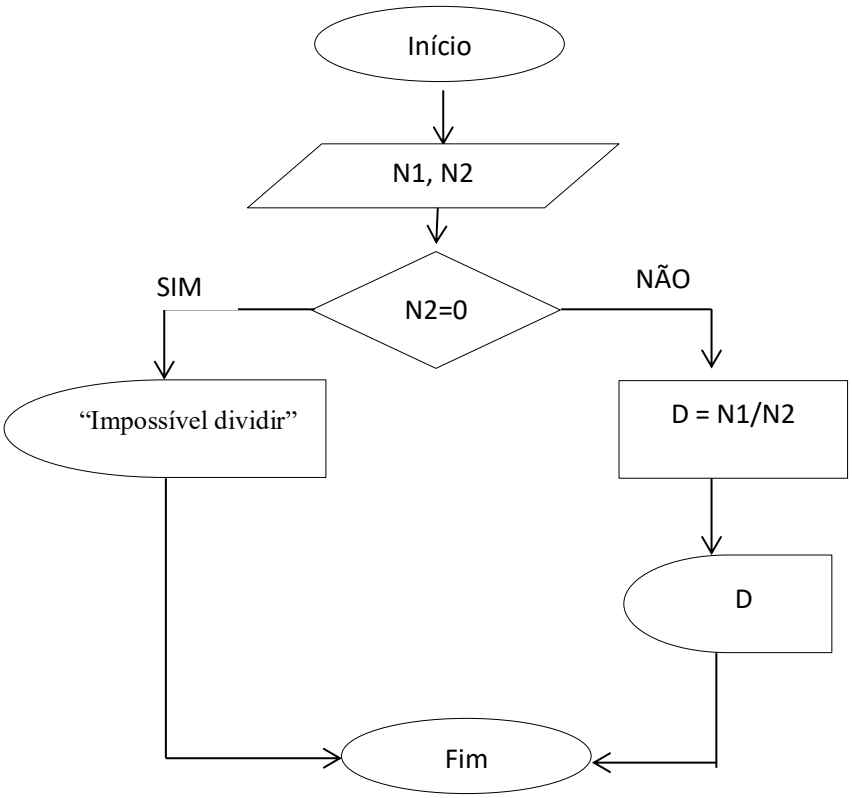
Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

1. Desenvolvam as tarefas conforme o solicitado.

Tarefa 1: Diga que valores serão impressos, após a execução do algoritmo, conforme as entradas dadas abaixo. Nas letras a, b, c e d ao lado de cada símbolo identifique-os com o seu nome.

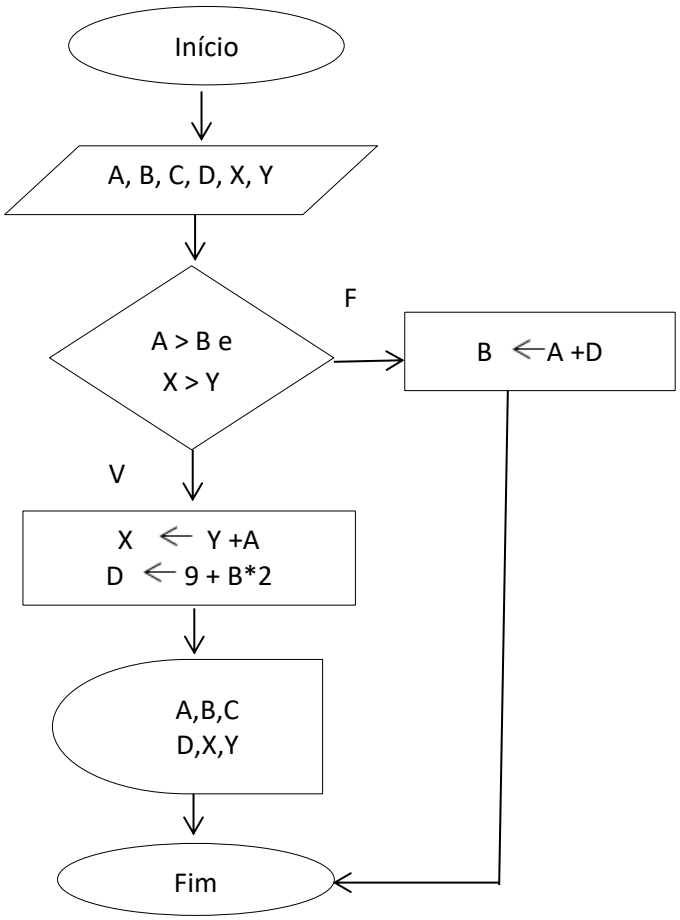
a)



Conforme as entradas abaixo escrevam ao lado a saída correspondente.

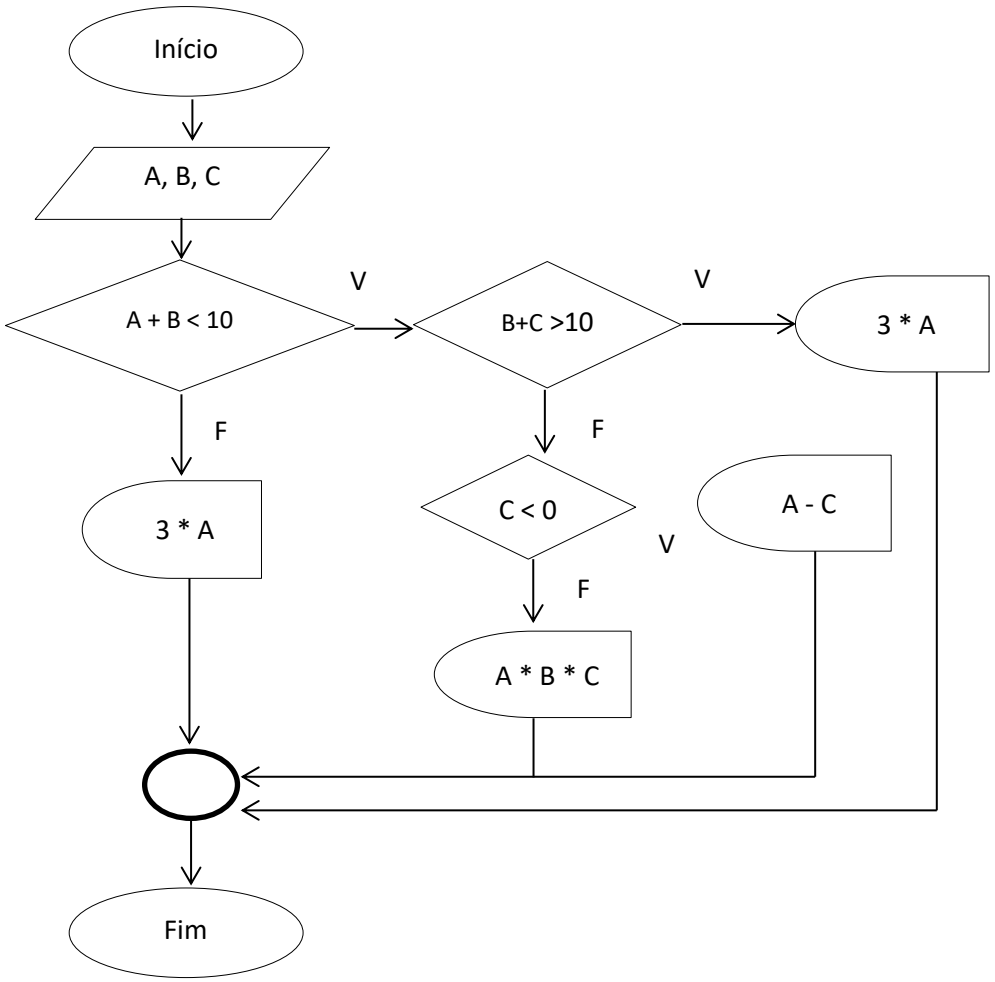
a) 5 e 10 - Saída: _____
b) A e 2 - Saída: _____
c) -2 e -15 - Saída: _____
d) 0 e 2 - Saída: _____
e) -35 - Saída: _____

b)



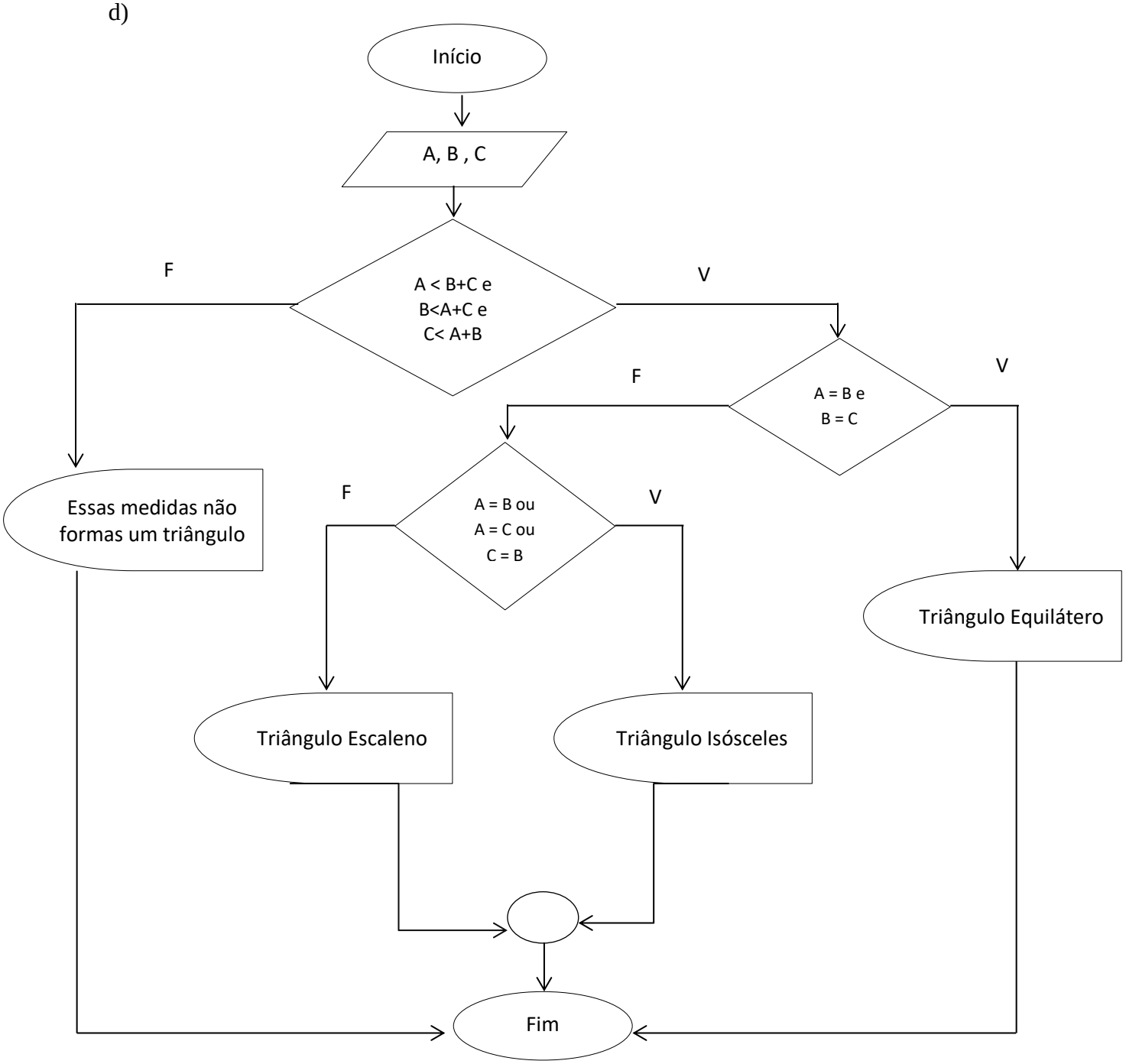
a) A=62, B=30, C=44, D=75, X=4 e Y=3. Saída_____
b) A=-20, B=-2, C=-40, D=35, X=8 e Y=10.Saída_____
c) A=40, B=-20, C=-90, D=20, X=-1 e Y=-2. Saída_____
d) A=-2, B=10, C=8, D=-10, X=15 e Y=-10.Saída_____
e) A=10, B=2, C=-30, D=-15, X=60 e Y=200. Saída_____

c)



a) Se forem lidos 3, 5 e 8, o que será escrito pelo algoritmo?
b) Se forem lidos -4, -6 e 8, o que será escrito pelo algoritmo?
c) Se forem lidos 89, 2 e 0, o que será escrito pelo algoritmo?
d) Que valores deveriam ser lidos para que fosse impresso ao usuário os valores de "A", “B” e “C”?

d)



Conforme as entradas abaixo identifiquem as respectivas saídas.

f) a= 10, b = 9 , c=5 . Saída:_____
g) a= 17, b =17 ,c=17. Saída:_____
h) a= 2, b =-5 , c=20. Saída:_____
i) a= 9, b =8 , c=8. Saída:_____
j) a= x, b =z , c=2. Saída:_____

Tarefa 2: Desenvolva as questões abaixo utilizando fluxogramas.

a) Faça um fluxograma que tendo como dados de entrada o preço de custo de um produto e um código de origem, emita o preço junto de sua procedência. Caso o código não seja nenhum dos especificados, o produto deve ser classificado como importado. Código de origem: 1-Sul, 2-Norte, 3-Leste, 4- Oeste, 5 ou 6-Nordeste, 7 ou 8-Centro-Oeste.

b) Desenvolva um fluxograma que entre com a idade e o tempo de serviço dos funcionários de uma empresa. Se ele preencher uma das condições abaixo ele está apto a se aposentar, mostrando a mensagem "Requerer aposentadoria" caso contrário mostrar a mensagem "Não requerer aposentadoria".

- Ter no mínimo 65 anos de idade.
- Ter trabalhado no mínimo 30 anos.
- Ter no mínimo 60 anos e ter trabalhado no mínimo 25 anos.

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Aprendizagem:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Foco nos estudos:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
o 1 (Nada trabalhosa) o 2 (Pouco trabalhosa) o 3 (Trabalhosa) o 4 (Muito trabalhosa) o 5 (Excessivamente trabalhosa)

APÊNDICE C – ATIVIDADES DIDÁTICAS COMANDO IF..ELSE - LINGUAGEM DE PROGRAMAÇÃO C

TURMA INFO 1A

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: Completa ☐ Parcial ☐ Não realizada ☐

Observação: Todas as atividades devem ser desenvolvidas e enviadas pelo sistema de aprendizagem SIGAA.

Tarefa 1: Analise os trechos de código abaixo e identifique a função de cada comando ao lado de cada linha na forma de comentário.

a)

```
#include <stdio.h>
#include <conio.h>

void main()
{
  float NotaDaP1, NotaDaP2;
  float Media;
  printf(“Entre com a primeira nota”);
  scanf(“%d”,&NotaDaP1);
  printf(“Entre com a segunda nota”);
  scanf(“%d”,&NotaDaP2);
  Media = (NotaDaP1 + NotaDaP2) / 2.0;
  printf("Média Final : %6.3f", Media);
  getch();
 }
```

b)

```
#include <stdio.h>
#include <stdlib.h>
int main (void)
{
  int N1, N2 ;
  printf("Digite o primeiro numero: ");
  scanf("%d", &N1);
  printf("Digite o segundo numero: ");
  scanf("%d", &N2);
```

```
 if (N1 == N2)
     printf("Os numeros sao iguais!");
  else
  if (N1 > N2)
     printf("O maior valor e = %d", N1);
 else
 printf("O maior valor e = %d", N2);

 printf("\n");
 system("pause");
}
```

c) Elaborar um programa em linguagem C para somar dois números inteiros e mostrar o valor da soma na tela. Caso a soma dos números seja maior que 10 mostrar uma mensagem informativa na tela.

Tarefa 2: Analise os códigos abaixo e informe o que será impresso pelo programa. Após, escreva um enunciado que irá guiar outra pessoa a construir o mesmo algoritmo.

a)

```
#include
int main()
{
   float preco = 1.99;
   int pera = 3;
   char qualidade = 'A';
   float peso = 2.5;
   printf("Existem %d peras de qualidade %c ", pera, qualidade);
   printf("pesando %f quilos.\n", peso);
   printf("O preco por quilo e %f, total e %f\n", PRECO, peso * PRECO);
}
```

A saída do programa será:

Enunciado:

b)

```
#include <stdio.h>
#include <stdib.h>
 int main()
{
        int idade;
        printf("Digite sua idade");
```

```
scanf(“%d”,&idade);

if((idade >=0) && (idade <18))
{
     printf(“Não possui carteira de habilitação.\n”);
}
else
if ((idade >= 18) && (idade <20))
{
     printf(“Renove, exames a cada 5 anos.\n”);
}
else
if (idade > = 65)
{
    printf(“Renove exames a cada 3 anos.\n”);
}
return 0;
```

Como ficará a saída do programa com as seguintes entradas:

Primeiro exemplo de execução:
Digite sua idade: 15
Saída:

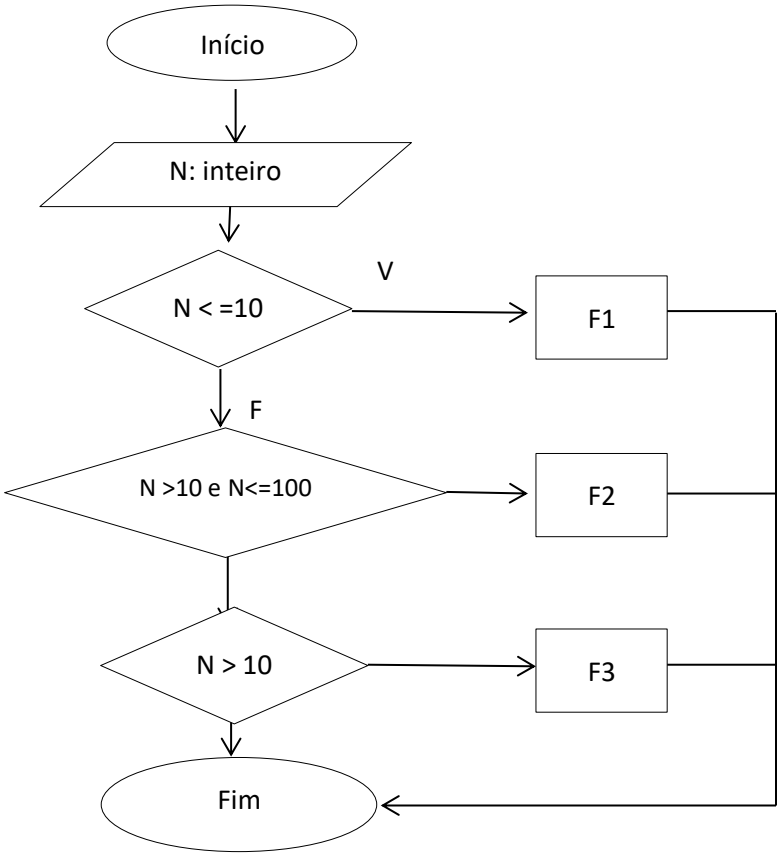
Segundo exemplo de execução:
Digite sua idade: 25
Saída:

Terceiro exemplo de execução:
Digite sua idade: 75
Saída:

Enunciado:

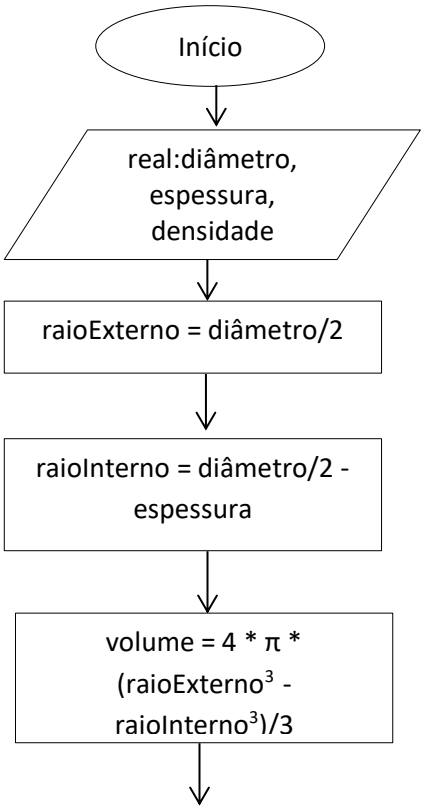
Tarefa 3: Analise os fluxogramas abaixo e codifique na linguagem de programação C.

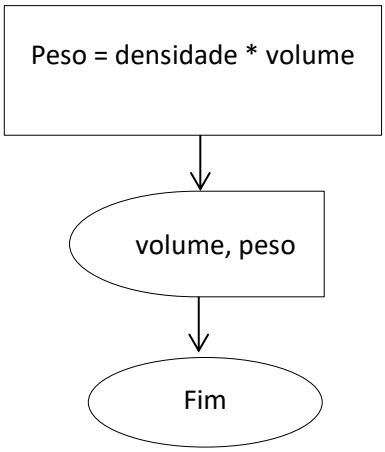
a)



Na linguagem C:

b)





Na linguagem C:

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Aprendizagem:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Foco nos estudos:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
o 1 (Nada trabalhosa) o 2 (Pouco trabalhosa) o 3 (Trabalhosa) o 4 (Muito trabalhosa) o 5 (Excessivamente trabalhosa)

TURMA INFO 1B

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Observação: Todas as atividades devem ser desenvolvidas e enviadas pelo sistema de aprendizagem SIGAA.

Tarefa 1: Analise os trechos de código abaixo e identifique a função de cada comando ao lado de cada linha na forma de comentário.

a)

```
#include <stdio.h>

int main()
```

```
{
   float num1, num2, resultado;
   printf("Digite um numero: ");
   scanf("%f", &num1);

   printf("Digite outro numero: ");
   scanf("%f", &num2);

   resultado = num1 * num2;

   printf("%f + %f = %f", num1,num2,resultado);
}
```

b)

```
#include <stdio.h>
int main ()
{
        int num;
        printf ("Digite um numero: ");
        scanf ("%d",&num);
        if (num>10)
            printf ("\n\nO numero e maior que 10");
        else
        if (num==10)
        {
            printf ("\n\nVoce acertou!\n");
            printf ("O numero e igual a 10.");
        }
        else
        if (num<10)
            printf ("\n\nO numero e menor que 10");
}
```

c) Desenvolva um programa que verifique se um número inserido pelo usuário é múltiplo de 5. Se for verdadeiro imprima “MÚLTIPLO DE CINCO” caso contrário imprima “NÃO É MÚLTIPLO DE CINCO”.

Tarefa 2: Analise os códigos abaixo e informe o que será impresso pelo programa. Após, escreva um enunciado que irá guiar outra pessoa a construir o mesmo algoritmo.

a)

```
#include<stdio.h>
int main()
{
   float qtd_voltas, raio_roda,dist_percorrida;
   float comprimento,pi=3.14;
```

```
    raio_roda = 22
    qtd_voltas= 12;
    comprimento = 2*pi*raio_roda;
    dist_percorrida=comprimento*qtd_voltas;
    printf("comprimento da roda=%.1f",comprimento);
    printf("\nDistância percorrida: %8.1f cm",dist_percorrida);
    printf("\n Que equivale a %.1f metros",dist_percorrida/100);
   return 0;
}
```

A saída do programa será:

Enunciado:

b)

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    int a, b, c, d, e;
    printf(“Entre com os números abaixo:”);
    scanf(“%d,%d,%d,%d”, &a,&b,&c,&d);
     if((a == 0) && (b == 2))
     {
     if(c > 0)
         printf("Comando 1\n");
     else
     if(d >= 10)
    {
     if(e < 0)
         printf("Comando 2\n");
     else
       printf("Comando 3\n");
    }
    }
    printf("O programa sera encerrado.\n");
}
```

Como ficará a saída do programa com as seguintes entradas:

Primeiro exemplo de execução:
Entradas: a = 0, b = 2, c = 0, d = 10, e = -8
Saída:

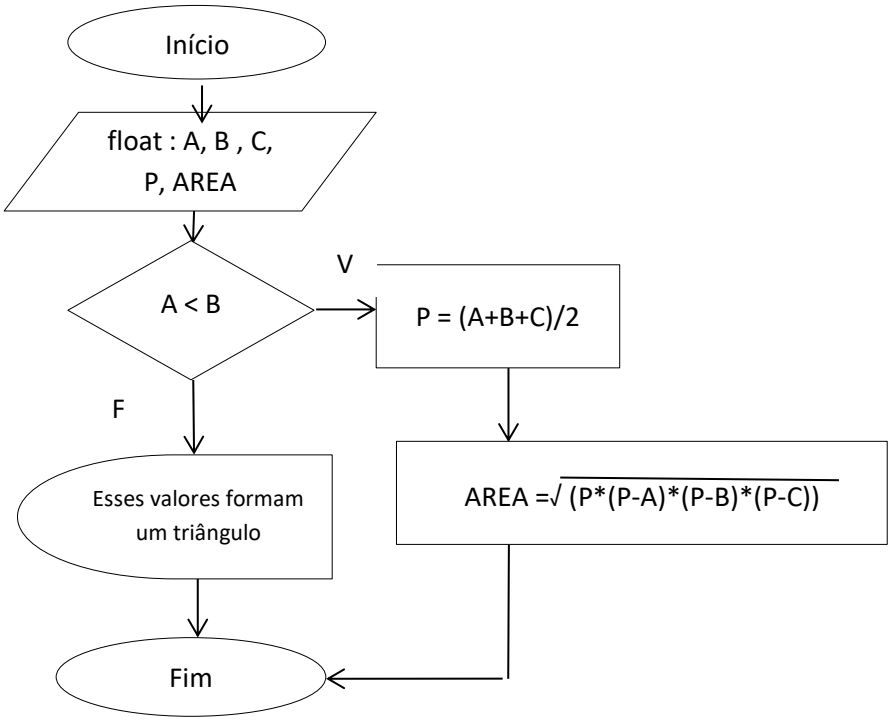
Segundo exemplo de execução:
Entradas: a = 2, b = 5, c = 4, d = 10, e = 20
Saída:

Terceiro exemplo de execução:
Entradas: a = 0, b = 2, c = 9, d = 3, e = -5
Saída:

Enunciado:

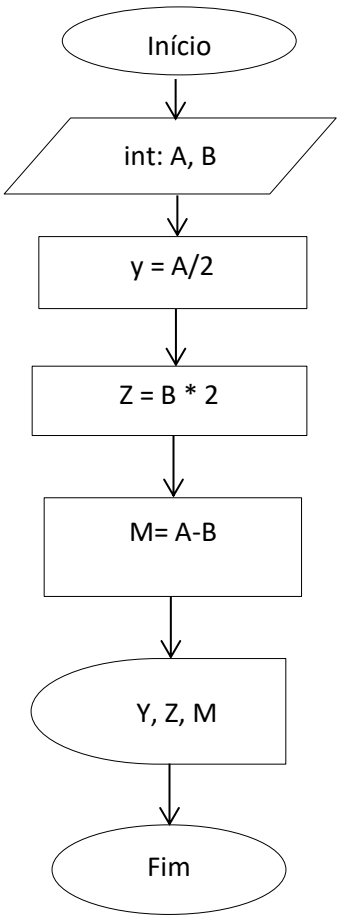
Tarefa 3: Analise os fluxogramas abaixo e codifique na linguagem de programação C.

a)

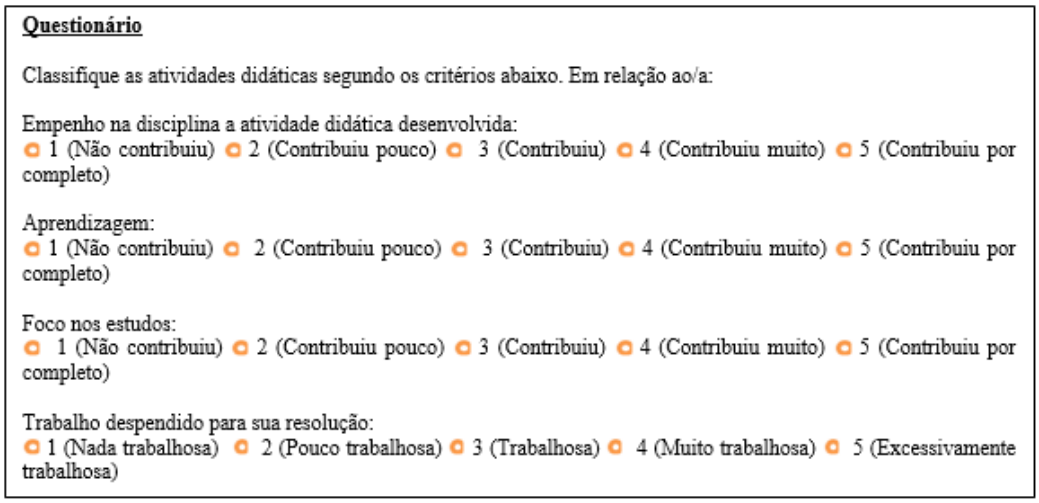


Na linguagem C:

b)



Na linguagem C:



Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Aprendizagem:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Foco nos estudos:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
○ 1 (Nada trabalhosa) ○ 2 (Pouco trabalhosa) ○ 3 (Trabalhosa) ○ 4 (Muito trabalhosa) ○ 5 (Excessivamente trabalhosa)

TURMA INFO 1C

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Observação: Todas as atividades devem ser desenvolvidas e enviadas pelo sistema de aprendizagem SIGAA.

Tarefa 1: Analise os trechos de código abaixo e identifique a função de cada comando ao lado de cada linha na forma de comentário.

a)

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
main()
{
    int x, n1, n2;
    printf("\n\n Digite um numero: ");
    scanf("%d",&x);
    n1=x+1;
    n2=x-1;
```

```
        printf("\n\nSeu sucessor e : %d",n1);
        printf("\n\nSeu antecessor e : %d",n2);
        system("pause");
}
```

b)

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
        float numero1, numero2;
        printf("Informe o primeiro numero: ");
        scanf("%f",&numero1);
        printf("Informe o segundo numero: ");
        scanf("%f",&numero2);
        if ((numero1 + numero2) > 10)
              printf("\nA soma dos numeros informados e %f", numero1 + numero2);
        system("PAUSE");
}
```

c) Desenvolva um programa na linguagem C que receba um valor qualquer do teclado e imprima esse valor com reajuste de 10%.

Tarefa 2: Analise os códigos abaixo e informe o que será impresso pelo programa. Após, escreva um enunciado que irá guiar outra pessoa a construir o mesmo algoritmo.

a)

```
#include<stdio.h>

int main()
{
float salar_min,salar_pessoa, qtos_sal_ganha;

printf("Digite o valor do salario minimo ",salar_min);
scanf("%f",&salar_min);
printf("Digite o valor do salario recebido ",salar_pessoa);
scanf("%f",&salar_pessoa);
```

```
qtos_sal_ganha=(salar_pessoa/salar_min);
printf("\n Por receber R$ %5.1f - Ganha %.1f salarios
minimos\n",salar_pessoa,qtos_sal_ganha);

return 0;
}
```

A saída do programa será:

Enunciado:

b)

```
#include<stdio.h>

main(){

float Valor_teste,horas_extras, Faltas,aumento;
char nome;

printf("Entre com o nome da pessoa");
scanf("%c",&nome);

printf("Entre com o valor das variáveis horas_extra e Faltas");
scanf("%f,%f",&horas_extras,&Faltas);

Valor_teste = horas_extras - (2/3) * Faltas ;

if (Valor_teste > 40)
{
aumento=50;
printf("\nSera pago 50 reais de gratificação");
}
if((Valor_teste > 30) &&(Valor_teste< 40))
{
aumento=40;
printf("\nSera pago 40 reais de gratificação");
}
if ((Valor_teste > 20) &&(Valor_teste <30))
{
aumento=30;
printf("\nSera pago 30 reais de gratificação");
}
if ((Valor_teste > 10)&& (Valor_teste <20))
{
aumento=20;
printf("\nSera pago 20 reais de gratificação");
}
if(Valor_teste <= 10)
```

```
{
aumento=10;
printf("\nSera pago 10 reais de gratificação");
}

printf("%c tera um aumento de %f",nome,aumento);
printf("Numero de faltas %f e valor das horas extras %f",Faltas,horas_extras);
}
```

Como ficará a saída do programa com as seguintes entradas:

Primeiro exemplo de execução:
Entradas: A, 20, 2.
Saída:

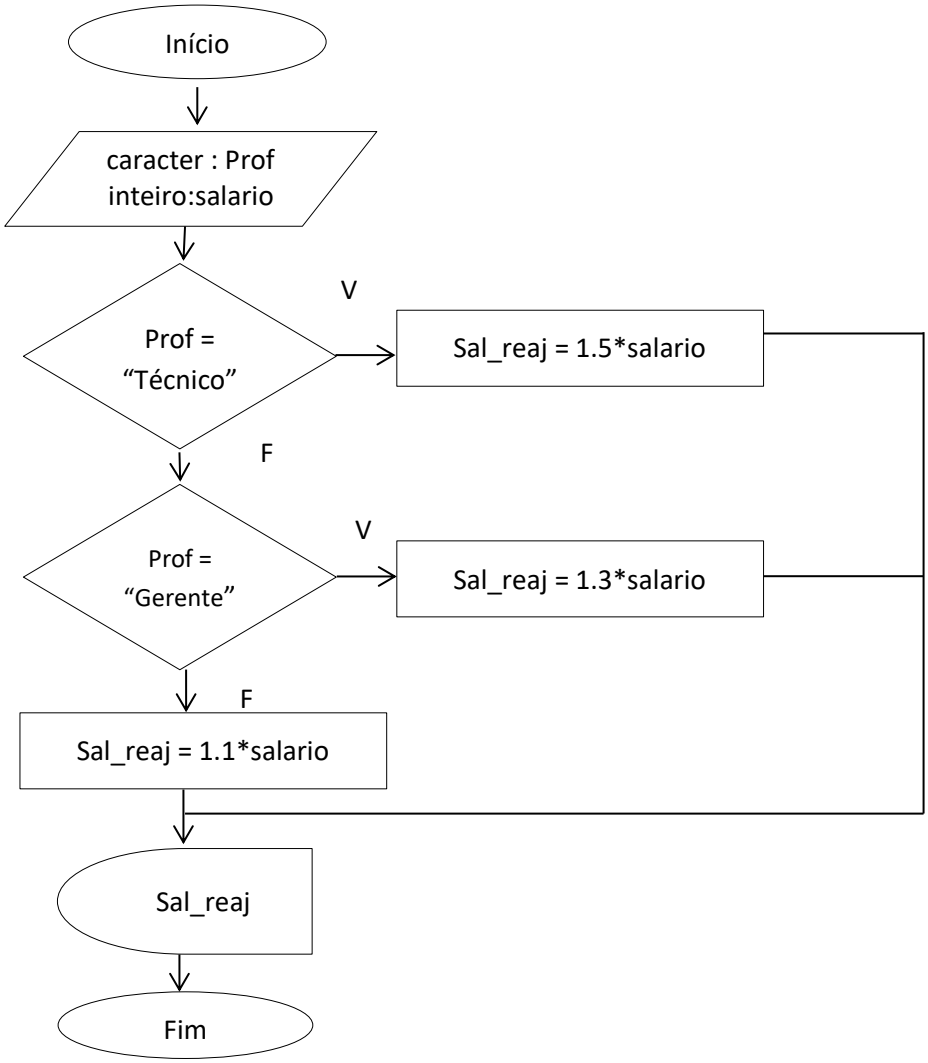
Segundo exemplo de execução:
Entradas: B, 15,10.
Saída:

Terceiro exemplo de execução:
Entradas: D, 30,0.
Saída:

Enunciado:

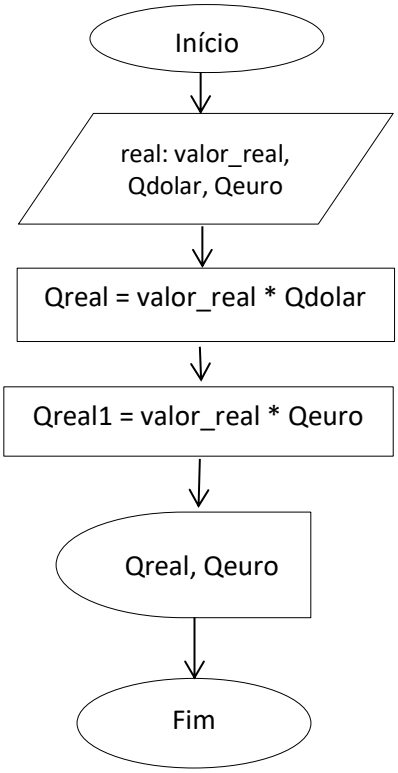
Tarefa 3: Analise os fluxogramas abaixo e codifique na linguagem de programação C.

a)



Na linguagem C:

b)



Na linguagem C:

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Aprendizagem:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Foco nos estudos:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
o 1 (Nada trabalhosa) o 2 (Pouco trabalhosa) o 3 (Trabalhosa) o 4 (Muito trabalhosa) o 5 (Excessivamente trabalhosa)

APÊNDICE D – ATIVIDADES DIDÁTICAS SWITCH...CASE

TURMA INFO 1A

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: Todas as tarefas devem ser enviadas pelo ambiente de aprendizagem SIGAA.

Tarefa 1: Analisando o exercício abaixo, identifique a melhor estrutura estudada até o momento e construa o programa utilizando a linguagem de programação C. Justifique sua escolha.

a) Uma loja fornece 10% de desconto para funcionários e 5% de desconto para clientes vips. Faça um programa que calcule o valor total a ser pago por uma pessoa. O programa deverá ler o valor total da compra efetuada e um código que identifique se o comprador é um cliente comum, funcionário ou vip.

Na linguagem C:

Justificativa:

Tarefa 2**:** Analise os códigos abaixo e reescreva-os utilizando o comando switch..case.

a)
```
#include <stdio.h>
#include<stdlib.h>
int main() {
int x;
printf("Escolha o codigo do produto\n");
printf("1 - Vestuario\n");
printf("2 - Higiene Pessoal\n");
printf("3 - Produto perecivel\n");
printf("4 - Produto nao perecivel\n");
scanf("%d",&x);
if (x==1) {
 printf("Voce quer comprar uma blusa?\n");
}
 else
if(x==2){
printf("Voce quer comprar um creme dental?\n");
} else
if(x==3) {
printf("Voce quer comprar um kg de carne?\n");
}else
```

```
if(x==4){
printf("Voce quer comprar uma lata de oleo ?\n");
}
}
```

Reescrevendo o código:

b) #include<stdio.h>

```
int main() {
float compras, desconto, taxa, totpagar;

printf(“Entre com o valor da compra”);
scanf(“%f”,&compras);

if(compras <= 2000) {
  taxa=0.1;
}
else {
  if (compras <= 3000) {
    taxa = 0.05;
  }
  else {
    if (compras <= 5000) {
      taxa = 0.03;
    }
    else{
      taxa=0.02;
    }
  }
}
desconto = compras * taxa;
totpagar = compras - desconto;
printf("O seu desconto foi de %f e você ira pagar %f reais.", desconto, totpagar);
}
```

Reescrevendo o código:

Tarefa 3: Transforme o pseudocódigo abaixo em um programa que deve ser desenvolvido utilizando a linguagem de programação C.

a) algoritmo “semnome”

```
var
n1, n2:real
inicio
escreval ("Digite o valor do primeiro numero")
leia(n1)
```

```
escreval ("Digite o valor do segundo numero")
leia(n2)
se n1 > n2 entao
escreval("Primeiro Numero MAIOR")
senao
escreval("Segundo Numero MAIOR")
fimse
fimalgoritmo
```

Código na linguagem C:

b)
```
algoritmo preferencia
var sexo:inteiro
inicio
escreva("Entre com o sexo 1 para Masculino e 2 para Feminino")
leia(sexo)
escolha
caso 1:
escreva("Futebol")
caso 2:
escreva("Culinária")
senao:
escreva("Opção inválida");
fim
```

Código na linguagem C:

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Aprendizagem:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Foco nos estudos:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
o 1 (Nada trabalhosa) o 2 (Pouco trabalhosa) o 3 (Trabalhosa) o 4 (Muito trabalhosa) o 5 (Excessivamente trabalhosa)

TURMA INFO 1B

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: Todas as tarefas devem ser enviadas pelo ambiente de aprendizagem SIGAA.

Tarefa 1: Analisando o exercício abaixo, identifique a melhor estrutura estudada até o momento e construa o programa utilizando a linguagem de programação C. Justifique sua escolha.

a) Um funcionário irá receber um aumento de acordo com o seu plano de trabalho. Plano A aumento de 10%, plano B aumento de 15% e plano C aumento de 20%. Faça um programa que leia o plano de trabalho e o salário atual de um funcionário e calcula e imprime o seu novo salário.

Na linguagem C:

Justificativa:

Tarefa 2**:** Analise os códigos abaixo e reescreva-os utilizando o comando switch..case.

```
#include <include stdio.h>
#include <stdlib.h>
void main() {
int destino, trecho;
printf("Informe o destino conforme tabela a seguir: \n");
printf("1-Regiao Norte \t 3-Regiao Centro-oeste \n");
printf("2-Regiao Nordeste \t 4-Regiao Sul \n");
scanf("%d", &destino);
printf("Informe o trecho: ");
scanf("%d", &trecho);
if (destino == 1) {
  if (trecho == 1)
      printf("Regiao norte[IDA] = 500,00");
  else
     printf("Regiao norte[IDA E VOLTA] = 900,00");
}
else
if (destino == 2) {
    if (trecho == 1)
         printf("Regiao nordeste[IDA] = 350,00");
           else
        printf("Regiao nordeste[IDA E VOLTA] = 650,00");
}
else
```

```
if (destino == 3) {
   if (trecho == 1)
       printf("Regiao centro-oeste[IDA] = 350,00");
   else
      printf("Regiao centro-oeste[IDA E VOLTA] = 600,00");
}
else {
if (trecho == 1)
     printf("Regiao sul[IDA] = 300,00");
       else
    printf("Regiao sul[IDA E VOLTA] = 550,00");
 } }
```

Reescrevendo o código:

b)

```
#include <stdio.h>
    main() {
    float SAtual, SNovo, Indice;
    printf("Digite o salário atual");
    scanf("%f", &SAtual);
    if (SAtual <= 200)
    Indice = 1.13;
    else
    if (SAtual <= 400)
    Indice = 1.11;
    else
    if (SAtual <= 800)
    Indice = 1.09;
    else
    Indice = 1.07;
    SNovo = SAtual*Indice;
    printf("Atual = %.2f \n Novo = %.2f \n" , SAtual, SNovo);
    }
```

Reescrevendo o código:

Tarefa 3: Transforme o pseudocódigo abaixo em um programa que deve ser desenvolvido utilizando a linguagem de programação C.

a) algoritmo tarifaestacionamento

```
var
tempo:inteiro
inicio
escreval ("Digite o tempo em minutos")
leia(tempo)
se (tempo <=15) entao
escreval("Tarifa Grátis")
senao
```

```
        se(tempo <=60) então
            escreval("Valor a pagar: R$ 1,00")
        senao
            escreval("Valor a pagar: R$ 2,00")
        fimse
    fimse
fimalgoritmo
```

Código na linguagem C:

b) algoritmo descobrindoprofissao
var nome:char
inicio
escreva("Escolha o nome Pedro, Maria, Alice e Carlos")
leia(nome)
escolha
caso "Pedro":
escreva("Gerente")
caso "Maria":
escreva("Vendedora")
caso "Alice":
escreva("Supervisora")
caso "Carlos":
escreva("Diretor")
senao:
escreva("Usuário Desconhecido");
fim

Código na linguagem C:

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Aprendizagem:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Foco nos estudos:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
○ 1 (Nada trabalhosa) ○ 2 (Pouco trabalhosa) ○ 3 (Trabalhosa) ○ 4 (Muito trabalhosa) ○ 5 (Excessivamente trabalhosa)

TURMA INFO 1C

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: Todas as tarefas devem ser enviadas pelo ambiente de aprendizagem SIGAA.

Tarefa 1: Analisando o exercício abaixo, identifique a melhor estrutura estudada até o momento e construa o programa utilizando a linguagem de programação C. Justifique sua escolha.

a) Faça um programa que pergunte a mesada de um adolescente, após isto pergunte se ele quer comprar algum produto, caso ele responda não, o programa irá imprimir uma mensagem mostrando qual o valor da mesada dele, caso responda sim, o programa irá perguntar qual o valor do produto e mostrar o saldo dele.

Na linguagem C:

Justificativa:

Tarefa 2**:** Analise os códigos abaixo e reescreva-os utilizando o comando switch..case.

a)

```
#include <stdio.h>
#include <stdlib.h>
  main() {
 int cargo;
 float salAtual, reajuste;
 printf("Informe o cargo do funcionario:");
 scanf("%d", &cargo);
 printf("Informe o salario atual:");
 scanf("%f", &salAtual);
 if (cargo == 1)
        reajuste = (salAtual * 7) / 100;
 else
if (cargo == 2)
        reajuste = (salAtual * 9) / 100;
 else
if (cargo == 3)
        reajuste = (salAtual * 5) / 100;
else
reajuste = (salAtual * 12) / 100;
printf("O reajuste e: %f", reajuste);
printf("O novo salario e: %f", salAtual + reajuste); }
```

Reescrevendo o código:

b)

```
#include<stdio.h>
#include<math.h>

int main()
{
int x, y;
scanf("%d",&x);
if(x < 1)
{
y = x;
printf("Y = %d\n",y);
}
else if(x == 1)
{
y = 0;
printf("Y = %d\n",y);
}
else if(x > 1)
{
y = pow(x,2);
printf("Y = %d\n",y);
}
return 0;
}
```

Reescrevendo o código:

Tarefa 3: Transforme o pseudocódigo abaixo em um programa que deve ser desenvolvido utilizando a linguagem de programação C.

a)

```
algoritmo "SacarDinheiro"
var
   SaldoDisponivel : real
   ValorDoSaque : real
inicio
      SaldoDisponivel := 1000
      escreva ("Informe o valor do Saque: ")
      leia (ValorDoSaque)
      se (ValorDoSaque <= SaldoDisponivel) entao
         SaldoDisponivel := SaldoDisponivel - ValorDoSaque
         escreval ("Sacando R$ ", ValorDoSaque, ".")
      senao
         escreval ("O valor solicitado é maior que o valor disponível para saque!")
      fimse
```

```
    escreval ("Saldo disponível: R$ ", SaldoDisponivel)
fimalgoritmo
```

Código na linguagem C:

b)

```
algoritmo localehorariopalestra
var codigo:inteiro
inicio
escreva(“Escolha o numero a palestra:”)
escreval( “1- Linux”)
escreval( “2- Recuperação de Desastre”)
escreval(“ 3- Windows Server”)
escreval(“4- Lógica de Programação”)
leia(codigo)
escolha
caso 1:
escreva(“Auditorio 1 \n 8h as 9h”)
caso 2:
escreva(“Auditorio 2 \n 9h as 10h”)
caso 3:
escreva(“Auditorio 3 \n 13h as 14h”)
caso 4:
escreva(“Auditorio 4 \n 15h as 17h”)
senao:
escreva(“Opção Inválida”);
fimescolha
fimalgoritmo
```

Código na linguagem C:

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Aprendizagem:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Foco nos estudos:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
o 1 (Nada trabalhosa) o 2 (Pouco trabalhosa) o 3 (Trabalhosa) o 4 (Muito trabalhosa) o 5 (Excessivamente trabalhosa)

APÊNDICE E – ATIVIDADES DIDÁTICAS WHILE

TURMA INFO 1A

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: A primeira e segunda tarefa deve ser respondida na folha em que os exercícios foram apresentados. A terceira tarefa deve ser desenvolvida no computador e todas dever ser enviadas pelo ambiente de aprendizagem SIGAA.

Tarefa 1: Descreva o que os trechos de código abaixo estão executando (letra a e b) e suas respectivas saídas, após (letra c) desenvolva o algoritmo correspondente na linguagem c.

a)
```
int cont=0;
int nota = 3;
while(cont<7)
{
   nota=nota+1;
     if(nota>=7)
         printf("Aprovado");
     else
         printf("Reprovado");

   cont=cont+1;
}
```

b)
```
int i = 1;
  while (i < 10)
{
  if(i == 5)
     break;
  printf("O valor de i:%d",i);
  i++;
}
```

c) Faça um programa em C que realize a soma de todos os valores inteiros de 1 a n, sendo que n deve ser informado pelo usuário.

Tarefa 2**:** Analise os códigos abaixo e escreva suas respectivas saídas. Após, descreva o que cada programa está fazendo.

a)

```
#include<stdio.h>
#include<stdlib.h>

int main(){

   int repetir, i, cont_menor10 = 0, cont_entre10e20 = 0, cont_maior20 = 0;
   float preco_compra, preco_venda, calculo, total_venda, total_compra, lucro, aux;
   i = 0;

   scanf("%d",&repetir);

   while(i < repetir)
   {
     i++;
     scanf("%f",&preco_compra);
     scanf("%f",&preco_venda);

     total_compra = total_compra + preco_compra;
     total_venda = total_venda +preco_venda;
     lucro = total_venda – total_compra;

     aux = preco_venda – preco_compra;
     calculo = (100 * aux) / preco_compra;


       if (calculo < 10)
      {
           cont_menor10++;
       }
      else

      if ((calculo >= 10) && (calculo <= 20))
      {
          cont_entre10e20++;
      }
      else
      if (calculo > 20)
     {
          cont_maior20++;
      }
  }
 printf("%d com lucro < 10%\n",cont_menor10);
 printf("%d com lucro >= 10%% ou lucro <= 20%\n",cont_entre10e20);
 printf("%d com lucro > 20%\n",cont_maior20);
 printf("Total de compra = %.2f\n",total_compra);
 printf("Total de venda = %.2f\n",total_venda);
```

```
  printf("Lucro = %.2f\n",lucro);

return 0;
}
```

Entradas:
repetir: 3
preco_compra: 100, 200, 300
preco_venda: 50, 300, 350

O que o programa irá mostrar ao usuário

Saídas:

Descreva o funcionamento do programa:

b)

```
#include <stdio.h>
#include <conio.h>

int main()
{
   float soma=0, cont=0;
   float numero;
   printf("\nEntre com um numero: ");
   scanf("%f", &numero);

while(numero>0)
{
   soma=soma+numero;
   cont ++;
   printf("\nEntre com um numero: ");
   scanf("%f", &numero);
}

if(cont ==0)
   printf("\nNao foi entrado nenhum numero");
else
if(cont ==1)
   printf("\nFoi entrado %.0f numero", total);
else
   printf("\nForam entrados %.0f numeros", total);
if(cont!=0)
   printf("\nA media dos numeros entrados e: %f", soma/total);
getch();
}
```

De acordo com as entradas, vá colocando ao lado o número assumido pelas variáveis soma e total. Ao final, escreva o que o programa irá mostrar ao usuário.

Entradas:
numero: 10, 2, 5, 0, -2

Valores assumidos pelas variáveis abaixo conforme as entradas acima:
soma:
cont:

Saída Final:

Descreva o funcionamento do programa:

Tarefa 3: Analisando os exercícios abaixo, utilize as estruturas de comando vistas até o momento e construa o programa utilizando a linguagem de programação C.

a) Faça um programa que leia as médias finais de vários alunos de uma turma e mostre a maior média, a menor média e a média aritmética da turma. O programa para quando encontrar uma média negativa.

Código na linguagem C:

b) Desenvolva na linguagem C um programa que leia diversos números informados pelo usuário, e após cada leitura exibir se o número é par ou ímpar. O término do programa deve ser realizado pelo usuário.

Código na linguagem C:

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Aprendizagem:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Foco nos estudos:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
○ 1 (Nada trabalhosa) ○ 2 (Pouco trabalhosa) ○ 3 (Trabalhosa) ○ 4 (Muito trabalhosa) ○ 5 (Excessivamente trabalhosa)

TURMA INFO 1B

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: A primeira e segunda tarefa deve ser respondida na folha em que os exercícios foram apresentados. A terceira tarefa deve ser desenvolvida no computador e todas devem ser enviadas pelo ambiente de aprendizagem SIGAA.

Tarefa 1: Descreva o que os trechos de código abaixo estão executando (letra a e b) e suas respectivas saídas, após (letra c) desenvolva o algoritmo correspondente na linguagem c.

a)
```
int dobro;
  int cont=0;
    while(cont<8)
    {
          if(cont>0)
        {
            dobro=cont*2;
            printf("\nDobro:%d",dobro);
        }
        else
            printf("Número Negativo");
        cont=cont+1;
    }
```

b)
```
int i = 1;
    while(i < 10)
   {
     i++;
      if(i %2 == 1)
          break;
     printf("O valor de i:%d",i);
   }
```

c) Faça um programa em C que imprima todos os valores entre 0 e 100 múltiplos de 10.

Tarefa 2**:** Analise os códigos abaixo e escreva suas respectivas saídas. Após, descreva o que cada programa está fazendo.

a)

```
#include <stdio.h>
#include <stdlib.h>
   int main() {
   int numero;
   int divisor;
   int resto;
   printf("Digite o numero: ");
   scanf("%d", &numero);
   divisor = 1;
   while (divisor <= numero)
  {
        resto = numero % divisor;
           if (resto == 0)
          {
               printf("Divisor encontrado: %d \n", divisor);
          }
        divisor = divisor + 1;
   }
return 0;
}
```

Possíveis entradas:

Entrada:

Saídas:

O que ocorre se o número informado for zero? E se for negativo?

Descreva o funcionamento do programa:

b)

```
#include <stdio.h>
#include <stdlib.h>

main(){

   float PESO, ALTURA, ALTUMAS=0, PESO_FEM=0;
   int IDADE, SOMA,CONT_MAS, CONT_FEM,MEDIA;
```

```
  char NOME, SEXO;
  CONT_MAS = 0;
  CONT_FEM = 0;
  SOMA = 0;
  scanf("%c",&NOME);
  while (NOME!='@')
 {
     scanf("%s",&SEXO);
     scanf("%f",&PESO);
     scanf("%f",&ALTURA);
     scanf("%d",&IDADE);

   if (SEXO=='M')
   {
       CONT_MAS = CONT_MAS + 1;
       if ((ALTURA > ALTUMAS)||( CONT_MAS == 1 ))
      {
         ALTUMAS = ALTURA;
      }
   }
   if (SEXO=='F')
   {
        CONT_FEM = CONT_FEM + 1;
        if ((PESO > PESO_FEM) || (CONT_FEM == 1))
        {
            PESO_FEM = PESO;
        }
   }
  SOMA = SOMA + IDADE;
  scanf("%s",&NOME);
 }

MEDIA = SOMA / (CONT_MAS + CONT_FEM);
printf("\nA altura do mais alto atleta do sexo masculino é:%f ",ALTUMAS);
printf("\nO peso da atleta feminina mais pesada é:%f",PESO_FEM);
printf("\nA média de idade dos atletas é:%d",MEDIA);
}
```

Entradas:

Saídas:

Descreva o funcionamento do programa:

Tarefa 3: Analisando os exercícios abaixo, utilize as estruturas de comando vistas até o momento e construa o programa utilizando a linguagem de programação C.

a) Fazer um algoritmo que:

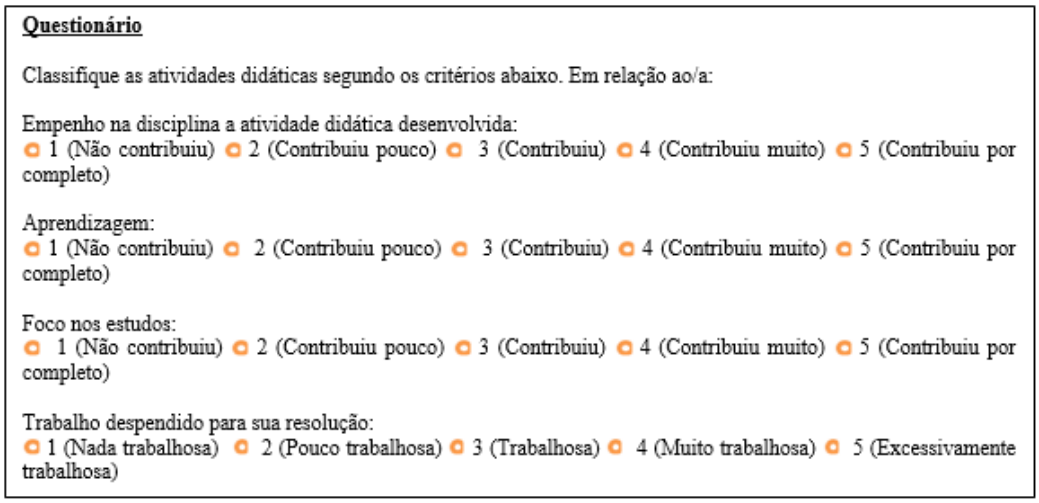
• Leia um número indeterminado de linhas contendo cada uma a idade de um indivíduo. A última linha, que não entrara nos cálculos, contem o valor da idade igual à zero.

• Calcule e escreva a idade média deste grupo de indivíduos.

Código na linguagem C:

b) Uma loja tem 15 clientes e deseja enviar uma correspondência a cada um deles anunciando um bônus especial. Faça um programa que leia o nome do cliente e o valor de suas compras no ano passado. Calcule e mostre um bônus de 10% se o valor das compras for menor de R$ 1.000,00 e de 15%, caso contrário.

Código na linguagem C:



Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Aprendizagem:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Foco nos estudos:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
○ 1 (Nada trabalhosa) ○ 2 (Pouco trabalhosa) ○ 3 (Trabalhosa) ○ 4 (Muito trabalhosa) ○ 5 (Excessivamente trabalhosa)

TURMA INFO - 1C

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: A primeira e segunda tarefa deve ser respondida na folha em que os exercícios foram apresentados. A terceira tarefa deve ser desenvolvida no computador e todas devem ser enviadas pelo ambiente de aprendizagem SIGAA.

Tarefa 1: Descreva o que os trechos de código abaixo estão executando (letra a e b) e suas respectivas saídas, após (letra c) desenvolva o algoritmos correspondente na linguagem c.

a)
```
int num=1;
    while(num>=1 && num<=4)
    {
```

```
      printf("\n O NUMERO DIGITADO ESTÁ CORRETO.");
      num=num+1;
      printf("\n\n O NUMERO DIGITADO E:% d",num);
      }
      printf("\n\n O NUMERO INCORRETO E:% d",num);
     }
```

b)
```
int i = 0;
 while(i < 10)
{
   i++;
   if(i %2 == 0)
      break;
   printf("O valor de i:%d",i);
}
```

c) Faça um programa em linguagem C que leia 10 números positivos e imprima o quadrado de cada número.

Tarefa 2**:** Analise os códigos abaixo e escreva suas respectivas saídas. Após, descreva o que cada programa está fazendo.

a)
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
main()
{
   int num, soma=0;
   float media=0, cont=0;
   printf("\n DIGITE UM NUMERO INTEIRO: ");
   scanf("%d",&num);
        if(num %3==0 && num!=0)
        {
             soma=soma+num;
             cont++;
        }
while(num!=0)
{
  printf("\n DIGITE UM NUMERO INTEIRO: ");
  scanf("%d",&num);
        if(num %3==0 && num!=0)
        {
            soma=soma+num;
            cont++;
        }
}
media=soma/cont;
printf("\n\n A media dos numeros e: %3.2f ",media);
printf("\n\n");
system("pause");
```

```
return(0);
}
```

Entradas:

DIGITE UM NÚMERO:

De acordo com os números recebidos atualizem os valores das variáveis

soma:
cont:

Saídas:

Descreva o funcionamento do programa:

b)

```
#include <stdio.h>
#include <conio.h>

int QTDPROD,QTDVEND,ESTOQUE,ESTOQUE_MAIOR;
char NOME,NOME_MAIOR;
main( )
{
   ESTOQUE_MAIOR = 0;
   printf("Entre com o Nome: ");
   scanf("%c",&NOME);
while ( NOME != '@' )
{
   printf("Quantidade Produzida: ");
   scanf("%d",&QTDPROD);
   printf("Quantidade Vendida: ");
   scanf("%d",&QTDVEND);
   ESTOQUE = QTDPROD - QTDVEND;
   printf("Nome: %c",NOME);
   printf("\nEstoque: %d",ESTOQUE);
         if ( ESTOQUE < 50 )
            printf("\nEstoque em baixa");
         if (ESTOQUE > ESTOQUE_MAIOR )
         {
            ESTOQUE_MAIOR = ESTOQUE;
            NOME_MAIOR = NOME;
         }
   printf("Nome: ");
   scanf("\n%c",&NOME);
}
printf("Produto: %c",NOME_MAIOR);
printf("Com maior estoque: %d",ESTOQUE_MAIOR);
}
```

De acordo com as entradas, vá colocando ao lado das saídas os valores correspondentes assumidos pelas variáveis NOME_MAIOR E ESTOQUE_MAIOR.

Entradas:
NOME: l, d, @
QTDPROD: 150, 200
QTDVEND: 50, 100

Saídas:

Descreva o funcionamento do programa:

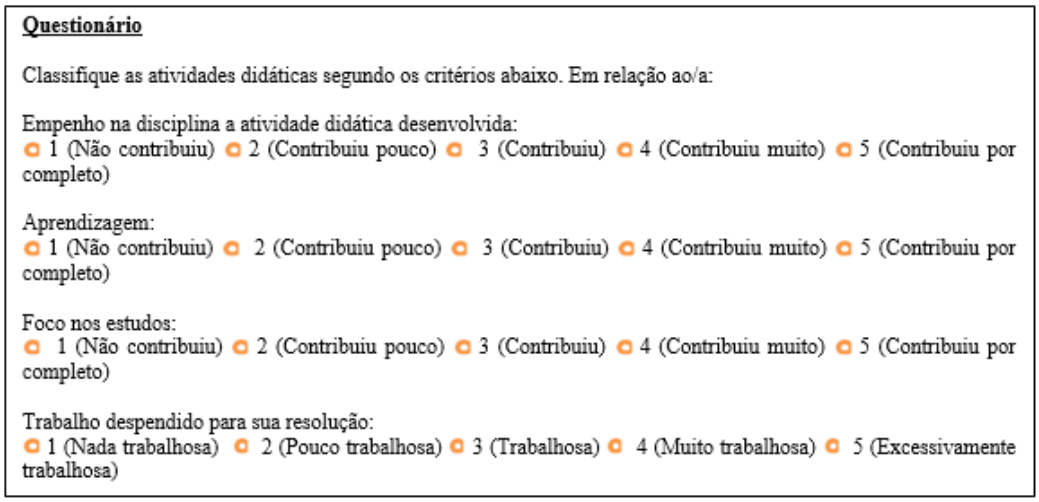
Tarefa 3: Analisando os exercício abaixo, utilize as estruturas de comando vistas até o momento e construa o programa utilizando a linguagem de programação C.

a) Uma loja está levantando o valor total de todas as mercadorias em estoque. Escreva um algoritmo que permita ler o valor da mercadoria e perguntar 'MAIS MERCADORIAS (S/N)?'. Ao final, imprimir o valor total em estoque e a média de valor das mercadorias em estoque.

Código na linguagem C:

b) Faça um programa que receba uma senha formada de quatro números inteiros, verifique se a senha está correta e, caso não esteja, solicite novamente a senha. Se a senha digitada for à correta, deverá ser apresentada a mensagem "Senha Correta", e finalizar o programa, caso contrário, "Senha Incorreta" e o usuário poderá tentar mais quatro vezes, após finalizar o programa.

Código na linguagem C:



Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Aprendizagem:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Foco nos estudos:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
○ 1 (Nada trabalhosa) ○ 2 (Pouco trabalhosa) ○ 3 (Trabalhosa) ○ 4 (Muito trabalhosa) ○ 5 (Excessivamente trabalhosa)

APÊNDICE F – ATIVIDADES DIDÁTICAS DO....WHILE

TURMA INFO 1A

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: Todas as tarefas devem ser implementadas no computador e enviadas pelo ambiente de aprendizagem SIGAA. O fluxograma da terceira tarefa deve ser desenvolvido na folha em que os exercícios foram apresentados e também ser enviado pelo sistema.

Tarefa 1: Analisando os exercícios abaixo, utilize as estruturas de comando adequadas para desenvolver a devida tarefa.

a) Faça um programa que, para um número indeterminado de pessoas leia a idade de cada uma, sendo que a idade 0 (zero) indica o fim da leitura e não deve ser considerada. A seguir calcule:
   • o número de pessoas;
   • a idade média do grupo;
   • a menor idade e a maior idade.
   Considere que o programa deve obrigatoriamente ler ao menos uma vez a idade.

Código na linguagem C:

b) Faça um algoritmo que calcule a média de salários de uma empresa, pedindo ao usuário o nome dos funcionários e os salários e devolvendo a média, o salário mais alto e o salário mais baixo. Use nome = "fim" para encerrar a leitura. Considere que ao menos seja realizada a leitura do salário de um funcionário.

Código na linguagem C:

Tarefa 2**:** Analise os códigos abaixo e reescreva-os utilizando o comando do...while.

a)

```
#include<stdio.h>
#include<stdlib.h>

main(){
int num, soma=0, cont=0;
float media;
printf ("\nDigite um numero inteiro. Digite zero para encerrar a execucao: ");
scanf ("%d", &num);
while (num!=0) {
```

```
        if (num%2==0){
        soma=soma+num;
        cont++;
        }
        printf ("\nDigite um numero inteiro. Digite zero para encerrar a execucao: ");
        scanf ("%d", &num);
        }
        if (cont>0){
        media=soma/cont;
        printf ("\nA media eh %.2f\n", media);
        }
        else
        printf ("\nNenhum valor foi digitado");
        system("pause");
}
```

Código do...while:

Explique a diferença entre os códigos:

Tarefa 3: Faça um fluxograma dos algoritmos abaixo. Após, transforme o pseudocódigo abaixo em um programa que deve ser desenvolvido utilizando a linguagem de programação C.

a)

```
algoritmo_notas
Var
nota1,nota2,media:real
resp:inteiro
inicio
repita
        escreva("Digite a primeira nota")
        leia(nota1)
        escreva("Digite a segunda nota")
        leia(nota2)
        media := (nota1+nota2)/2
        escreva(media)

escreva("Digite 1 para continuar ou 2 para sair")
leia(resp)
ate(resp=2)
fimalgoritmo
```

Código na linguagem C:

Fluxograma:

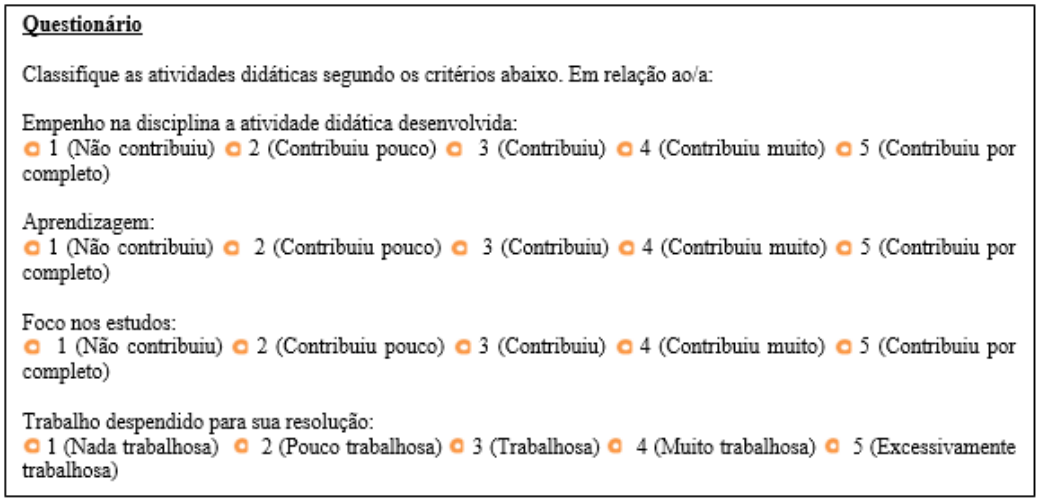
b)

```
var
n, qtde, soma:inteiro
```

```
inicio
qtde := 0
soma := 0
repita
escreva ("Informe um número inteiro: ")
leia (n)
se (n >= 0) e (n mod 2 = 1) entao
        soma := soma + n
        qtde := qtde + 1
fimse;
até (n < 0)
        escreva (qtde, soma)
Fimalgoritmo
```

Código na linguagem C:

Fluxograma:



Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Aprendizagem:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Foco nos estudos:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
○ 1 (Nada trabalhosa) ○ 2 (Pouco trabalhosa) ○ 3 (Trabalhosa) ○ 4 (Muito trabalhosa) ○ 5 (Excessivamente trabalhosa)

TURMA INFO 1B

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: Todas as tarefas devem ser implementadas no computador e enviadas pelo ambiente de aprendizagem SIGAA. O fluxograma da terceira tarefa deve ser desenvolvido na folha em que os exercícios foram apresentados e ser enviado pelo sistema.

Tarefa 1: Analisando os exercícios abaixo, utilize as estruturas de comando adequadas para desenvolver a devida tarefa.

a) Deseja-se fazer uma pesquisa a respeito do consumo mensal de energia elétrica em uma determinada cidade. Para isso, são fornecidos os seguintes dados:
   • Número do consumidor;
   • Preço do Kw consumido;
   • Quantidade de Kw consumido durante um mês;
   • Código do tipo de consumidor (residencial, industrial, comercial);
   O número do consumidor igual a zero deve ser usado para finalizar o algoritmo. Considere que ao menos seja realizada uma vez a leitura de todos os dados solicitados acima.

Código na linguagem C:

b) Desenvolver um programa que leia um número não determinado de valores, calcule e escreva a média aritmética deles, a quantidade de valores positivos, a quantidade de valores negativos e o percentual de valores negativos e positivos. Esse programa deve parar quando for digitada a letra 'z'. Os dados devem serem lidos ao menos uma vez.

Código na linguagem C:

Tarefa 2**:** Analise os códigos abaixo e reescreva-os utilizando o comando do..while. Após, explique a diferença entre os programas.

a)

```
#include <stdio.h>
#include <stdlib.h>

main() {
int num, soma=0;
float media=0, cont=0;
printf("\n DIGITE UM NUMERO INTEIRO: ");
scanf("%d",&num);

if(num %3==0 && num!=0)
{
soma=soma+num; cont++;
}
while(num!=0) {
printf("\n DIGITE UM NUMERO INTEIRO: ");
scanf("%d",&num);
if(num %3==0 && num!=0)
{
  soma=soma+num; cont++;
}}
media=soma/cont;
printf("\n\n A media dos numeros e: %3.2f ",media);
```

```
printf("\n\n");
system("pause");
return(0);
}
```

Código do...while:

Explique a diferença entre os códigos:

Tarefa 3: Faça um fluxograma dos algoritmos abaixo. Após, transforme o pseudocódigo abaixo em um programa que deve ser desenvolvido utilizando a linguagem de programação C.

a)

```
algoritmo_numeros
var
qtd,valor,maiorValor,menorValor,total:inteiro

inicio
qtd<-1
repita
        escreva("Informe o valor")
        leia(valor)
        se (qtd=1) entao
                maiorValor <-valor
                menorValor<-valor
                senao
                se (valor>maiorValor) então
                        maiorValor<- valor
                fimse
                    se (valor<menorValor) então
                        menorValor<-valor
                   fimse
        fimse
        qtd<-qtd+1
        total<-total+valor
        ate(qtd>10)
        escreva(“O maior valor e: ”,maiorValor)
        escreva(“O menor valor e: ”,menorValor)
        escreva(“A media dos valores e: ”,total/10)
fimalgoritmo
```

Fluxograma:

Código na linguagem C:

b)

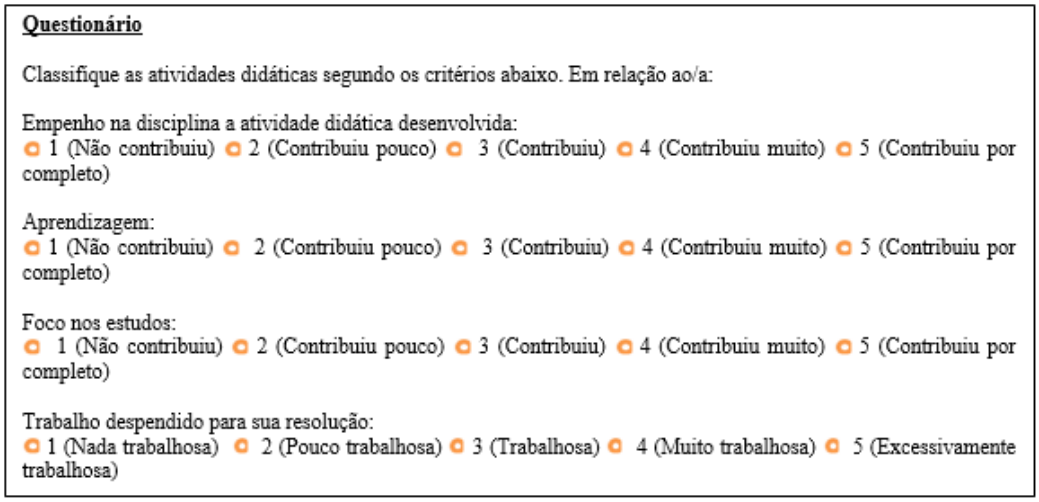
```
var
nome:caracter
valor,bônus:real
```

```
op:inteiro

inicio
op<-0
repita
escreval ("Informe o nome: ")
leia (nome)
escreva(“Informe o valor da compra”)
leia(valor)
op<-op+1
se (valor<=500) então
    bonus<-(valor*10)/100
     escreva(nome)
     escreva(bonus)
senao
     bônus<-(valor*15)/100
     escreva(nome)
     escreva(bonus)
fimse
até (op =3)
fimalgoritmo
```

Código na linguagem C:

Fluxograma:



Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Aprendizagem:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Foco nos estudos:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
o 1 (Nada trabalhosa) o 2 (Pouco trabalhosa) o 3 (Trabalhosa) o 4 (Muito trabalhosa) o 5 (Excessivamente trabalhosa)

TURMA INFO 1C

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: Todas as tarefas devem ser implementadas no computador e enviadas pelo ambiente de aprendizagem SIGAA. O fluxograma da terceira tarefa deve ser desenvolvido na folha em que os exercícios foram apresentados e também ser enviado pelo sistema.

Tarefa 1: Analisando os exercícios abaixo, utilize as estruturas de comando adequadas para desenvolver a devida tarefa.

a) Dado que cada pessoa tenha o seu nome, a sua idade e o seu peso em uma ficha, faça um algoritmo que:

a) Imprima o nome da pessoa cuja idade esta na faixa de 20 a 30 anos inclusive os extremos;

b) Calcule a idade média das pessoas com peso maior que 80 Kg, considere que exista N pessoas.

Para finalizar o algoritmo o usuário deve digitar 'F' quando entrar com o nome.

Código na linguagem C:

b) Uma empresa com X funcionários precisa saber a média de seus salários. Faça um algoritmo para ler a quantidade de funcionários e o salário de cada um e escrever a média dos salários. O programa encerra quando for digitada a letra 'p'. O programa deve permitir que o usuário entre pelo menos uma vez com os dados solicitados.

Código na linguagem C:

Tarefa 2**:** Analise os códigos abaixo e reescreva-os utilizando o comando do..while. Após, explique a diferença entre os programas.

a)

```
#include<stdio.h>
#include<stdlib.h>

main(){
int num,multiplo_10=0,multiplo_3=0,cont=0,produto_par=1;
float soma_impar=0;
while(cont<5)
{
cont =cont + 1;
printf ("entre com o numero: ");
scanf ("%d",&num);
```

```
if (num%2!=0) {

soma_impar= soma_impar + num;
}
if (num%2==0){

produto_par=produto_par*num;
}
if (num%10==0){

multiplo_10= multiplo_10 + 1;
}
if (num%3==0){

multiplo_3= multiplo_3 + 1;
}
}
printf ("\nsoma dos numeros impares:%f",soma_impar);
printf ("\nproduto dos números pares:%d",produto_par);
printf ("\nnumeros multiplos de dez:%d",multiplo_10);
printf ("\nnumeros multiplos de três:%d",multiplo_3);
}
```

Código do...while:

Explique a diferença entre os códigos:

Tarefa 3: Faça um fluxograma dos algoritmos abaixo. Após, transforme o pseudocódigo abaixo em um programa que deve ser desenvolvido utilizando a linguagem de programação C.

a)
```
var
num:inteiro
chute:inteiro
tent:inteiro
inicio
   tent<-0
   leia(num)
repita
   leia(chute)
   tent <- tent+1
         se(chute>num)entao
         escreva(“Chutou alto!”)
         senao
         se(chute<num)entao
         escreva(“Chutou baixo!”)
         fimse
         fimse
ate(num=chute)
escreva(“Número de tentativas:”,barraca)
```

fimalgoritmo

Fluxograma:

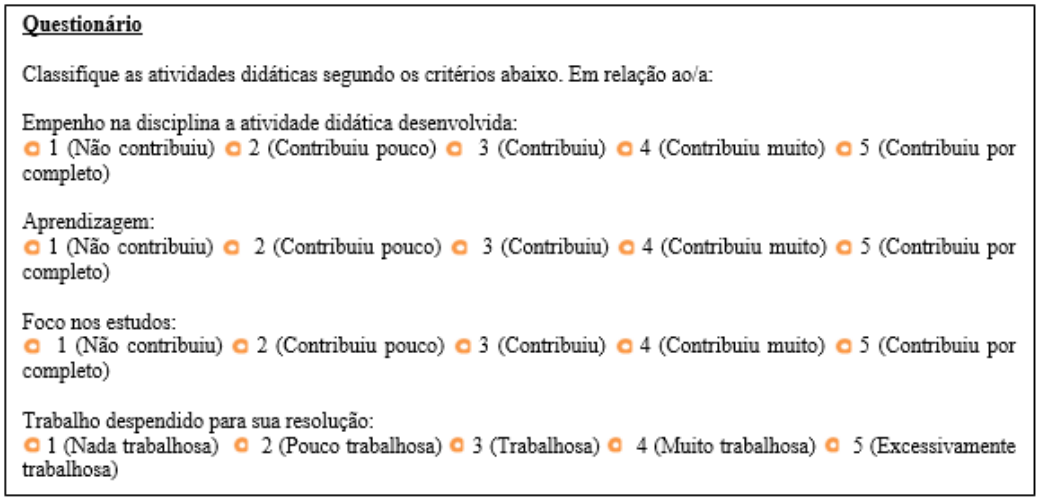
Código na linguagem C:

b)
var
n,con:inteiro

inicio
escreva(“Digite um número para uma tabuada ou (-1) para sair:”)
leia(n)
repita
con<-1
repita
escreva(com, “x”, n, “=”, con*n)
con<-con+1
ate(com>10)
escreva(“Digite outro número ou (-1) para sair:”)
leia(n)
ate(n = -1)
se (n=-1) então
escreva(“Até logo!”)
fimse
fimalgorirmo

Código na linguagem C:

Fluxograma:



Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
1 (Não contribuiu) 2 (Contribuiu pouco) 3 (Contribuiu) 4 (Contribuiu muito) 5 (Contribuiu por completo)

Aprendizagem:
1 (Não contribuiu) 2 (Contribuiu pouco) 3 (Contribuiu) 4 (Contribuiu muito) 5 (Contribuiu por completo)

Foco nos estudos:
1 (Não contribuiu) 2 (Contribuiu pouco) 3 (Contribuiu) 4 (Contribuiu muito) 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
1 (Nada trabalhosa) 2 (Pouco trabalhosa) 3 (Trabalhosa) 4 (Muito trabalhosa) 5 (Excessivamente trabalhosa)

APÊNDICE G – ATIVIDADES DIDÁTICAS FOR

TURMA INFO 1A

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: A segunda e terceira tarefa devem ser implementadas no computador e enviadas pelo ambiente de aprendizagem SIGAA. A primeira tarefa e a análise da terceira devem ser respondidas na folha em que os exercícios foram apresentados e enviada pelo sistema.

Tarefa 1: Descreva o que os trechos de código abaixo estão executando (letra a e b) e suas respectivas saídas, após (letra c) desenvolva o algoritmo correspondente na linguagem c.

a)

```
#include<stdio.h>
int main()
{
   int num,cont,resul;

   printf("\n Informe o número: \n");
   scanf("%d",&num);

   if(num>=1 && num<=10)
   {
      for(cont=1;cont<=10;cont++)
     {
         resul=(num*cont);
         printf("%d * %d =%d\n ",num,cont,resul);
     }
   }
   else
         printf("\n o número não está entre 1 e 10");
}
```

b)

```
#include<stdio.h>
int main()
{
int idades, soma_idades=0, qtde_idades=5, cont;
```

```
float media;

for(cont=1;cont<=qtde_idades;cont++)
   {
    printf("Entre com a idade %d : ",cont);
    scanf("%d", &idades);
    soma_idades = soma_idades + idades;
   }

media = soma_idades/qtde_idades;
printf ("\n A media de idades é de : %.1f", media);

}
```

c) Ler dez números e imprimir quantos são pares e quantos são ímpares. Explique sua resolução.

Tarefa 2: Analisar os códigos abaixo e codificá-los utilizando o comando de repetição for na linguagem de programação c.

a)

```
#include<stdio.h>
float r1,r2,r3,req;
int cont,n;
main()
{
  printf("Digite o número de repetições:");
  scanf("%d",&n);
  cont=0;
do
{
  printf("Digite o valor de r1: ");
  scanf("%f",&r1);
  printf("Digite o valor de r2: ");
  scanf("%f",&r2);
  printf("Digite o valor de r3: ");
  scanf("%f",&r3);
  req= 1/(1/r1+1/r2+1/r3);
  printf("%f",req);
  cont=cont+1;
}while(cont<n);
}
```

Código com laço de repetição For:

b)

```
#include<stdio.h>
main()
{
```

```
int n,c;
float x,y;

printf("Entre com n:");
scanf("%d",&n);
if(n<=0)
{
    printf("Valor de n e inválido");
}
else
{
x=2;
c=1;
}
while(c<=n)
{
    y=(x*x)+(2*x)-5;
    printf("Valor de x:%f",x);
    printf("Valor de y:%f",y);
    x=x+2;
    c=c+1;
}
}
```

Código com laço de repetição For:

Tarefa 3**:** Desenvolva os códigos abaixo e implemente-os utilizando os três comandos de repetição vistos em sala de aula. Essa tarefa irá gerar três programas. Após, descreva a diferença entre eles.

a) Dado o valor de N, calcular e exibir os N primeiros múltiplos de 11 superiores a zero.

Programas em C:

Análise do aluno:

b) Entre com o valor de N, calcule e exiba o valor da soma das N primeiras parcelas:
1+3+5+7+ .....

Programas em C:

Análise do aluno:

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Aprendizagem:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Foco nos estudos:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
o 1 (Nada trabalhosa) o 2 (Pouco trabalhosa) o 3 (Trabalhosa) o 4 (Muito trabalhosa) o 5 (Excessivamente trabalhosa)

TURMA INFO 1B

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: A segunda e terceira tarefa devem ser implementadas no computador e enviadas pelo ambiente de aprendizagem SIGAA. A primeira tarefa e a análise da terceira devem ser respondidas na folha em que os exercícios foram apresentados e também serem enviadas pelo sistema.

Tarefa 1: Descreva o que os trechos de código abaixo estão executando (letra a e b) e suas respectivas saídas (caso necessário simule as entradas), após (letra c) desenvolva o algoritmo correspondente na linguagem c.

a)

```
#include<stdio.h>

int main()
{
 int num;
   for(num=1;num<100;num++)
  {
       if(num%2==0)
        {
          printf("%d ",num);
         }
  }}
```

b)

```
#include<stdio.h>
int main()
 {
int n=0, num, i;
  for (i=1; i<=10; i++)
 {

     printf("Informe número: ");
     scanf("%d", &num);
     if (num>0)
       {
        n=n+1;
        }
  }

printf("\n: %d\n", n);
printf(": %d", 10-n);
printf("\n");
}
```

c) Faça um programa que imprima os múltiplos de 5, no intervalo de 1 até 500. Explique sua resolução.

Tarefa 2: Analisar os códigos abaixo e codificá-los utilizando o comando de repetição for na linguagem de programação c.

a)

```
#include<stdio.h>
#include <conio.h>
main()
{ int i, ano_atual;
  float salario, novo_salario, percentual;

  printf("Digite o ano atual: ");
  scanf("%d",&ano_atual);
  salario = 1000;
  percentual = 1.5/100;
  novo_salario = salario + percentual * salario;
  i = 2015;
          do
                  {
                          percentual = 2 * percentual;
                          novo_salario = novo_salario + percentual * novo_salario;
                          i = i + 1;
                  }       while (i <= ano_atual);

           printf("\nNovo salario = %f", novo_salario);
```

```
        getch();
}
```

Código com laço de repetição For:

b)

```
#include<stdio.h>
main()
{
int CODIGO;
int PRECO, TOTAL;
TOTAL=0;
printf("Digite o codigo do produto ou 123 para finalizar :");
scanf ("%d",&CODIGO);
while (CODIGO != 123)
{   printf ("Digite o preco :");
   scanf ("%d",&PRECO);
   TOTAL = (TOTAL + PRECO);
   printf ("Digite o código ou 123 para encerrar :");
   scanf ("%d",&CODIGO);
}
printf ("O total da compra e %d:", TOTAL);
}
```

Código com laço de repetição For:

Tarefa 3**:** Desenvolva os códigos abaixo e implemente-os utilizando os três comandos de repetição vistos em sala de aula. Essa tarefa irá gerar três programas. Após, descreva a diferença entre eles.

a) Num frigorífico existem 90 ovelhas. Cada ovelha traz preso em seu pescoço um cartão contendo seu número de identificação e seu peso. Faça um programa que imprima a identificação e o peso da ovelha mais gorda e da ovelha mais magra (supondo que não haja empates).

Programas em C:

Análise do aluno:

b) Criar um algoritmo na linguagem C que imprima todos os números de 1 até 100, inclusive, e a média de todos eles.

Programas em C:

Análise do aluno:

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Aprendizagem:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Foco nos estudos:
o 1 (Não contribuiu) o 2 (Contribuiu pouco) o 3 (Contribuiu) o 4 (Contribuiu muito) o 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
o 1 (Nada trabalhosa) o 2 (Pouco trabalhosa) o 3 (Trabalhosa) o 4 (Muito trabalhosa) o 5 (Excessivamente trabalhosa)

TURMA INFO 1C

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: A segunda e terceira tarefa devem ser implementadas no computador e enviadas pelo ambiente de aprendizagem SIGAA. A primeira tarefa e a análise da terceira devem ser respondidas na folha em que os exercícios foram apresentados e também enviada pelo sistema.

Tarefa 1: Descreva o que os trechos de código abaixo estão executando (letra a e b) e suas respectivas saídas (caso necessário simule as entradas), após (letra c) desenvolva o algoritmo correspondente na linguagem c.

a)

```
#include<stdio.h>
#include <conio.h>
main()
{
int n,i;
int soma = 0;
printf("Digite um numero: ");
scanf("%d", &n);
for(i=0;i<5;i++)
{
if( n >= 0)
soma = soma + n;
else
break;
```

```
printf("Digite um numero positivo para ser somado ou negativo para sair: ");
scanf("%d", &n);
}
printf("A soma eh %d\n", soma);
}
```

b)

```
#include<stdio.h>
#include <conio.h>
 main()
{
    int cont, num1, num2, res;
    num1 = 0;
    num2 = 1;
    printf("\n%d",num1);
    printf("\n%d",num2);
  for (cont=3;cont<=8;cont++)
    {
          res = num1 + num2;
          printf ("\n%d",res) ;
          num1 = num2;
          num2 = res;
    }
 getch();
}
```

c) Faça um algoritmo que leia 10 números inteiros e imprima o menor, o maior e a soma deles. Explique sua resolução.

Tarefa 2: Analisar os códigos abaixo e codificá-los utilizando o comando de repetição for na linguagem de programação c.

a)

```
#include<stdio.h>
```

```
#include<stdio.h>
main(){
 float divida, juros, aplicacao, rendimento;
 int meses;
 meses = 0;
 divida = 10000;
 juros = 0.025;
 aplicacao = 1500;
 rendimento = 0.04;

do
{
   divida = (divida * juros) + divida;
```

```
  aplicacao = (aplicacao * rendimento) + aplicacao;
  meses ++;
 }while (divida > aplicacao);

 printf("Quantidade de meses para liquidar a divida e %d\n",meses);
 printf("Sua divida esta em %f\n",divida);
 printf("Sua aplicacao esta em %f\n", aplicacao);
}
```

Código com laço de repetição For:

b)

```
#include<stdio.h>
main()
{
  int cont, ta, te, tr;
  float n1, n2, media, media_classe, total_classe;
  tr = 0;
  te = 0;
  ta = 0;
  total_classe = 0;
  cont = 1;
 while (cont <= 6)
   {
        printf( "\nDigite as duas notas do aluno:”);
        scanf(“%d”,&n1);
        scanf(“%d”,&n1);
        media = (n1 + n2) /2;
        printf(“\n A media do aluno%f”,media);
    if (media <= 3)
    {
        tr = tr + 1;
        printf(" Reprovado");
     }
    if (media > 3 && media < 7)
   {
        te = te + 1;
        printf(" Exame");
    }
    if (media >= 7)
    {
         ta = ta + 1;
        printf("Aprovado");
     }
        total_classe = total_classe + media;
        cont = cont + 1;
      }
        printf("\nTotal de reprovados = %d",tr);
        printf("\nTotal de alunos em exame = %d",te);
```

```
        printf("\nTotal de aprovados = %d",ta);
        media_classe = total_classe/6;
        printf("\nMedia da classe = %d", media_classe);
        getch();
}
```

Código com laço de repetição For:

Tarefa 3**:** Desenvolva os códigos abaixo e implemente-os utilizando os três comandos de repetição vistos em sala de aula. Essa tarefa irá gerar três programas. Após, descreva a diferença entre eles.

a) Construa um programa em C que leia vários números e informe quantos números entre 100 e 200 foram digitados. Quando o valor 0 (zero) for lido, o algoritmo deverá cessar sua execução.

Programas em C:

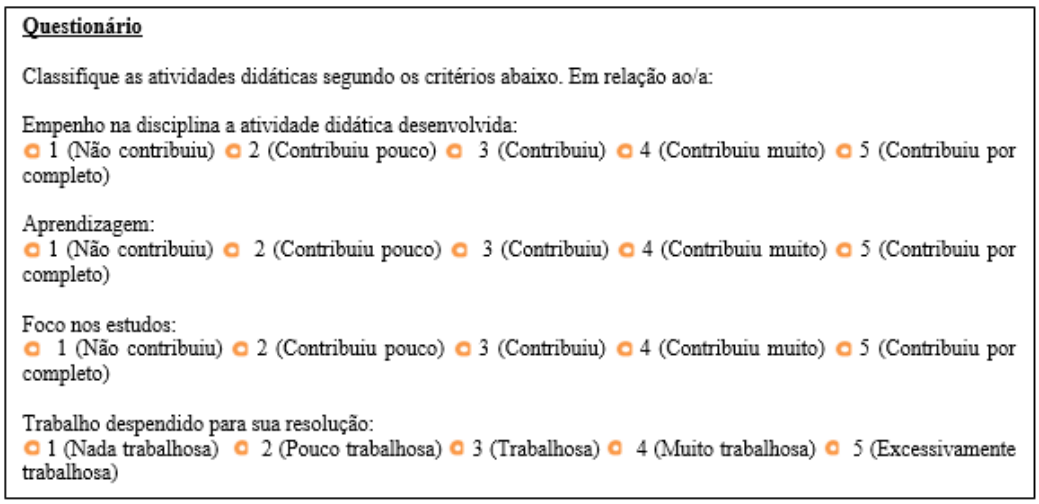
Análise do aluno:

b) Faça um programa em C que permita entrar com a idade de várias pessoas e imprima:

- total de pessoas com menos de 21 anos
- total de pessoas com mais de 50 anos.

Programas em C:

Análise do aluno:



Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Aprendizagem:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Foco nos estudos:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
○ 1 (Nada trabalhosa) ○ 2 (Pouco trabalhosa) ○ 3 (Trabalhosa) ○ 4 (Muito trabalhosa) ○ 5 (Excessivamente trabalhosa)

APÊNDICE H – ATIVIDADES DIDÁTICAS VETOR

TURMA INFO 1A

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: A primeira tarefa deve ser desenvolvida no laboratório de informática e enviada pelo sistema de aprendizagem SIGAA. A segunda tarefa deve ser respondida na folha em que os exercícios foram apresentados e também ser enviada pelo sistema.

Tarefa 1: Desenvolva os programas abaixo de acordo com o que é solicitado. Os códigos devem ser desenvolvidos utilizando a linguagem de programação c.

a) Preencher um vetor com números inteiros (8unidades); solicitar um número do teclado. Pesquisar se esse número existe no vetor. Se existir, imprimir em qual posição do vetor. Se não existir, imprimir mensagem que não existe.

Código em C:

b) Neste exercício temos dois vetores com 5 posições (0 a 4). Em cada vetor entraremos cinco números. Mostrar os números e depois somar números que pertençam a mesma posição ou seja: [0]+[0],[1]+[1],...

Código em C:

Tarefa 2: Analise os códigos abaixo, descreva o que o programa faz e também identifique as saídas do programa conforme solicitado. No primeiro programa o aluno deve escolher as entradas para que possa descrever as saídas correspondentes.

a)

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
main()
{
float vetp[5], vetc[5];
int x;
printf ("\n");
for(x=1; x<=5; x++)
{
```

```
printf ("\tDigite um numero: ");
scanf("%f",&vetp[x]);
vetc[x]=pow(vetp[x],3);
printf (" %3.0f",vetc[x]);
printf ("\n");
}
printf("\n");
system("pause");
return(0);
}
```

Entradas:

Saídas:

Análise do aluno:

b)

```
#include <stdio.h>
int main()
{
int v[10]={10,9,10,7,3,20,10,2,1,10};
int i,maior=0,igual=0,menor=0,soma1=0,soma2=0,soma=0;
for(i=0;i<10;i++)
{
if(v[i]>v[0])
{
maior=maior+1;
soma=soma+v[i];
}

if(v[i+1]==v[0])
{
igual=igual+1;
soma1=soma1+v[i];
}
if(v[i]<v[0])
{
menor=menor+1;
soma2=soma2+v[i];
} }
printf("\n %d números são maiores que %d %d",maior,v[0],soma);
printf("\n %d números são menores que %d %d",menor,v[0],soma2);
printf("\n números iguais a %d = %d %d",v[0],igual,soma1);
return 0;
}
```

Entradas:

Saídas:

Análise do aluno:

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Aprendizagem:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Foco nos estudos:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
○ 1 (Nada trabalhosa) ○ 2 (Pouco trabalhosa) ○ 3 (Trabalhosa) ○ 4 (Muito trabalhosa) ○ 5 (Excessivamente trabalhosa)

TURMA INFO 1B

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: A primeira tarefa deve ser desenvolvida no laboratório de informática e enviada pelo ambiente de aprendizagem SIGAA. A segunda tarefa deve ser respondida na folha em que os exercícios foram apresentados e também ser enviada pelo sistema.

Tarefa 1: Desenvolva os programas abaixo de acordo com o que é solicitado. Os códigos devem ser desenvolvidos utilizando a linguagem de programação c.

a) Dado um vertor X de n elementos faça um algoritmo que:

a) Crie outro vetor Y contendo os elementos de x que estão na faixa entre 10 e 40;
b) Crie outro vetor W contendo os números que estão nas posições pares;
c) Pesquise a existência de um determinado elemento Y no vetor X;
d) Escreva o menor e maior elemento do vetor X;

Código em C:

b) Preencher um vetor com os números pares do número 2 a 20. Preencher outro vetor com os números de 10 a 19. Ao final os valores dos dois vetores devem ser somados e o resultado mostrado ao usuário.

Código em C:

Tarefa 2: Analise os códigos abaixo, descreva o que o programa faz e também identifique as saídas do programa conforme solicitado.

a)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
main() {
int vet1[8], x, cont=0, soma=0, soma2=0;
for(x=0;x<=7;x++) {
printf("\nDigite um valor: ");
scanf("%d",&vet1[x]);
if(vet1[x]>30) {
cont++; soma=soma+vet1[x];
} }
for(x=0;x<=7;x++)
printf("\t%d",vet1[x]);
printf("\n\n %d Numeros sao maiores que 30",cont);
printf("\n\n A Soma dos numeros maiores que 30 e = %d",soma);
for(x=0;x<=7;x++)
soma2=soma2+vet1[x];
printf("\n\n A Soma dos numeros digitados e = %d",soma2);
printf("\n\n");
system("pause");
return(0);
}
```

Entradas:

Saídas:

Análise do aluno:

b)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
main( )
{
int vet1[8], x, cont=0, m5=0, NF=0, MN=0;
float soma=0;
for(x=0;x<=7;x++) {
printf("Informe um numero %d: ", x+1);
scanf("%d",&vet1[x]);
printf("\n");
}
 printf("\n\n");
 for(x=0;x<=7;x++)
 printf("\t%d",vet1[x]);
 printf("\n\n");
```

```
for(x=0;x<=7;x++) {
soma=soma+vet1[x];
if(vet1[x]%5==0)
m5++;
if(vet1[x]>10 && vet1[x]<30)
NF++;
if(vet1[x]>MN)
MN=vet1[x];
}
printf(" _____");
printf("\n| A media do vetor e: %3.2f |",soma/8);
printf("\n| Multiplos de 5: %d |",m5);
printf("\n| Entre 10 e 30: %d |",NF);
printf("\n| Maior numero: %d |",MN);
printf("\n|_____|");
printf("\n\n"); system("pause");
return(0);
}
```

Entradas:

Saídas:

Análise do aluno:

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Aprendizagem:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Foco nos estudos:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
○ 1 (Nada trabalhosa) ○ 2 (Pouco trabalhosa) ○ 3 (Trabalhosa) ○ 4 (Muito trabalhosa) ○ 5 (Excessivamente trabalhosa)

TURMA INFO 1C

Nome:_____

Data: ___/___/_____

Feedback qualificado:_____

Atividade Desenvolvida: ☐ Completa ☐ Parcial ☐ Não realizada

Orientações: A primeira tarefa deve ser desenvolvida no computador e enviada pelo sistema de aprendizagem SIGAA. A segunda tarefa deve ser respondida na folha em que os exercícios foram apresentados e também ser enviada pelo sistema.

Tarefa 1: Desenvolva os programas abaixo de acordo com o que é solicitado. Os códigos devem ser desenvolvidos utilizando a linguagem de programação c.

a) Leia 10 números inteiros e armazene em um vetor v. Crie dois novos vetores v1 e v2. Copie os valores ímpares de v para v1, e os valores pares de v para v2. Note que cada um dos vetores v1 e v2 têm no máximo 10 elementos, mas nem todos os elementos são utilizados. No final escreva quantas posições foram preenchidas em cada vetor v1 e v2.

Código em C:

b) Escrever um programa que declare um vetor de 20 inteiros, leia um valor para cada posição e no final mostre quantos elementos possuem valor maior, menor e igual ao primeiro elemento do vetor.

Código em C:

Tarefa 2: Analise os códigos abaixo, descreva o que o programa faz e também identifique as saídas do programa conforme solicitado.

a)

```
#include <stdio.h>
#include <stdlib.h>

int main() {
   int vet[7], i, acimaMedia = 0, soma = 0;
   float media;
   for (i = 0; i < 7; i++) {
      vet[i] = rand()%51;
   }

   for (i = 0; i < 7; i++) {
      soma = soma + vet[i];
   }
   media = soma /7;
   for (i = 0; i < 7; i++) {
      if (vet[i] > media) {
         acimaMedia++;
      }
   }
   printf("Vetor: ");
   for (i = 0; i < 7; i++) {
      printf("%i ", vet[i]);
   }
```

```
    printf("\nMedia: %f", media);
    printf("\nAcima da media: %i", acimaMedia);
  return 0;
}
```

Entradas:

Saídas:

Análise do aluno:

b)

```
#include <stdio.h>

int main() {
    int vet[5], i, maior, posMaior, menor, posMenor;

    for (i = 0; i < 5; i++) {
        scanf("%i", &vet[i]);
    }
    maior    = vet[0];
    posMaior = 0;
    for (i = 1; i < 5; i++) {
        if (vet[i] > maior) {
            maior    = vet[i];
            posMaior = i;
        }
    }
    menor    = vet[0];
    posMenor = 0;

    for (i = 1; i < 5; i++) {
        if (vet[i] < menor) {
            menor    = vet[i];
            posMenor = i;
        }
    }

    printf("Vetor: ");
    for (i = 0; i < TAM; i++) {
        printf("%i ", vet[i]);
    }
    printf("\nMaior valor: %i - posicao: %i", maior, posMaior+1);
    printf("\nMenor valor: %i - posicao: %i", menor, posMenor+1);
    return 0;
}
```

Entradas:
Saídas:
Análise do aluno:

Questionário

Classifique as atividades didáticas segundo os critérios abaixo. Em relação ao/a:

Empenho na disciplina a atividade didática desenvolvida:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Aprendizagem:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Foco nos estudos:
○ 1 (Não contribuiu) ○ 2 (Contribuiu pouco) ○ 3 (Contribuiu) ○ 4 (Contribuiu muito) ○ 5 (Contribuiu por completo)

Trabalho despendido para sua resolução:
○ 1 (Nada trabalhosa) ○ 2 (Pouco trabalhosa) ○ 3 (Trabalhosa) ○ 4 (Muito trabalhosa) ○ 5 (Excessivamente trabalhosa)

APÊNDICE I – ATIVIDADE 1 – CONCEITOS INICIAIS DE PROGRAMAÇÃO - FLUXOGRAMAS

Introdução

Para desenvolver essa atividade é preciso que o aluno tenha compreendido o conceito, o uso e a interpretação de fluxogramas. Essa forma de representação de algoritmos foi apresentada após o aluno ter tido contato com o conteúdo introdutório de algoritmo (o que é um algoritmo, formas de representação, identificadores, constantes, variáveis e seus tipos, operadores [aritméticos, relacionais e lógicos] e suas precedências). Também foi desenvolvida junto aos alunos uma lista de problemas, na qual o conteúdo de fluxogramas é abordado. Na sequência, a atividade didática apresentada será realizada.

Objetivo da atividade didática 1

Cada atividade didática elaborada possui objetivos tanto para os alunos como o professor.

Aluno

- Interpretar, reconhecer, analisar e criar fluxogramas.
- Ditar ritmo de estudo.

Professor

-Analisar os objetivos elencados em cada tarefa a fim de elaborar as matrizes de análise que servirão de auxílio para o professor acompanhar o aprendizado dos alunos durante o ano letivo.

Tarefa 1

A fim de desenvolver esta tarefa, em que os fluxogramas já estão prontos, ao aluno compete identificar as saídas, de acordo com os valores que são atribuídos às variáveis, os símbolos e os operadores lógicos.

Objetivos a serem verificados pelo professor ao final da tarefa:

-Analisar as saídas identificadas pelo aluno.

-Verificar a interpretação do aluno referente ao funcionamento do fluxograma, analisando o significado de cada símbolo.

-Identificar todas as possíveis saídas que podem ser visualizadas pelo usuário após percorrer caminhos diferente no fluxograma.

-Verificar se o aluno consegue simular entradas específicas para que certa informação seja obtida.

-Identificar a presença e a respectiva forma de uso dos operadores lógicos.

Tarefa 2

Para desenvolver esta tarefa, será necessário que o aluno interprete corretamente o enunciado do problema proposto e construa fluxogramas, a fim de que o desenvolvimento e o aprendizado do aluno sejam avaliados.

Objetivos a serem verificados pelo professor ao final da tarefa:

-Analisar a interpretação do aluno referente ao que foi solicitado no enunciado.

-Analisar a organização do raciocínio lógico (encadeamento das informações).

-Verificar a compreensão e interação das etapas algorítmicas.

- Analisar o aprendizado do aluno referente ao comando de decisão.

- Analisar o uso de operadores lógicos na construção do algoritmo.

APÊNDICE J - ATIVIDADE 2 – COMANDO IF...ELSE - LINGUAGEM C

Introdução

O propósito dessa atividade é verificar se o aluno consegue ler fluxogramas e codificá-los na linguagem de programação C. Também irá avaliar a interpretação no momento em que o aluno tiver que ler os códigos e analisar o que foi implementado. Para realizar essa atividade, o aluno já teve contato com conceitos referentes à linguagem de programação C (entradas, saídas, comentários, definições de pré-processamento, definições de tipo, variáveis, operador de endereço e estrutura de decisão). Para melhor entendimento do discente, todos esses comandos foram sendo apresentados com pequenos exemplos desenvolvidos na ferramenta DEV-C++ em sala de aula. Posteriormente, o aluno realizou tarefas em que se enfatiza tanto o raciocínio lógico quanto a escrita de programas na linguagem C. Depois de finalizar as etapas supracitadas, a atividade didática abaixo foi desenvolvida em sala de aula.

Objetivo da atividade didática 2

Cada atividade didática elaborada possui objetivos tanto para os alunos como o professor.

Aluno

- Desenvolver habilidades referente a conceitos e procedimentos de linguagem de programação.

- Ditar ritmo de estudo.

Professor

-Analisar os objetivos elencados em cada tarefa a fim de elaborar as matrizes de análise que servirão de auxílio para o professor acompanhar o aprendizado dos alunos durante o ano letivo.

Tarefa 1

Esta tarefa é composta por trechos de códigos desenvolvidos na linguagem C, em que o aluno deve identificar cada comando exposto a ele. Também será necessário desenvolver um

programa com comandos básicos da linguagem estudada a fim de analisar possíveis erros semânticos[16] e sintáticos[17].

Objetivos a serem verificados pelo professor ao final da tarefa:

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.
- Identificar a existência de dados inconsistentes.
- Analisar a interpretação do aluno referente ao solicitado.
- Analisar a estrutura utilizada no desenvolvimento da tarefa.

Tarefa 2

Nessa tarefa compete aos alunos identificar os conceitos de programação e da linguagem C (vistos em sala de aula), interpretar os códigos e verificar a saída correta que o programa irá apresentar.

Objetivos a serem verificados pelo professor ao final da tarefa:

- Analisar o código para detectar a existência de erros lógicos.
- Identificar a interpretação do aluno referente ao que foi exposto para ele.

Tarefa 3

Esta tarefa é composta por fluxogramas que precisam ser interpretados pelos alunos e codificados na linguagem de programação C.

Objetivos a serem verificados pelo professor ao final da tarefa:

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.
- Analisar a interpretação do aluno referente ao que foi exposto para ele (leitura correta do fluxograma).

[16] Erro semântico se refere a um erro na "lógica de seu código". O código compila, mas não faz o que foi solicitado. Este tipo de erro é geralmente mais difícil de ser identificado e corrigido.

[17] Erro sintático se refere ao erro de escrita da linguagem C, por exemplo, usar letra maiúscula inicial para o comando de seleção (o sistema não reconhecerá como "comando de seleção", tentará como uma "atribuição", mas como "atribuição" também não estará correta).

APÊNDICE K - ATIVIDADE 3 – COMANDO SWITCH....CASE

Introdução

Esta atividade tem como finalidade verificar a escolha do aluno em utilizar o comando *if...else* ou *switch...case* no desenvolvimento da tarefa apresentada a ele. Também será verificada a capacidade do aluno em modificar da estrutura *if...else* para a *switch..case* sem alterar o propósito do programa. Antes de desenvolver essa atividade, o aluno estudou o comando *switch...case*, o qual foi desenvolvido logo após o comando *if...else*. Ambos os comandos são pertencentes a estrutura de decisão e semelhantes na forma de implementação, sendo importante ao aluno visualizar formas diferenciadas de programar essas estruturas. Para melhor entendimento do discente, o comando foi apresentado com pequenos exemplos desenvolvidos em conjunto com os estudantes na ferramenta DEV-C++. Posteriormente, foram disponibilizados aos alunos problemas para que pudessem desenvolver tanto o raciocínio lógico quanto a escrita de programas na linguagem C. Depois de finalizar as etapas supracitadas, a atividade didática foi desenvolvida.

Objetivo da atividade didática 3

Cada atividade didática elaborada possui objetivos tanto para os alunos como o professor.

Aluno

- Analisar o uso dos comandos *if...else* e *switch...case* em diferentes contextos.
- Ditar ritmo de estudo.

Professor

-Analisar os objetivos elencados em cada tarefa a fim de elaborar as matrizes de análise que servirão de auxílio para o professor acompanhar o aprendizado dos alunos durante o ano letivo.

Tarefa 1

Esta tarefa será desenvolvida na linguagem de programação C, em que, os alunos devem identificar qual a estrutura ministrada até o momento (*if...else* ou *switch...case*) é a melhor opção para desenvolver o problema proposto, interpretar corretamente o que foi solicitado na tarefa e saber realizar os cálculos matemáticos necessários para o desenvolvimento da atividade.

Objetivos a serem verificados pelo professor ao final da tarefa:

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.
- Analisar a interpretação do aluno referente ao que foi exposto para ele (leitura correta do enunciado programa).
- Identificar se o discente consegue perceber os comandos necessários a serem utilizados no desenvolvimento do programa, utilizando, para isso, a linguagem de programação C.
- Analisar se o aluno consegue desenvolver os cálculos necessários para fazer com que o programa desempenhe sua função corretamente.
-Analisar a escolha feita pelo usuário da estrutura utilizada para desenvolver o programa.

Tarefa 2

Para desenvolver esta tarefa o aluno deve utilizar os conceitos de programação vistos em sala de aula, a fim de interpretar os códigos e reorganizá-los, utilizando o comando *switch..case*.

Objetivos a serem verificados pelo professor ao final da tarefa:

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.
- Analisar a interpretação do aluno referente ao que foi exposto para ele (leitura correta do programa).
-Verificar se o aluno é capaz de codificar programas desenvolvidos com o comando *if...else* aninhados em programas que utilizam o comando *switch...case*, visto que, muitas vezes, o aluno, por já ter assimilado o comando *if...else*, pode manifestar desinteresse pelo uso do comando *switch..case*.

Tarefa 3

Nesta tarefa, o aluno deverá ser capaz de interpretar pseudocódigos e conseguir codifica-lo facilmente na linguagem de programação C. O pseudocódigo foi apresentado em sala de aula apenas como uma forma de representação de algoritmos, mas não foi desenvolvida nenhuma atividade utilizando essa forma de representação.

Objetivos a serem verificados pelo professor ao final da tarefa:

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.

- Analisar a interpretação do aluno referente ao que foi exposto para ele (leitura correta do programa).
-Analisar se o aluno é capaz de ler programas em pseudocódigo e codificá-los na linguagem C, sem ter a necessidade de iniciar o conteúdo de programação por meio dessa abordagem.

APÊNDICE L - ATIVIDADE 4 – COMANDO WHILE

Introdução

Essa atividade foi elaborada com o objetivo de analisar a compreensão do aluno em utilizar, em seu código, o laço de repetição *while* contado ou condicional, por meio de tarefas que irão permitir que o aluno desenvolva e interprete programas que contenham essa característica. Esse comando é apresentado aos alunos na sequência do comando *switch..case*. Para melhor entendimento do discente, o comando foi apresentado com pequenos exemplos desenvolvidos em conjunto com os estudantes na ferramenta DEV-C++. Posteriormente, foram disponibilizados aos alunos problemas para que pudessem desenvolver tanto o raciocínio lógico quanto a escrita de programas na linguagem C. Durante o desenvolvimento dessa atividade, os alunos são acompanhados pelo professor, que os auxilia em seu desenvolvimento. Depois de finalizar as etapas anteriormente citadas, a atividade didática abaixo foi desenvolvida com os alunos.

Objetivo da atividade didática 4

Cada atividade didática elaborada possui objetivos tanto para os alunos como o professor.

Aluno

-Analisar o uso e aplicação do comando *while* em variados contextos.

- Ditar ritmo de estudo.

Professor

-Analisar os objetivos elencados em cada tarefa a fim de elaborar as matrizes de análise que servirão de auxílio para o professor acompanhar o aprendizado dos alunos durante o ano letivo.

Tarefa 1

Para desenvolver esta tarefa, o aluno terá que identificar o modo de funcionamento do comando *while*, sua condição de parada e seu incremento. Também será necessário desenvolver um programa utilizando a linguagem de programação C. Ressalta-se que os outros comandos continuam a ser utilizados no desenvolvimento dos códigos, visto que os conteúdos são ministrados de forma cumulativa.

Objetivos a serem verificados pelo professor ao final da tarefa:

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.

- Analisar a interpretação do aluno referente ao que foi exposto (leitura correta do programa).

- Identificar a percepção do discente sobre os comandos necessários de serem utilizados no desenvolvimento do programa, como as entradas, saídas, tipos de variáveis e as estruturas já trabalhadas em sala de aula.

- Analisar a descrição do aluno com relação ao entendimento do comando *while*, se consegue ou não identificar o seu funcionamento.

-Verificar o uso dos comandos abordados até o momento e suas devidas aplicações dentro dos códigos.

- Analisar se o aluno consegue utilizar os comandos contador e acumulador de forma correta.

Tarefa 2

A fim de realizar esta atividade, os alunos devem utilizar os conceitos de programação e a linguagem C, vistos em sala de aula. Os discentes deverão analisar os códigos e escrever as saídas conforme solicitado. Também será necessário que o aluno consiga identificar o funcionamento do comando *while*, contador e acumulador, os quais causam muita confusão ao discente no momento de sua aplicação.

Objetivos a serem verificados pelo professor ao final da tarefa:

- Analisar a interpretação do aluno referente ao que foi exposto para ele (leitura correta do programa).

-Verificar se o aluno é capaz de analisar corretamente o fluxo que o programa irá percorrer com as entradas estipuladas, bem como a forma de compreensão dos comandos dados até o momento.

-Analisar se o aluno observa que não há alteração nos valores das variáveis soma e total quando não for possível entrar no laço de repetição.

- Verificar se o aluno compreendeu o funcionamento dos comandos contador e acumulador.

Tarefa 3

Para desenvolver esta tarefa, o aluno terá que utilizar o comando de repetição *while* e identificar a necessidade em aplicar o *while* contado ou condicional. Também será analisada a

interpretação do discente, bem como, a capacidade de realizar os cálculos matemáticos necessários no desenvolvimento da tarefa. Ressalta-se que os outros comandos continuam a serem utilizados no desenvolvimento dos códigos, visto que os conteúdos são ministrados de forma cumulativa.

Objetivos a serem verificados pelo professor ao final da tarefa:

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.

- Analisar possíveis falhas na interpretação do aluno referente ao que foi exposto (leitura correta do programa).

- Identificar se o aluno conseguiu entender a diferença em quando utilizar o comando while de forma contada ou condicional.

-Verificar o uso dos comandos abordados até o momento e suas devidas aplicações dentro dos códigos.

- Analisar se o aluno consegue desenvolver os cálculos necessários para fazer com que o programa desempenhe sua função corretamente.

- Analisar se o aluno consegue utilizar os comandos contador e acumulador de forma correta.

APÊNDICE M - ATIVIDADE 5 – COMANDO DO...WHILE

Introdução

Essa atividade foi desenvolvida com o intuito de analisar a compreensão do discente em utilizar em seu código o laço de repetição *do..while*. Esse comando será apresentado aos alunos na sequência do comando while. Para melhor entendimento do discente, o comando foi apresentado com pequenos exemplos desenvolvidos em conjunto com os estudantes na ferramenta DEV-C++. Posteriormente, foram disponibilizadas aos alunos tarefas para que pudessem desenvolver tanto o raciocínio lógico quanto a escrita de programas na linguagem C. Durante o desenvolvimento das tarefas, os alunos são acompanhados pelo professor, que os auxilia em seu desenvolvimento. Posteriormente, a atividade didática apresentada abaixo é aplicada.

Objetivo da atividade didática 5

Cada atividade didática elaborada possui objetivos tanto para os alunos como o professor.

Aluno

- Identificar o uso do comando *do...while* em problemas específicos.
- Reconhecer e compreender as diferenças entre os comandos *do..while* e *while.*
-Interpretar os comandos em pseudocódigos e reescrevê-los utilizando a linguagem de programação C.
-Ditar ritmo de estudo.

Professor

-Analisar os objetivos elencados em cada tarefa a fim de elaborar as matrizes de análise que servirão de auxílio para o professor acompanhar o aprendizado dos alunos durante o ano letivo.

Tarefa 1

Para desenvolver esta tarefa, o aluno terá que utilizar o comando *do...while*, mas para isso terá que identificar que é essa a estrutura a ser utilizada. Também será necessário desenvolver o programa utilizando a linguagem de programação C e desenvolver os cálculos matemáticos necessários na resolução dos problemas. Ressalta-se que os outros comandos

continuam a serem utilizados no desenvolvimento dos códigos, visto que, os conteúdos são ministrados de forma cumulativa.

Objetivos a serem verificados pelo professor ao final da tarefa:

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.

- Analisar a interpretação do aluno referente ao que foi exposto (leitura correta do programa).

- Identificar a escolha do comando a ser utilizado.

-Verificar o uso dos comandos abordados até o momento e suas devidas aplicações dentro dos códigos.

- Analisar o desenvolvimento dos cálculos matemáticos solicitados nos programas.

Tarefa 2

Para realizar esta tarefa, o discente deve utilizar os conceitos de programação e a linguagem C, vistos em sala de aula. Deve interpretar os códigos dados e reorganizá-los utilizando o comando *do...while*, identificando a diferença entre os comandos *while* e *do...while*.

Objetivos a serem verificados pelo professor ao final da tarefa

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.

- Analisar a interpretação do aluno referente ao que foi exposto para ele (leitura correta do programa).

-Verificar se o aluno é capaz de codificar programas desenvolvidos com o comando while em programas que utilizam o comando do...while, identificando suas diferenças na hora da construção do programa.

- Analisar o desenvolvimento dos cálculos matemáticos solicitados nos programas.

Tarefa 3

Para que esta tarefa seja desenvolvida, o aluno deve interpretar os pseudocódigos dados e codificá-los na linguagem de programação C. Também será necessário desenvolver um fluxograma de cada pseudocódigo, o qual foi apresentado em sala de aula apenas como uma forma de representação de algoritmos, mas não foi desenvolvida nenhuma atividade utilizando essa forma de representação.

Objetivos a serem verificados pelo professor ao final da tarefa

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.

- Analisar a interpretação do aluno referente ao que foi exposto para ele (leitura correta do programa).

-Analisar se o aluno é capaz de ler programas em pseudocódigo e codificá-los na linguagem de programação C, sem a necessidade de ter que iniciar o conteúdo utilizando essa abordagem.

-Verificar se o fluxograma desenvolvido pelo discente está de acordo com o pseudocódigo dado e o programa desenvolvido na linguagem C.

APÊNDICE N - ATIVIDADE 6 – COMANDO FOR

Introdução

Para desenvolver essa atividade, já terá sido exposto ao aluno o conteúdo referente ao comando for. Esse comando é apresentado na sequência dos comandos *if*, *switch..case*, *while* e *do...while*. Para melhor entendimento do discente, o comando foi apresentado com pequenos exemplos desenvolvidos em conjunto com os estudantes na ferramenta DEV-C++. Posteriormente, é disponibilizada uma lista de problemas para que possam ser desenvolvidos o raciocínio lógico, a escrita de programas na linguagem C e o comando for. Durante o desenvolvimento dessas tarefas, os alunos são acompanhados pelo professor, que os auxilia em seu desenvolvimento. Em seguida, a atividade didática é aplicada.

Objetivo da atividade didática 6

Cada atividade didática elaborada possui objetivos tanto para os alunos como o professor.

Aluno

-Identificar o uso e funcionamento do comando *For*.

- Reconhecer e aplicar o comando *For* em diferentes contextos.

- Ditar ritmo de estudo.

Professor

-Analisar os objetivos elencados em cada tarefa a fim de elaborar as matrizes de análise que servirão de auxílio para o professor acompanhar o aprendizado dos alunos durante o ano letivo.

Tarefa 1

Com o intuito de desenvolver essa tarefa o aluno terá que identificar o modo de funcionamento do comando for (inicialização;condição;incremento). Também será necessário desenvolver um programa utilizando a linguagem de programação C enfatizando a utilização desse comando. Ressalta-se que os outros comandos continuam a ser utilizados no desenvolvimento dos códigos, visto que, os conteúdos são ministrados de forma cumulativa.

Objetivos a serem verificados pelo professor ao final da tarefa

-Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.

- Analisar a interpretação do aluno referente ao que foi exposto (leitura correta do programa).

-Verificar o uso dos comandos abordados até o momento e suas devidas aplicações dentro dos códigos.

Tarefa 2

Para desenvolver esta tarefa, os alunos irão utilizar o laboratório de informática, o editor de texto DEVC++ e a linguagem de programação C. Será necessário analisar os programas desenvolvidos com os comandos *while* e *do..while* para, após, desenvolvê-lo com o laço de repetição *For*. Ressalta-se que todos os outros comandos trabalhados em sala de aula continuam a serem utilizados no desenvolvimento dos códigos, visto que os conteúdos são ministrados de forma cumulativa.

Objetivos a serem verificados pelo professor ao final da tarefa

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.

- Analisar a interpretação do aluno referente ao que foi exposto (leitura correta do programa).

-Identificar a compreensão do aluno referente à forma de implementação do laço de repetição for, bem como a diferença entre as três formas de implementar programas que utilizam esses comandos.

-Verificar o uso dos comandos abordados até o momento e suas devidas aplicações dentro dos códigos.

Tarefa 3

A fim de utilizar os conceitos de programação e a linguagem C, vistos em sala de aula, os alunos devem codificar o programa abaixo, utilizando as três estruturas de repetição *while*, *do..while* e *for*. Após, o discente deve explicar a diferença implementada entre os três programas desenvolvidos.

Objetivos a serem verificados pelo professor ao final da tarefa

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.

- Analisar a interpretação do aluno referente ao que foi exposto para ele (leitura correta do programa).

-Verificar se o aluno é capaz de codificar programas utilizando os diferentes comandos de repetição – while, do..while e for.

APÊNDICE O - ATIVIDADE 7 – ESTRUTURA HOMOGÊNEA – VETOR

Introdução

Para desenvolver essa atividade, o aluno já terá visto o conteúdo referente à estrutura homogênea vetor. Esse comando é apresentado aos alunos na sequência dos comandos *if*, *switch..case*, *while*, *do...while* e *for*. Para melhor entendimento do discente, o comando foi apresentado com pequenos exemplos desenvolvidos em conjunto com os estudantes na ferramenta DEV-C++. Posteriormente, foram disponibilizados, aos alunos, problemas para que pudessem desenvolver tanto o raciocínio lógico quanto a escrita de programas na linguagem C. Durante o desenvolvimento dessas tarefas, os alunos são acompanhados pelo professor, que os auxilia em seu desenvolvimento. Em seguida, a atividade didática apresentada abaixo é aplicada.

Objetivo da atividade didática 7

Cada atividade didática elaborada possui objetivos tanto para os alunos como o professor.

Aluno

- Identificar o uso e o funcionamento da estrutura homogênea – Vetor.
- Ditar ritmo de estudo.

Professor

-Analisar os objetivos elencados em cada tarefa a fim de elaborar as matrizes de análise que servirão de auxílio para o professor acompanhar o aprendizado dos alunos durante o ano letivo.

<u>Tarefa 1</u>

A fim de desenvolver esta atividade, os alunos deverão analisar os códigos abaixo, sendo necessário identificar o que é índice, elemento, acumulador e contador a ser utilizado no programa. Ressalta-se que todos os outros comandos trabalhados em sala de aula continuam a ser utilizados no desenvolvimento dos códigos, visto que os conteúdos são ministrados de forma cumulativa.

Objetivos a serem verificados pelo professor ao final da tarefa

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.

- Identificar se o aluno é capaz de compreender a forma de implementação da estrutura homogênea chamada vetor, manipulando corretamente índice e elemento, pois essa é uma dificuldade que o aluno possui.

-Verificar o uso dos comandos abordados até o momento e suas devidas aplicações dentro dos códigos.

Tarefa 2

Para desenvolver esta tarefa, o aluno deverá explicar o funcionamento dos códigos dados e identificar suas saídas. Ressalta-se que todos os outros comandos trabalhados em sala de aula continuam a ser utilizados no desenvolvimento dos códigos, visto que os conteúdos são ministrados de forma cumulativa.

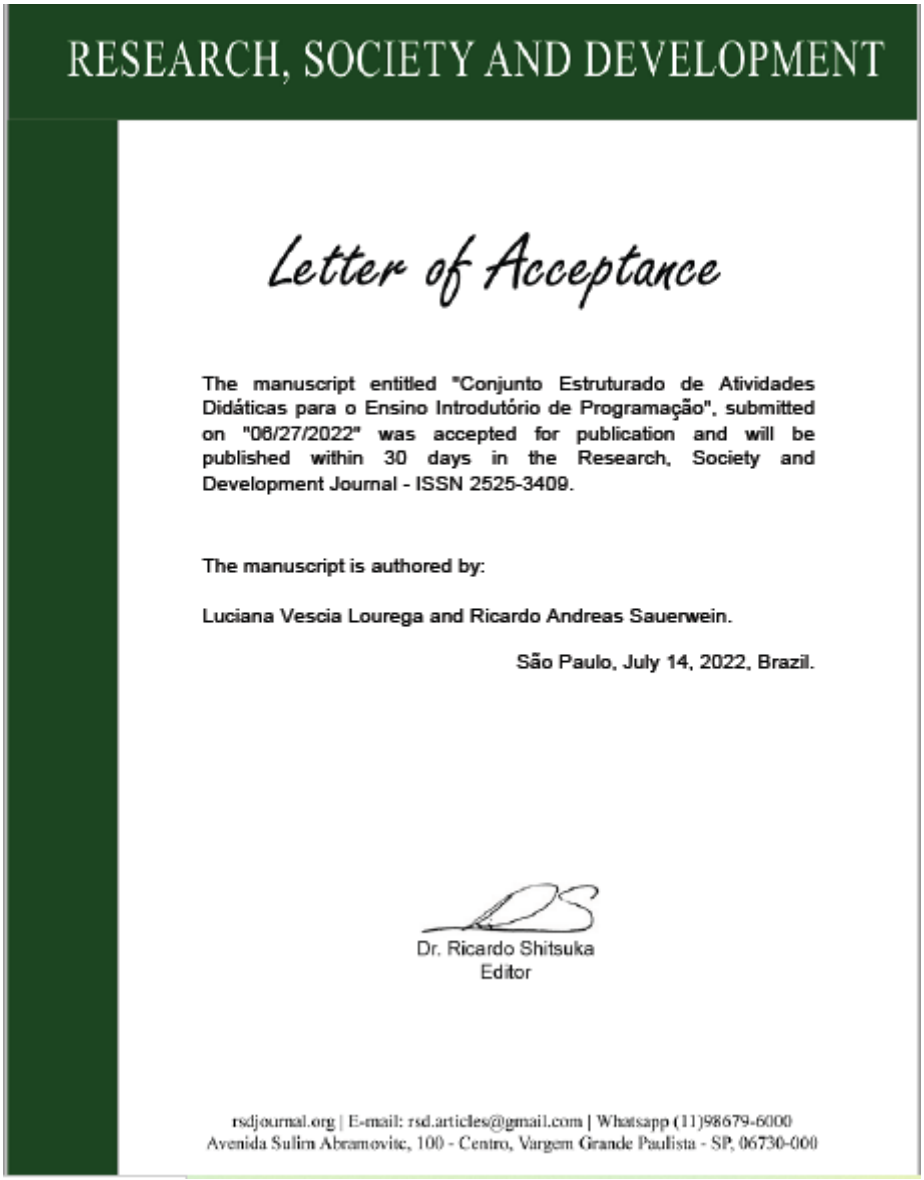
Objetivos a serem verificados pelo professor ao final da tarefa

- Analisar o código para detectar a existência de erros lógicos e/ou sintáticos.

- Analisar a interpretação do aluno referente ao que foi exposto para ele (leitura correta do programa).

-Verificar se o aluno é capaz de explicar a forma de funcionamento da estrutura homogênea chamada vetor, bem como identificar o que será impresso na tela, com as respectivas entradas dadas a eles.

APÊNDICE P – CARTA ACEITE DO ARTIGO PUBLICADO



RESEARCH, SOCIETY AND DEVELOPMENT

Letter of Acceptance

The manuscript entitled "Conjunto Estruturado de Atividades Didáticas para o Ensino Introdutório de Programação", submitted on "06/27/2022" was accepted for publication and will be published within 30 days in the Research, Society and Development Journal - ISSN 2525-3409.

The manuscript is authored by:

Luciana Vescia Lourega and Ricardo Andreas Sauerwein.

São Paulo, July 14, 2022, Brazil.

Dr. Ricardo Shitsuka
Editor

rsdjournal.org | E-mail: rsd.articles@gmail.com | Whatsapp (11)98679-6000
Avenida Sulim Abramovite, 100 - Centro, Vargem Grande Paulista - SP, 06730-000