UNIVERSIDADE FEDERAL DO AMAPÁ

DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

COORDENAÇÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

FELIPE MONTEIRO FARIAS

Papel ou IDE: Uma análise sobre as formas de avaliação em turmas iniciantes de programação

Projeto de Trabalho de Conclusão de Curso

Macapá

2023

Felipe Monteiro Farias

Papel ou IDE: Uma análise sobre as formas de avaliação em turmas iniciantes de programação

Trabalho de Conclusão de Curso submetido à Banca Examinadora do Curso de Ciência da Computação da UNIFAP para a obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Prof. DSc. Julio Cezar Costa Furtado

Macapá
2023

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Central/UNIFAP-Macapá-AP
Elaborado por Mário das Graças Carvalho Lima Júnior – CRB-2 / 1451

F224 Farias, Felipe Monteiro.
Papel ou IDE: uma análise sobre as formas de avaliação em turmas iniciantes de programação / Felipe Monteiro Farias. - Macapá, 2023. 1 recurso eletrônico. 39 folhas.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Amapá, Coordenação do Curso de Ciência da Computação, Macapá, 2023. Orientador: Júlio Costa Furtado.

Modo de acesso: World Wide Web.
Formato de arquivo: Portable Document Format (PDF).

1. Educação. 2. Programação para iniciantes. 3. Papel e caneta. I. Furtado, Júlio Costa, orientador. II. UNIFAP. III. Título.

CDD 23. ed. – 370

FARIAS, Felipe Monteiro. **Papel ou IDE**: Uma análise sobre as formas de avaliação em turmas iniciantes de programação . Orientador: Júlio Costa Furtado. 2023. 39 f. Trabalho de Conclusão de Curso (Graduação) - Coordenação do Curso de Ciência da Computação. Universidade Federal do Amapá, Macapá, 2023.

UNIVERSIDADE FEDERAL DO AMAPÁ
DEPARTAMENTO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
COORDENAÇÃO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

ATA DE DEFESA DE TCC

Realizou-se, no dia 31 de agosto de 2023, às 17h, na Universidade Federal do Amapá, a defesa do TCC intitulado: **"Papel ou IDE: Uma análise sobre as formas de avaliação em turmas iniciantes de programação"**, do discente **Felipe Monteiro Farias**, matrícula 201412200027. A Banca Examinadora foi composta pelo Prof. Dr. Julio Cezar Costa Furtado, presidente da banca e orientadora; Prof. Me. Thiago Pinheiro do Nascimento e Prof. Me. Marco Antonio Leal da Silva, examinadores. Concluída a defesa, foram realizadas as arguições e comentários. Em seguida, procedeu-se o julgamento pelos membros da Banca Examinadora, tendo o trabalho sido APROVADO.

E, para constar, eu, Prof. Dr. Julio Cezar Costa Furtado, orientador e presidente da Banca Examinadora, lavrei a presente ata que, após lida e achada conforme, foi assinada por mim e demais membros da Banca Examinadora.

Macapá-Ap., 31 de agosto de 2023.

gov.br Documento assinado digitalmente
JULIO CEZAR COSTA FURTADO
Data: 31/08/2023 22:42:14-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Julio Cezar Costa Furtado
Orientador do Projeto de TCC

gov.br Documento assinado digitalmente
THIAGO PINHEIRO DO NASCIMENTO
Data: 04/09/2023 18:28:37-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Thiago Pinheiro do Nascimento
Examinador (UNIFAP)

gov.br Documento assinado digitalmente
MARCO ANTONIO LEAL DA SILVA
Data: 04/09/2023 18:55:20-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Marco Antonio Leal da Silva
Examinador (UNIFAP)

Resumo

Este trabalho apresenta os resultados de uma revisão sistemática da literatura que contou com a análise dos artigos sobre o processo de ensino-aprendizagem de programação para iniciantes em cursos voltados para computação. Os resultados obtidos mostraram que há ainda muito para se evoluir no cenário de ensino-aprendizagem de programação na comunidade brasileira na área, que a maioria das pesquisas neste trabalho apresentadas se encontram principalmente na educação superior e que grande parte delas apresenta novas ferramentas de software que buscam tornar mais práticos os métodos de ensino das linguagens de programação. Este trabalho busca evidencias empíricas dos artefatos que influenciam as taxas de sucesso/reprovação nos cursos voltados para computação, e neste trabalho além das pesquisas neste documento apresentadas, também será apresentado um experimento feito em sala de aula utilizando dois métodos de ensino e avaliação em programação efetuado em uma turma iniciante do curso de Ciência da Computação da Universidade Federal do Amapá UNIFAP.

PALAVRAS-CHAVE: Educação; Programação para iniciantes; Papel e caneta; Computador/IDE; Ensino; Avaliação

Abstract

This work presents the results of a systematic review of the literature that included the analysis of articles on the teaching-learning process of programming for beginners in courses focused on computing. The results obtained showed that there is still a lot to evolve in the programming teaching-learning scenario in the Brazilian community in the area, that most of the research presented in this work is mainly in higher education and that a large part of them presents new software tools that seek to make teaching methods of programming languages more practical. This work seeks empirical evidence of the artifacts that influence the success/failure rates in courses focused on computing, and in this work, in addition to the research presented in this document, an experiment carried out in the classroom using two methods of teaching and evaluation in programming carried out in a beginner class of the Computer Science course at the Federal University of Amapá UNIFAP will also be presented.

KEYWORDS: Education; Programming for beginners; Paper and pen; Computer/IDE; Teaching; Assessment

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

MEC	Ministério da Educação
IDE	Integrated Development Environment

SUMÁRIO

1 INTRODUÇÃO

Os cursos de programação têm sofrido desde o seu surgimento com a baixa retenção de alunos, e muito disso se deve por conta da dificuldade de aprendizado (OQVIST e NOURI, 2018). Existem várias possibilidades de origem das dificuldades no ensino de programação, como destaca (RAABE e SILVA, 2005). E também de acordo com Branco Neto e Schuvartz (2007), deve se levar em consideração devido ao conjunto de habilidades que a programação exige como capacidade para solucionar problemas, raciocínio lógico, habilidade matemática, capacidade de abstração, entre outras, fazem acadêmicos iniciantes por sua vez se sentirem incapazes de programar (PRIETCH e PAZETO, 2010).

Segundo as diretrizes curriculares para os cursos de informática e computação do MEC, a matéria de Programação faz parte da área de formação básica em Computação e Informática, juntamente com as matérias de Computação, Algoritmos e Arquitetura de computadores (AZEREDO, 2000). O ensino da programação não é fácil e, devido a isso, muitas universidades discutem com frequência seus currículos de Ciência da Computação em busca de alternativas para diminuir o índice de evasões deste curso. É comum observarmos pesquisas que apontam o grande número de evasões neste curso, fato que tem relação com as dificuldades de aprendizagem (CASTRO *et al.*, 2003).

E como já foi mencionado que a demanda por pessoas qualificadas na área de desenvolvimento supera a oferta em determinados graus, situações essas relatadas por muitas regiões globais como nos Estados Unidos (SIMPSON, 2016), no Canadá (ARELLANO, 2015), no Reino Unido (ANDERSON, 2015) e em vários outros países da União Europeia (VAN HEUR, 2016).

Segundo Raphael Magalhães Hoed (2016) algo que não se pode negar é fato de que aprender a programar por si só já é um grande desafio que por muitas vezes se torna estressante, e uma forma de mitigar tal fato seria se avaliar o aluno no seu aprendizado de uma forma que o seu perfil fosse levado em consideração, sabendo-se que algumas pessoas se sentem mais confortáveis aprendendo a partir do uso de papel e caneta, e outras com o uso de um Ambiente de Desenvolvimento Integrado (IDE).

E todos aqueles envolvidos com gestão acadêmica, devem estar cientes dos fatores relacionados à evasão, e busque formas mudar esse quadro negativo dentro dos cursos de programação combatendo da melhor forma, seja com políticas ou métodos de ensino mais atraentes para o discente (HOED, 2016)

Estudos na área que apoiam a suposição de que um exame digital ou prático avalia com mais precisão a capacidade de programação de um aluno, em comparação com os

exames com caneta e papel, fazem essa afirmação usando marcas de exame ou o número de alunos que passaram no curso antes e após a implementação, como principal medida de comparação (BENNEDSEN e CASPERSEN, 2007; CHAMILLARD e JOINER, 2001; DALY e WALDRON, 2004).

Neste contexto, este trabalho tem como objetivo avaliar e analisar qual método de avaliação pode ser considerado mais eficaz no intuito de garantir a internalização dos conteúdos de programação para alunos iniciantes em cursos de computação.

Também é importante ressaltar que se deve manter um alinhamento construtivo entre o modo de avaliação e os objetivos de aprendizagem, para que os métodos de aprendizagem não tenham finalidades muito diferentes, fazendo assim que seja mais difícil dizer qual método de ensino é mais eficaz em relação ao outro, (HAGHIGHI *et al.*, 2005) e (BIGGS, 2003).

Este trabalho está organizado da seguinte forma. No capítulo 1 a motivação, justificativa e contribuição deste trabalho para a área e, os objetivos deste documento, gerais, específicos e a metodologia de pesquisa utilizada na produção deste trabalho. No capítulo 2 é falado sobre do problema com os métodos avaliativos e de ensino da programação para alunos iniciantes e outras conclusões a respeito, e ainda dentro do capítulo 2, no 2.1 é mostrado as formas de avaliação e ensino de programação, 2.3 fala de forma mais aprofundada sobre formas de ensinar programação por meio da principal discussão desse trabalho, no 2.4 fala sobre outros trabalhos que tem relação com o problema principal apresentado, nos capítulos seguintes 3, 4, 5 e 6 é falado sobre o experimento em sala de aula, a metodologia de pesquisa e avaliação do experimento, bem como sua instrumentação, execução e discussão dos resultados obtidos e finalizando com as considerações finais e referências bibliográficas que serviu como base principal na confecção deste trabalho.

1.1. MOTIVAÇÃO, JUSTIFICATIVA e CONTRIBUIÇÃO à ÁREA.

Um trabalho feito por Pimentel *et al.* (2003) onde fala que os critérios de avaliação utilizados pelas instituições não são satisfatórios por dois motivos. Primeiro porque o fato de um aluno obter média 5.0 numa disciplina não indica que este aluno sabe 50% de tudo aquilo que foi abordado. Ainda segundo Pimentel *et al.* (2003) pode ser que ele tenha aprendido muito bem um tópico e quase nada de outros tópicos correlacionados. Com a ausência do aprendizado destes tópicos, o estudante acabará tendo problemas de acompanhamento nas disciplinas avançadas.

E de acordo com Tobar *et al.* (2001) como consequência muitos estudantes completam seus cursos sem as habilidades de programação desejadas. Segundo um mapeamento sistemático feito a partir da revisão outros trabalhos produzidos por Souza *et al.* (2016), fala que problemas motivacionais estão relacionados com as dificuldades de aprendizagem e aplicação dos conceitos de programação.

Daly e Waldron (2004) falam que na avaliação de um estudante deve ser dada a ênfase à questão relativa às reais competências de programação adquirida. No mesmo trabalho estes autores examinam ainda a razão pela qual os métodos tradicionais de avaliação (escritos e trabalhos de programação) são falaciosos e defeituosos, considerando um outro método de avaliação (avaliações em laboratório) como mais certo.

Ainda segundo o trabalho de Daly e Waldron (2004), fala que se quisermos que nossos alunos sejam bem avaliados para que aprendam a programar com sucesso dando ao aluno um ambiente em que ele possa produzir um programa e ter o *feedback* se o seu programa funciona e desenvolvendo o seu interesse em programar.

Um trabalho feito por Mourant *et al.* (1981) mostraram que os alunos ficam mais cansados ao ler um texto na tela do computador do que quando lêem o mesmo texto no papel, e tambem apesar de alguns autores como Chamillard (2001), Koile e Singer (2006), Daly e Waldron (2004) entre outros autores ja citados anteriormente relatarem diversas vantagens nos testes baseados em computador, Bodmann e Robinson (2004) tambem fala que se o aluno não for familiarizado com o ambiente informatizado ele apresentará dificuldades no seu desempenho em testes usando um computador.

Uma discussão presente no trabalho de Lappalainen *et al.* (2016), onde fala que trabalhar o ensino de programação para iniciantes usando IDE, (Ambiente de Desenvolvimento Integrado), ignorando pequenos erros sintáticos, produz melhores soluções do que somente trabalhar com papel e caneta.

Depois de tudo relatado acima ainda não é possível afirmar com 100% de certeza que o ensino e avaliação a partir de um computador é suficiente para suprir todas as necessidades e problemas no ensino da programação de computadores, dado o fato que existem outros fatores que implicam num aprendizado 100% eficaz, por conta de particularidades individuais de cada aluno.

Assim, este trabalho busca explorar qual a melhor forma de avaliar as disciplinas iniciais de quem está começando o estudo de programação, falando das vantagens e desvantagem de cada método avaliativo buscando nas literaturas, exemplos que mostram os desafios do ensino de programação para iniciantes.

1.2. OBJETIVOS

1.2.1 Objetivo Geral.

Avaliar qual método de avaliação (papel e caneta ou computador/IDE) atinge o melhor resultado no processo de aprendizagem de um aluno iniciante em programação.

1.2.2 Objetivos Específicos.

1. Realizar uma Revisão da Literatura;
2. Identificar, nos trabalhos selecionados durante a revisão, técnicas e ferramentas de apoio à avaliação das habilidades de programação para alunos iniciantes;
3. Realizar um experimento em sala de aula comparando os métodos de avaliação de programação através do papel e caneta e pelo uso de um computador/IDE.

1.3 METODOLOGIA DE PESQUISA

A metodologia utilizada neste trabalho se divide em 3 fases.

Na primeira fase, será feita uma revisão sistemática da literatura, a fim de identificar técnicas, metodologias e ferramentas de software que visassem resolver ou ao menos mitigar o problema que é tentar ensinar e avaliar o aprendizado em programação para alunos iniciantes de forma efetiva.

Na segunda fase, será elaboração do texto principal do trabalho onde será apresentado à experiencia de diversos autores sobre o assunto mostrando o que cada um tem a falar sobre qual metodologia é mais eficaz e menos eficaz no ensino e na avaliação de programação para novatos levando em consideração alguns experimentos mostrados em alguns trabalhos, e dando base para um experimento que será produzido em sala de aula.

Como já foi citado acima, a terceira fase deste trabalho se dará por meio de um experimento em sala de aula que será da seguinte forma:

— Em uma disciplina de programação inicial de um curso de Computação, os alunos irão receber o treinamento formal sobre o conteúdo.

— No momento da avaliação, a turma será divindade em 2 grupos. Onde, o Grupo 1 fará as avaliações fazendo uso do recurso papel e caneta, e o Grupo 2 fará as avaliações usando uma IDE (Computador).

— Ao, será realizado um pequeno projeto prático utilizando-se de uma ferramenta computacional sem o poder de uma IDE (ex.: sem sugestões de autocompletar). Assim, serão comparados os efeitos das duas abordagens de avaliação sobre a aprendizagem do conteúdo e aquisição das competências pelos alunos.

Em paralelo a estas fases, será realizada a escrita do documento do TCC.

2 CONTEXTUALIZAÇÃO

Neste capítulo mostraremos os métodos de ensino e avaliação de programação para quem está começando e buscando expor nesse capítulo o que melhor ajuda o aluno e o que pode não ser tão eficaz no seu processo de aprendizagem, mostrando também que cada aluno reage de uma forma específica a cada método aqui apresentado, levando em consideração várias características.

2.1 MÉTODOS DE ENSINO E AVALIAÇÃO EM EDUCAÇÃO

A metodologia de ensino-aprendizagem e de avaliação são vitais para o sucesso de um curso a distância. O desenvolvimento de uma metodologia pedagógica que tenha como objetivo repensar o papel do professor e do aluno no processo de ensinar e aprender deve ser constantemente revisado e atualizado. Para que o processo de ensino-aprendizagem, bem como o de avaliação, seja eficaz deve-se levar em consideração o processo de reflexão sobre as experiências individuais de cada participante juntamente com a abordagem teórica das metodologias pedagógicas, as quais conduzirão ao autodesenvolvimento, à aprendizagem colaborativa e às aulas com maior interação entre professor e alunos. (MAIA, *et al.*, 2005).

2.1.1 Métodos de Ensino

2.1.1.1 Aulas Expositivas.

Sabe-se que o conteúdo de uma disciplina de programação é apresentado aos alunos através de aulas expositivas ou práticas, onde o ritmo de evolução do conteúdo é dado pelo professor, tendo-se em vista o cumprimento de um conteúdo programático. (ROCHA, 2010).

A aula expositiva foi o único procedimento empregado durante muito tempo em sala de aula. No século passado, no entanto, ela perdeu espaço na escola, mas se bem planejada e realizada, essa estratégia de ensino onde o protagonista(professor) e conduz a turma por um raciocínio — pode ser o melhor meio de ensinar determinados conteúdos, mas nunca deve ser o único recurso. (FERNANDES, 2011).

Com base em Prikladnicki *et al.* (2009), uma aula dada por meio de abordagem expositiva não costuma possuir muita eficiência, uma vez que apenas o sentido da audição é utilizado. Já eventos que simulem problemas reais e atividades vivenciais permitem que o acadêmico assimile situações diferentes, na prática.

2.1.1.2 Aulas Laboratoriais

Segundo Prior (2003), a habilidade de programação de computadores não pode ser adquirida sem um significante esforço em atividades práticas de laboratório.

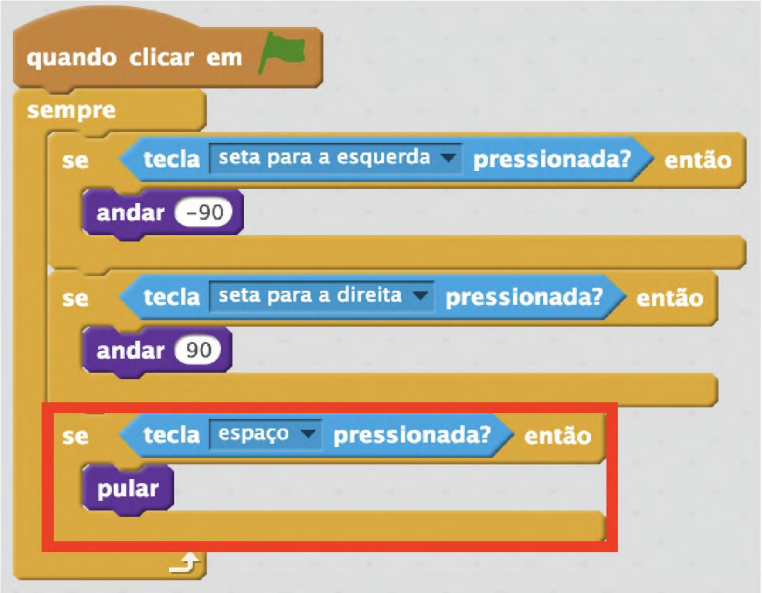
Aulas em laboratórios são extremamente importantes para o desenvolvimento do aluno durante seu aprendizado no ensino de programação, uma vez que o ambiente mais prático como um laboratório de informática, instiga o aluno e o deixa mais familiarizado com o mundo computacional. As Instituições de Ensino Superior (IES) cujos cursos apresentam necessidades de experimentação laboratorial têm enfrentado muitas dificuldades quanto à qualidade de ensino. De acordo com Victorino *et al.* (2009), a realização de trabalhos experimentais em laboratórios devidamente equipados é essencial para se aprender (ALMEIDA, 2016).

2.1.1.3 Casos Práticos

Um bom exemplo de caso prático envolvendo o ensino de programação é o uso da ferramenta SCRATCH que foi desenvolvida com o objetivo de incentivar a aprendizagem da programação de forma intuitiva por meio da montagem dos blocos de comandos.

Segundo Bressan e Amaral (2015). No Scratch, o algoritmo, a sequência de instruções pode ser modificada a qualquer momento, facilitando a experimentação simples de novas ideias e o multiprocessamento é integrado maneira simplificada e, podendo ainda serem executadas instruções paralelamente por outros conjuntos de blocos conforme (Figura 1).

Figura 1. Blocos de construção do Scratch (Building Blocks) Fonte: Dados de pesquisa



Construindo e arrastando os blocos de comandos, os sujeitos discentes aprendem com o erro, ainda por meio do erro, a analisar e elaborar hipóteses para alcançar o objetivo desejado, buscando soluções (BRESSAN E AMARAL, 2015).

Outros tipos de ferramentas e ambientes têm sido propostos com o objetivo de facilitar o aprendizado de lógica e linguagens de programação. Dentre outras, podemos destacar: ASTRAL (REZENDE e GARCIA, 1997) AUTOMATA SIMULATOR e IC (JANDL e ZUCHINI, 1999), C-Tutor (SONG *et al.*, 1997), BALSA-II (BROWN, 1988), ZEUS (BROWN, 1991), SICAS (GOMES e MENDES, 2000).

2.1.1.4 Uso de Jogos

O uso e jogos educativos com o objetivo e ensinar iniciantes a programar é algo que é inovador e tem um impacto grande no aprendizado, já que busca de uma forma lúdica fazendo o ato de jogar se transformar num ato de aprender destaca-se: podem ser ferramentas eficientes, pois, eles divertem enquanto motivam, facilitam a aprendizagem e aumentam a capacidade de retenção do que é ensinado, exercitando as funções mentais e intelectuais do jogador (TAROUCO *et. al.*, 2004).

Mas o uso de jogos digitais na educação não apresenta só benefícios. Segundo Falkembach (2007), mesmo um jogo bem projetado pode ter algumas desvantagens como: se não for bem aplicado perde o objetivo; nem todos os conceitos podem ser explicados por meio dos jogos; se o professor interferir com frequência, perde a ludicidade.

E segundo Patrick Moratori (2003), o educador precisa seguir alguns passos:

- propor regras ao invés de impô-las, permitindo que o aluno as elabore e tome decisões;
- promover a troca de idéias para chegar a um acordo sobre as regras;
- permitir julgar qual regra deve ser aplicada a cada situação;
- motivar o desenvolvimento da iniciativa, agilidade e confiança;
- contribuir para o desenvolvimento da autonomia.

2.1.2 Formas de Avaliação:

Avaliação é uma ferramenta educação que busca a eficácia da validação de todo o processo de ensino, onde se tem algumas formas de avaliar tornando o processo de aprendizagem algo evoluído e eficaz no que ele se propõe, como formas de avaliação automática, diagnóstica, formativa e somativa.

2.1.2.1 Avaliação Automática

Pesquisas sobre o uso de avaliação automática (LINO *et al* 2007), (SAIKKONEN *et al* 2001) (KAY *et al* 1994), em que o próprio sistema é um dos atores do processo. E um trabalho, de Moreira e Favero da UFPA (2009) propôs a implementação de um ambiente de aprendizagem utilizando avaliação automática para ensino de programação. Os principais benefícios para o educador são entre outros (KJOLLERSTROM & MARTENSSON, 1999):

- Menor esforço, uma vez que conta com o auxílio da ferramenta;
- Melhor administração dos estudantes e de suas tarefas;
- Melhor rastreamento individual dos estudantes;
- Melhor qualidade de ensino, devido o maior tempo de prática;
- Mais tempo para contato com os estudantes.

2.1.2.2 Avaliação Diagnóstica

Conhecer o aluno, seus gostos, seus hábitos e suas preferências, é o princípio base da avaliação diagnóstica. Dessa forma, assegura-se que o aluno esteja na turma correta e que o curso se encontre no nível adequado a ele. Nesta avaliação busca-se conhecer ideias e conhecimentos prévios do aluno (MASETTO, 1997).

Uma avaliação diagnóstica pode ser feita por meio de vários instrumentos, como questionários, contendo questões abertas e fechadas, entrevistas, pautas de observação e outros instrumentos escolhidos pelo professor e pela escola. Quando esta avaliação faz referência a um conjunto, ou seja, um grupo 'classe', dá-se o nome de prognose; quando é diferenciada, na qual cada aluno é analisado individualmente, dá-se o nome de diagnose (BALLESTER *et al.*, 2003).

Esta avaliação é feita quando o aluno chega à escola, em geral, no início do ano, seja de um curso, período letivo ou unidade de ensino. Mas é importante ressaltar que a avaliação diagnóstica pode ser refeita em qualquer momento pelo professor ou pela escola, uma vez que forem detectados problemas graves de aprendizagem, motivação ou adaptação à turma em que o aluno está inserido (OLIVEIRA & CHADWICK, 2007).

2.1.2.3 Avaliação Formativa

A avaliação formativa responde a uma concepção do ensino que considera que aprender é um longo processo, por meio do qual o aluno vai reestruturando seu conhecimento a partir das atividades que executa. De acordo com Ballester *et al.* (2003), se um estudante não aprende, não é apenas porque não estuda ou não possui as capacidades mínimas, a causa pode estar nas atividades que não lhes são propostas.

Este tipo de avaliação permite ajustar o processo de ensino-aprendizagem, detectando os pontos frágeis de cada estudante e respondendo às características de cada um deles. Assim o aluno conhece seus erros e acertos e encontra estímulo para um estudo sistemático. Os erros tornam-se objeto de estudo para o professor, por meio dos quais se diagnostica as principais dificuldades e facilidades dos alunos, permitindo assim a elaboração de novas estratégias de ensino (BALLESTER *et al.,* 2003; HAYDT, 2007).

A avaliação formativa está muito ligada ao processo de *feedback,* além do aperfeiçoamento do processo de ensino-aprendizagem. Quando bem empregado, este processo de avaliação garante a qualidade do ensino e assegura que a maioria dos alunos atinja o objetivo esperado, pois, tem como propósito ajudar o aluno a melhorar o seu desempenho (BALLESTER *et al.*, 2003; OLIVEIRA & CHADWICK, 2007).

2.1.2.4 Avaliação Somativa

Avaliação somativa é uma decisão que leva em conta a soma de um ou mais resultados e pode ser baseada numa só prova final (OLIVEIRA & CHADWICK, 2007).

É importante ressaltar que na avaliação somativa pode-se utilizar dados obtidos na avaliação formativa como forma de resultados, seja a partir de testes ou outros instrumentos (BALLESTER *et al.,* 2003). Porém, faz-se necessário lembrar que avaliação é diferente de teste em um processo avaliativo. Haydt (2007) diz que testar é submeter a um experimento ou teste, o que consiste em verificar o desempenho de alguém ou alguma coisa (material, máquina, etc.), por meio de situações com uma organização prévia.

Para Ballester *et al.* (2003), uma avaliação somativa possui uma função social de assegurar que as características dos estudantes respondam a determinadas exigências feitas pelo sistema. Porém, tem ainda uma função formativa de descobrir se os alunos conseguiram atingir comportamentos que haviam sido previstos pelos professores e como consequência, possuírem pré-requisitos básicos e necessários para aprendizagens posteriores ou até mesmo aspectos que deveriam ser modificados.

De acordo com Haydt (2007), esse tipo de avaliação tem por princípio classificar os resultados de aprendizagem alcançados pelos alunos de acordo com os níveis de aproveitamento estabelecidos, adotando assim uma função classificatória.

2.2 OUTRAS CONCLUSÕES.

Apesar de várias metodologias propostas terem verificado melhores índices de aprendizado no domínio de Programação, não se encontrou na literatura pesquisada, metodologias que possibilitem tratar cada aprendiz de maneira diferenciada. Alunos não são iguais: possuem origens, experiências e habilidades diferentes. Isto explica, em parte, o fato de alunos de uma mesma classe, submetidos às mesmas condições de ensino, apresentarem resultados distintos (CARDOSO e JANDL, 1998).

Observando-se o plano de desenvolvimento de disciplinas de programação, em diferentes instituições, constata-se que os critérios de avaliação não variam muito. Normalmente esta avaliação engloba uma prova e a elaboração de um projeto a cada bimestre, além da entrega de listas de exercícios (KURNIA *et al.*, 2001).

Não é suficiente apenas conhecer as ferramentas de programação e suas características. É necessário entender, de uma forma teórica e prática, como as crianças vêm a entender a programação e qual é a melhor forma de ensinar (GROVER, 2013; GARNELI, 2014).

Dentre os pontos mais discutidos no processo educativo está a avaliação; a coerência do que o professor ensina e a forma como ele avalia a aprendizagem são os primeiros fatores para encaminhar um bom processo educativo (MORETTO, 2007).

Observando-se o plano de desenvolvimento de disciplinas de programação, em diferentes instituições, constata-se que os critérios de avaliação não variam muito. Normalmente esta avaliação engloba uma prova e a elaboração de um projeto a cada bimestre, além da entrega de listas de exercícios (KURNIA *et al.*, 2001).

Uma grande problemática a respeito dos métodos de avaliação em áreas do ensino de programação, se trata da abordagem de alguns professores ainda tendem a ser tradicional, focando em aulas muito expositivas aplicando provas pouco interativas desmotivando o aluno que poderia acabar evoluindo como um programador, se caso recebesse um método de ensino e avaliativo mais motivador que buscasse uma aplicação mais prática dos principais tópicos de programação (PRIKLADNICKI *et al.*, 2009) (BESSA *et al.*, 2012).

Sobre os métodos tradicionais de ensino que vem sendo passados de professores para alunos ainda da mesma forma como tem sido há anos (MARQUES, QUISPE e CHOA, 2014) (MEIRA, 2015). E isso é ruim, pois, o método de ensino tradicional não acompanhou a evolução das práticas de negócios do mundo atual, tornando assim o método tradicional pouco efetivo na formação do profissional almejado pela indústria, já que essa prática tradicional não foca no envolvimento do aluno com atividades práticas, mas sim no professor e seu conteúdo (PRIKLADNICKI *et al.*, 2009).

Cada modo de avaliação, além de apresentar seu próprio conjunto de vantagens e desvantagens (CHAMILLARD e JOINER, 2001), também está associado a um provável resultado de aprendizado (BIGGS, 2003).

Por outro lado, não há um consenso em "como" ensinar a programar, pois, cada instituição de ensino adota seus próprios métodos ou abordagens de acordo com a experiência de seus professores (MARQUES, QUISPE e OCHOA, 2014). Nesse contexto, de acordo com o mapeamento realizado por Santos *et al.* (2014), existem estudos reportando a realização de projetos práticos, uso de jogos, discussão de casos práticos, dentre outras abordagens para ensinar a programação.

As abordagens mais comuns para ensinar programação incluem aulas expositivas, aulas de laboratório, entre outros. Entretanto, abordagens alternativas podem ajudar os alunos a aprender de maneira mais efetiva como, por exemplo: a substituição de aulas expositivas por discussão de casos práticos (GNATZ *et al.*, 2003), dinâmicas de grupo, o uso de jogos (WANGENHEIM e SHULL, 2009) e *Capstone projects* (um esforço em grupo em que alunos executam um projeto do início ao fim) (GOOLD e HORAN, 2002).

Práticas e abordagens que buscam o amadurecimento do aluno como aprendiz de programação, e futuramente um profissional da área de TI, deve receber de seu professor numa perspectiva construtivista, a finalidade última da intervenção pedagógica é contribuir

para que o aluno desenvolva a capacidade de realizar aprendizagens significativas por si mesmo numa ampla gama de situações e circunstâncias, que o aluno 'aprenda a aprender. (SALVADOR, 1994).

2.3 FORMAS DE ENSINAR PROGRAMAÇÃO.

Programar é apenas uma ferramenta, um meio para um fim. O que realmente almeja-se é desenvolver uma nova forma de pensar (BRIKMAN, 2014). Duas competências principais devem ser trabalhadas ao ensinar a programar: geração de programas e compreensão de programas. (VAN MERRIENBOER, 1987; ROBINS *et al.*, 2003; MANNILA, 2007, apud SAELI *et al.*, 2011, p. 78).

Schimiguel (2003), por exemplo, apresenta como alternativa para ensino da programação uma ferramenta que utiliza a técnica de desenvolvimento do fluxograma para construção da lógica de programação do aluno. Outros como PIER**SON** (1998) e STERN (1999) apresentam algoritmos animados utilizados para o ensino de estruturas de dados.

Uma ótima sugestão de que o aprendizado é uma atividade cumulativa, e que o sujeito aprende fazendo a ligação entre o conhecimento novo e aquele já adquirido. Assim, sendo, na programação o aluno deve sempre estar desempenhando alguma atividade, sendo o principal ator na resolução de problemas, acarretando assim um aprimoramento do seu pensamento computacional (NOVAK, 2002).

Um exemplo a ser citado como um método para o auxílio no ensino de programação é o Scratch, uma ferramenta que utiliza diferentes métodos aquém dos usados em sala de aula, com o objetivo de tornar a introdução a programação mais simples e intuitiva ao aluno. O Scratch torna a programação mais divertida e simples e aponta que o aluno que utilizar o software desde o início do seu aprendizado tende a ter melhor compreensão dos conceitos básicos da programação (PEREIRA *et al.*, 2012).

Também deve se dar notoriedade para o uso de um método ainda pouco explorado é a Gamificação no contexto de educação, e diversos autores citam que existem benefícios no uso da Gamificação em escolas e faculdades, fazendo com que seja vista de forma positiva já que a Gamificação busca "estabelecer uma interlocução diferenciada com os sujeitos de forma atraente e produtiva, promovendo uma maior apreensão, difusão e construção do conhecimento bem como propiciando o desenvolvimento de distintas competências e habilidades" (OLIVEIRA, 2015).

Uma forma prática e eficaz no ensino de programação é o uso de linguagens práticas como o Python (MENEZES, 2014) que é uma linguagem muito indicada para o ensino de

programação inicial que foi desenvolvida por Guido Van Rossum em 1991. Tal linguagem possui uma sintaxe de linhas de comandos e permite ao programador desenvolver tanto projetos simples quanto complexos. Também vale ressaltar que o python é uma linguagem de programação muito utilizada dada a sua simplicidade e praticidade, já que seu uso não se limita apenas ao desenvolvimento profissional, mas também ao ensino a jovens alunos iniciantes no estudo de programação em cursos da computação.

A programação de computadores tem sido uma das principais disciplinas responsáveis pelo aumento da taxa de reprovação e evasão em cursos da área de Computação (SILVA *et al.*, 2015, SANTIAGO, 2016). No estudo conduzido por Gomes e Mendes (2007), para muitos estudantes as dificuldades relacionadas ao aprendizado de programação começam na fase inicial de aprendizagem, quando eles precisam entender e aplicar conceitos abstratos de programação. (GOMES, 2007).

Então dentro do universo de formas de ensino da programação em cursos da computação, existem alguns métodos como o uso do papel e caneta e também o uso de um Ambiente de Desenvolvimento Integrado (IDE).

2.3.1 Aprendizagem a partir do papel e caneta.

Um estudo publicado na revista científica "*Psychological Science*" indica que fazer anotações no papel é melhor para o estudo do que fazê-las em computadores. De acordo com a pesquisa, os participantes que fizeram notas em papel sobre algumas palestras tiveram melhor desempenho nos testes realizados posteriormente do que os que usaram o notebook, mesmo com ele desconectado da internet. O levantamento foi feito com estudantes das universidades norte-americanas de Princeton e UCLA (Universidade da Califórnia). E isso nos mostra que o uso do papel e caneta no aprendizado da programação, se partirmos do pressuposto de que a memória é uma ferramenta poderosa e deve ser utilizada no aprendizado de alunos iniciantes em turmas de programação. Disponível em: < Papel e caneta são melhores para memória do que computador, diz pesquisa - Gabriel Chalita>

2.3.2 Aprendizagem a partir de uma IDE.

Uma solução aplicada por outras pessoas para combater os efeitos inerentemente negativos dos exames escritos foi a introdução de um exame computadorizado em que os alunos têm acesso a ambientes de programação apropriados e a capacidade de usar o

compilador para permitir que os alunos verifiquem suas soluções (CHAMILLARD E JOINER 2001; HAGHIGHI *et al.*, 2005; RAJALA *et al.*, 2016).

Segundo um trabalho feito por Dillon *et al.* (2012), onde fala que ambientes visuais, como IDE's pedagógicas, como o SCRATCH, são um apoio muito eficaz no aprendizado de programação para iniciantes devido a sua flexibilidade e quantidade de recursos, porém, também é recomendado que o aluno faça transição para um ambiente "menos" assistivo, como IDE's mais avançadas e menos pedagógicas, permitindo assim sua evolução no aprendizado de programação.

E de acordo com (RAJALA *et al.*, 2016), alguns alunos estão mais confortáveis com o uso do computador para codificar do que o papel e caneta por conta de achar o processo, não computadorizado, lento, tedioso e menos prático possibilitando uma maior interação do aluno com o ensino da programação.

2.4 TRANALHOS RELACIONADOS.

Um estudo bem elaborado feito por Jalal Nouri (2018) sobre a forma como pode ser ensinado e avaliado a programação para alunos iniciantes em cursos de computação, afim e mostrar os desafios do aprendizado da programação e tentar de alguma forma demonstrar uma maneira de mitigar os efeitos negativos dos métodos de ensino e avaliação da programação, evitando assim também a grande evasão de alunos dos cursos voltados para a computação permitindo a criação de profissionais qualificados para o mercado de trabalho tecnológico.

Através da realização de um experimento com dois grupos (teste/controle) onde seriam analisados dados quantitativos coletados dos alunos, comparando os resultados do grupo teste (modo de exame digital) e do grupo controle (modo de exame de papel e caneta). Como estudos anteriores já demonstraram que o acesso ao compilador e aos conjuntos de testes tem uma correlação positiva com os resultados dos testes dos alunos (ENGLISH, 2002; JACOBSON, 2000; RAJALA *et al.*, 2016), esses recursos não serão incluídos como variáveis neste estudo.

E os resultados obtidos a partir do experimento de acordo com o estudo se mostraram inconclusivos já que a diferença não foi estatisticamente significativa na maioria dos testes. Porém, apesar dos resultados obtidos, foi mencionado em Haghighi *et al.* (2005), que os alunos descobriram que a capacidade de testar sua solução é o aspecto mais útil do uso de uma plataforma digital em comparação com uma abordagem de caneta e papel.

Em um artigo de Barros (2018) foram apresentados os resultados de um estudo onde dois grupos de alunos, em uma introdução curso de programação, foram avaliados sobre suas preferências em relação a ambos os tipos de testes e também as razões pelas quais eles preferem um ao outro.

Neste trabalho foi feito o uso de um questionário anônimo, onde todos os alunos foram convidados a fazer, onde os alunos foram divididos em 2 grupos (A e B) e o mesmo questionário foi aplicado para ambos os grupos para se saber sobre a preferência sobre a modalidade dos testes se o aluno preferia que os testes fossem feitos em papel ou no computador.

Como resultado obtido os alunos têm uma preferência avassaladora por testes baseados em computador, mesmo que os alunos reconheçam as vantagens dos testes baseados em papel, eles preferem testes no computador revelando assim um efeito significativo na motivação do aprendizado dos alunos.

Um trabalho feito por Waldron (2014), procura mostrar, que dentre as formas de avaliação de habilidades em computação, os testes baseados em computador (exames de laboratório) são mais precisos do que exames escritos, apesar de falar também que sua aplicação é menos simples dos testes em papel.

Na revisão sistemática feita por Daly e Waldron (2014), os alunos passaram por dois testes em laboratórios, um teste feito após um período de experiência em programação e outro teste com um período maior de experiência e a nota dos alunos viria da média desses testes, e foi mostrado a partir dos resultados que pelo menos 90% dos alunos conseguiam compilar um simples programa.

E como resultados notamos que testes baseados em computadores de uma certa forma se mostram mais eficazes no aprendizado e evolução de alunos que estão iniciando sua vida acadêmica na área de programação, e juntamente com uma avaliação eficaz do seu aprendizado o aluno se sente incentivado a programar de uma forma que ele perceba que está evoluindo e aprendendo e sendo bem avaliado.

3 EXPERIMENTO EM SALA DE AULA

Neste capítulo será apresentado o estudo do experimento que foi executado em sala de aula buscando avaliar métodos de ensino de programação usando uma turma iniciante do curso de Bacharelado em Ciência da Computação da Universidade Federal do Amapá. O experimento aconteceu no contexto da disciplina de programação I apresentando para os alunos 2 problemas simples, envolvendo lógica de programação e seus conceitos básicos.

O objetivo do estudo foi descrito utilizando o *Goal, Question, Metric* — GQM (Basili, Caldiera e Rombach, 1994):

- Objetivo do Estudo: Avaliar a melhor forma de avaliação dos conhecimentos em uma disciplina de programa em cursos de graduação em Computação;

4 ESTRATÉGIA DE PESQUISA E AVALIAÇÃO

Um experimento formal foi aplicado, dividindo a população em grupo controle e grupo experimental, com o objetivo de avaliar a eficácia das atividades de avaliação planejadas (Cohen, Manion e Morrison, 2000). Este desenho do experimento permitiu uma comparação estatística do comportamento observado no grupo experimental em relação ao observado no grupo controle (Campbell e Stanley, 1963). O experimento foi organizado da seguinte forma:

1. No início do experimento, os grupos foram divididos de forma randomizada;
2. O conteúdo planejado foi ministrado parar ambos os grupos envolvidos no experimento;
3. Após o término da unidade estudada, os dois grupos responderam a um pós-teste sobre os conteúdos apresentados.

O Quadro 1.1 apresenta um resumo destas informações sobre o design do experimento.

Quadro 1.1. Sumário do experimento

Grupos	***Preparação***	***Intervenções***		***Término***
Controle	Alocação dos grupos.	Aula.	Avaliação Papel/Caneta	Questionários sobre a experiência
Experimental			Avaliação IDE	

Como forma de se avaliar a prova feita pelo estudante, o Quadro 1.2 apresenta a pontuação aplicada de acordo com os critérios de acerto do aluno.

Quadro 1.2. *Scoring Rubric*

Pontos	***Critérios***
1	Adicionou o include da stdio.h
1	Declarou corretamente a função main
1	Declarou corretamente as variáveis
1	Usou corretamente o printf
1	Fez o uso da sintaxe correta do if/else
1	O programa está divido em entrada, processamento e saída
1	A algoritmo resolveu o problema
0	O estudante não respondeu à questão

4.1 INSTRUMENTAÇÃO

Os pós-testes anônimos foram usados para coletar os dados necessários para responder as questões de pesquisa 1. O teste consistia de problemas práticos para serem solucionados, de acordo com o conteúdo ensinado na Unidade. Estes testes estão disponíveis para consulta no Apêndice A.

Foram aplicadas as mesmas questões nos pós-testes para ambos os grupos, sendo corrigidas por dois especialistas na área que não estavam envolvidos com o ensino do conteúdo. As pontuações destes testes foram calculadas conforme descrito no Quadro 1.3.

Em todos os testes nenhuma informação dos participantes era apresentada. As pontuações de todos estes testes foram disponibilizadas aos alunos apenas ao término do experimento.

4.2 EXECUÇÃO

O experimento foi realizado no segundo semestre de 2022, na disciplina de programação I do curso de Bacharelado em Ciência da Computação da Universidade Federal do Amapá. Os alunos apenas foram informados sobre a natureza do experimento durante a apresentação da disciplina.

Todos os participantes do experimento foram voluntários e assinaram um Termo de Consentimento Livre e Esclarecido — TCLE. O curso contou com 39 alunos matriculados (20 alunos em um grupo e 19 em outro grupo). Assim, a agenda do experimento é descrita no Quadro 1.3.

Quadro 1.3. Agenda do experimento

Dia	*Grupo de Controle*	*Grupo Experimental*
1	Aula expositiva tradicional sobre: • 1.1 – Estrutura de um Programa em C • 1.2 – Nomes e Tipos de Variáveis • 1.3 – Instruçoes de E/S • 1.4 – Comandos de E/S • 1.5 – Estruturas de Decisão	
2	Aplicação da Avaliação em Papel/Caneta	Avaliação na IDE

4.2.1 Análise das Questões de Pesquisa

O teste de normalidade de Shapiro e Wilk (1965) foi aplicado para verificar se as pontuações obtidas pelos alunos na avaliação possuíam uma distribuição normal. Para o conjunto de dados do experimento (Apêndice B), a aplicação do teste de normalidade resultou para os dados do pós-teste um W igual a 0,895, não sendo esse conjunto de dados normalmente distribuído para $p > 0,05$. Por este motivo, precisou-se optar por um teste estatístico mais fraco, escolhendo-se o Mann-Whitney U.

O grupo Experimental pontuou 45,00±23,24 no pós-teste, enquanto que o grupo de Controle pontuou 41,58±30,32 no pós-teste. Assim, existe uma diferença real entre os grupos, onde a pontuação do Experimental é D=3,42 maior que o de Controle. O grupo Experimental também apresento um menor desvio padrão do que o grupo controle. A Tabela 1.4 sintetiza os resultados obtidos.

Tabela 1.4 – Comparação da eficácia de aprendizagem entre os grupos participantes no pós-teste (*Student* t)

Variáveis	*Grupo Experimental*	*Grupo Controle*
	Pós-teste	*Pós-teste*
Tamanho da amostra	20	19
Mínimo	7,5	0,0
Máximo	70,0	85,0
Soma dos pontos	423,0	357,0
Mediana	21,25	27,50
Primeiro quartil	13,13	0,00
Terceiro quartil	58,75	57,5
Média	45,00	41,58
Desvio padrão	23,24	30,32
p-*value*	0,26435	

5 DISCUSSÃO DOS RESULTADOS

Com base nos resultados do experimento, parece que os alunos tiveram um desempenho ligeiramente melhor na avaliação realizada no computador (IDE) do que na avaliação realizada com papel e caneta. A média das notas foi ligeiramente maior para a avaliação no computador, e houve menos variabilidade nas notas (como indicado pelo desvio padrão menor). Isso poderia sugerir que o uso de uma IDE pode ter ajudado os alunos a escrever código mais eficaz ou correto, possivelmente devido aos recursos de uma IDE, como destaque de sintaxe, autocompletar e depuração. Esses recursos podem ter ajudado os alunos a evitar erros e a escrever código mais eficiente.

Porém, é importante notar que existem muitos fatores que podem influenciar o desempenho dos alunos em uma avaliação, e a diferença nas médias que observamos pode não ser inteiramente devido ao formato da avaliação. Outros fatores, como o conteúdo da avaliação, a preparação dos alunos, e até mesmo o dia e a hora em que a avaliação foi realizada, podem ter desempenhado um papel.

No entanto, também é importante notar que a avaliação com papel e caneta teve uma maior variabilidade nas notas. Isso poderia sugerir que essa forma de avaliação pode ser mais discriminatória em termos de habilidade dos alunos. Em outras palavras, pode ser mais capaz de diferenciar entre alunos de diferentes níveis de habilidade. Isso pode ser porque a avaliação com papel e caneta requer uma compreensão mais profunda da lógica de programação e da estrutura do código, já que os alunos não podem depender de recursos de autocompletar ou depuração.

O desvio padrão das notas médias da turma IDE é 23.12, enquanto o desvio padrão das notas médias da turma PAPEL é 29.54. O desvio padrão é uma medida de dispersão que indica o quanto os resultados individuais se desviam da média. Um desvio padrão maior indica uma maior variabilidade nos resultados. Neste caso, a turma PAPEL tem um desvio padrão maior, o que significa que as notas dos alunos nesta turma variaram mais amplamente. Isso poderia sugerir que, embora a média das notas tenha sido ligeiramente menor para a turma PAPEL, houve uma maior inconsistência nos resultados dos alunos quando a avaliação foi feita com papel e caneta.

Por outro lado, a turma IDE teve um desvio padrão menor, indicando que as notas dos alunos foram mais consistentes quando a avaliação foi feita no computador. Essas observações podem ser úteis para entender como diferentes métodos de avaliação podem impactar não apenas a média das notas dos alunos, mas também a consistência dessas notas.

Esses resultados, no entanto, são apenas indicativos e podem não ser conclusivos. Há muitos fatores que podem influenciar o desempenho dos alunos em uma avaliação, e a diferença nas médias que observamos pode não ser inteiramente devido ao formato da avaliação. Além disso, a diferença nas médias é relativamente pequena, então seria útil realizar mais pesquisas ou coletar mais dados para confirmar se essa diferença é estatisticamente significativa.

Com base na experiência da execução do experimento, foi possível concluir alguns pontos sobre o uso de IDE ou Papel na avaliação de alunos. Aqui estão algumas considerações:

- **Uso de IDE:**
  - Prática realista: Usar uma IDE para avaliações pode proporcionar uma experiência mais próxima da programação no mundo real. Os programadores geralmente usam IDE's ou editores de texto avançados em seu trabalho diário, que fornecem recursos como destaque de sintaxe, autocompletar e depuração.
  - Eficiência: As IDE's podem tornar a programação mais eficiente e menos propensa a erros, especialmente para programas mais complexos. Elas também permitem que os alunos testem e depurem seu código, o que pode ser uma habilidade importante para aprender.
  - Avaliação de código funcional: Se o professor está interessado em avaliar não apenas a capacidade dos alunos de escrever código, mas também de escrever código que compila e funciona corretamente, uma IDE seria necessária.
- **Papel e caneta:**
  - Foco na lógica: Escrever código à mão pode forçar os alunos a se concentrarem mais na lógica e na estrutura do código, já que eles não podem depender de recursos de autocompletar ou depuração. Isso pode ser útil para avaliar a compreensão dos alunos sobre os conceitos fundamentais de programação.
  - Prevenção de plágio: Em um ambiente de teste, o uso de papel e caneta pode tornar mais difícil para os alunos copiarem código uns dos outros ou de fontes online.

- Condições de teste: Em algumas situações, pode ser mais fácil administrar e monitorar um teste escrito à mão, especialmente se houver preocupações sobre a trapaça ou se os recursos tecnológicos forem limitados.

Em última análise, a escolha entre uma 'IDE' e 'papel e caneta' pode depender do que você deseja avaliar. Se o objetivo é avaliar a compreensão dos alunos sobre a lógica de programação e a estrutura do código, um teste escrito à mão pode ser adequado. Se o objetivo é avaliar as habilidades dos alunos em um ambiente de programação do mundo real, incluindo a capacidade de escrever, testar e depurar código, então uma avaliação baseada em IDE pode ser mais apropriada.

5.1 AMEAÇAS A VALIDADE:

A validade de um experimento refere-se à medida em que ele reflete com precisão o que pretende medir. Existem várias ameaças potenciais à validade deste experimento:

1. Validade Interna: A validade interna refere-se à medida que podemos atribuir causalidade nos resultados do experimento. Neste caso, estamos interessados em saber se o formato da avaliação (IDE versus papel e caneta) causou a diferença nas notas dos alunos. No entanto, existem várias variáveis de confusão potenciais que podem ameaçar a validade interna. Por exemplo, os professores envolvidos no experimento podem, inconscientemente, ter beneficiado um grupo em detrimento do outro. Além disso, as diferenças individuais entre os alunos, como seu estilo de aprendizado e nível de habilidade, também podem influenciar os resultados.
2. Validade Externa: A validade externa refere-se à medida que os resultados do experimento podem ser generalizados para outras situações. Se a amostra de alunos não for representativa da população geral de estudantes de programação, isso pode limitar a validade externa do experimento. Além disso, as condições específicas sob as quais o experimento foi realizado também podem limitar a generalização dos resultados.
3. Validade de Construção: A validade de construção refere-se à medida que o experimento mede o conceito ou construto que pretende medir. Neste caso, o experimento pretende medir o desempenho dos alunos em programação. No entanto, o desempenho em uma avaliação pode ser influenciado por muitos fatores além da habilidade de programação, como habilidades de teste, nível de preparação e até mesmo o estado emocional do aluno no dia do teste. Portanto,

as notas da avaliação podem não refletir com precisão a habilidade de programação dos alunos.

4. Validade de Conclusão: A validade de conclusão estatística refere-se à medida que as conclusões estatísticas obtidas a partir do experimento são válidas. Neste caso, a diferença nas médias observadas é relativamente pequena, e sem uma análise estatística adequada, não podemos dizer se essa diferença é significativa ou se poderia ter ocorrido por acaso.

Portanto, embora este experimento forneça alguns *insights* interessantes, é importante interpretar os resultados com cautela, dadas estas potenciais ameaças à validade.

6 CONSIDERAÇÕES FINAIS

Este trabalho consistiu em um experimento para comparar o desempenho dos alunos em programação quando submetidos a diferentes métodos de avaliação: uma avaliação realizada em uma IDE e uma avaliação realizada com papel e caneta. A planilha de notas de duas turmas, IDE e PAPEL, foi utilizada como base para a análise.

A diferença nas médias observadas neste experimento é relativamente pequena e não se pode dizer que essa diferença é significativa ou se poderia ter ocorrido por acaso. Portanto, embora os resultados deste experimento sejam úteis, eles devem ser interpretados com cautela. Seria benéfico realizar mais pesquisas, possivelmente com um desenho experimental mais rigoroso, para explorar mais a fundo a questão de como o formato da avaliação afeta o desempenho dos alunos em programação.

Os trabalhos futuros desta pesquisa podem se concentrar em expandir e aprofundar nosso entendimento sobre como diferentes métodos de avaliação impactam o desempenho dos alunos em programação. Isso pode incluir a realização de estudos de replicação para confirmar os resultados deste experimento inicial em diferentes grupos de alunos ou contextos educacionais.

Além disso, seria interessante explorar outros métodos de avaliação, como avaliações baseadas em projetos ou avaliações de pares. Investigar o impacto de variáveis individuais, como o estilo de aprendizado dos alunos, a familiaridade com a programação e a preferência por um método de avaliação, também pode fornecer insights valiosos.

Outro caminho seria desenvolver intervenções pedagógicas com base nos resultados da pesquisa e avaliar seu impacto no aprendizado dos alunos. Estudos longitudinais poderiam ser realizados para avaliar o impacto a longo prazo dos diferentes

métodos de avaliação. Além disso, seria interessante realizar estudos comparativos entre diferentes linguagens de programação ou níveis de complexidade de problemas de programação usando diferentes métodos de avaliação. Esses trabalhos futuros podem contribuir para uma compreensão mais abrangente e aprimorada sobre como otimizar a avaliação em cursos de programação, visando melhorar o aprendizado e o desempenho dos alunos.

7 REFERÊNCIAS

ARELLANO, N. E. **Canada needs 182,000 people to fill these IT positions by 2019.** IT World Canada. Retrieved September 19, 2016 from http://www.itworldcanada.com/article/canada-needs-182000-people-to-fill-these-it-positions-by-2019/287535. (2015).

ALMEIDA, T. O. **Laboratório Remoto de Robótica como Apoio ao Ensino de Programação,** Universidade Federal do Amazonas Instituto de Computação Programa de Pós-Graduação em Informática. (2016).

AMARAL, M. A.; BRESSAN, M. L. Q. **Avaliando a Contribuição do SCRATCH Para a Aprendizagem Pela Solução de Problemas e o Desenvolvimento do Pensamento Criativo** (2015)**.**

AZEREDO, P. A. **"Uma proposta de Plano Pedagógico para a Matéria de Programação"**, (2000).

ANDERSON, E. **Here are the workers most in demand in the uk. Telegraph Media Group Limited.** Retrieved September 19, 2016 from http://www.telegraph.co.uk/finance/jobs/11602670/Here-arethe-workers-most-in-emand-in-the-UK.html. (2015).

BIGGS, J. **Aligning teaching and assessing to course objectives.** ***Teaching and learning in higher education.*** (2003).

BALLESTER, M, et al. **Avaliação como apoio à aprendizagem.** Porto Alegre: Editora Artmed, (2003).

BENNEDSEN, Jens; CASPERSEN, M. E. **Assessing process and product: A practical lab exam for an introductory programming course.** *Innovation in Teaching and Learning in Information and Computer Sciences* (2007).

BESSA, Bruno; CUNHA, Mônica; FURTADO, Felipe. **ENGSOFT: Ferramenta para Simulação de Ambientes Reais para auxiliar o Aprendizado Baseado em Problemas (PBL) no Ensino de Engenharia de Software**. XX Workshop sobre Educação em Computação (WEI). Curitiba, PR. (2012).

BRIKMAN, Yevgeniy. **Don't learn to code. Learn to think**. Disponível em: <http://www.ybrikman.com/writing/2014/05/19/dont-learn-to-code-learn-to-think/>. Acessado em: 18 dez. 2021. (2014).

BROWN, M. **Exploring algorithms using BALSA-II.** IEEE Computer. V 21 No. 5. (1988).

BROWN, M.H. **Zeus: A System for Algorithm Animation and Multi-view Editing,** Research Report n. 75, DEC Systems Research Center, Palo Alto, CA. (1991).

BARROS, J. P. **Students' Perceptions of Paper-Based vs. Computer-Based Testing in an Introductory Programming Course** Polytechnic Institute of Beja, Beja, Portugal UNINOVA-CTS, Monte de Caparica, Portugal (2018).

BASILI, V. R., CALDIERA, G. e ROMBACH, H. D. **Goal/question/metric approach**. In: MARCINIAK, J. (ed) Encyclopedia of software engineering. New York: John Wiley & Sons, v. 1, p.528-532, (1994).

CHADWICK, Clifton; OLIVEIRA, J. B. A. **Aprender e Ensinar.** Belo Horizonte: Editora Alfa Educativa: 8a Ed., (2007).

CARDOSO, S. M. V.; JANDL, Peter. **Estilos de Aprendizagem: Aprender a Aprender**. (1998).

CHAMILLARD, A.; JOINER, J. K**. Using lab practica to evaluate programming ability.** In *ACM SIGCSE Bulletin* (2001).

CASTRO, C. T.; CASTRO JUNIOR, A.; MENESES, C., B., M. e RAUBER, M. **Utilizando Programação Funcional em Disciplinas Introdutórias de Computação**, In: *XI Workshop de Educação em Computação – WEI, Campinas/SP,* (2003).

CAVELLUCCI, L.C. B. **Estilos de Aprendizagem**: em busca das diferenças individuais. Curso de Especialização em Instrucional Design, 2005.Site Educacional.

COHEN, L.; MANION, L.; MORRISON, K**. Research methods in education**. Routledge Falmer, 2000.

CAMPBELL, D. T. e STANLEY, J. C. **Experimental and quasi-experimental designs for research**. Houghton Mifflin Company, Boston, MA, (1963).

DALY, Charlie; WALDRON, John. **Assessing the Assessment of Programming Ability.** (2004).

ENGLISH, J. **Experience with a computer-assisted formal programming examination.** In ACM SIGCSE bulletin (Vol. 34, pp. 51–54). ACM. (2002).

FERNANDES, Elisângela; SANTOMAURO, Beatriz. **Aula expositiva: o professor no centro das atenções.** (2011).

FAVERO, E.L; HARB, M.P.A.H; LINO, A. P.; SANTOS, T.L.T; SILVA, A. S.; SANTOS, T.L.T. **Avaliação automática de questões conceituais discursivas.** TIL´. Congresso da Sociedade Brasileira de Computação (submetido). (2007).

FALKEMBACH, G. A. M. **O lúdico e os jogos educacionais.** In: Mídias na Educação. CINTED, UFRGS. (2007).

FRANÇA, V. F. de; NORONHA, R. V.; OMAR, Nizam; PIMENTEL, E. P. **Avaliação Contínua da Aprendizagem, das Competências e Habilidades em Programação de Computadores.** (2003).

GROVER, Shuchi; PEA, Roy. **Computational Thinking in K-12: A Review of the State of the Field**. Educational Researcher, (2013).

GOMES, A.; MENDES, A. J. **Learning to program - difficulties and solutions.** International Conference on Engineering Education (ICEE), (2007).

GNATZ, M. et al. **A Practical Approach of Teaching Software Engineering**. Proceedings of the 16th Conference on Software Engineering Education and Training (CSEET'03). Madrid: IEEE. p. 120-128. (2003).

GOOLD, A.; HORAN, P. (2002**) Foundation software engineering practices for capstone projects and beyond**. Proc. 15th Conference on Software Engineering Education and Training (CSEE&T 2002), IEEE CS Press, pp 140-146, (2002).

HOED, R. M. **Análise da evasão em cursos superiores: o caso da evasão em cursos superiores da área de Computação,** Universidade de Brasília Instituto de Ciências Exatas Departamento de Ciência da Computação (2016).

HAGHIGHI, P. D; IKEDA, M.; JONASSEN, D.; LOOI, C.-K & SHEARD, J. **Summative computer programming assessment using both paper and computer.** (2005**).**

HAYDT, R. C. **Avaliação do processo Ensino-Aprendizagem**. São Paulo: Editora Ática: 6a ed., (2007).

JACOBSON, N. **Using on-computer exams to ensure beginning students' programming competency.** ACM SIGCSE Bulletin, 32(4), 53–56. (2000).

JANDL, Peter Jr.; ZUCHINI, Márcio H. **IC: Um Interpretador de Linguagem C.** IN Projeções - USF, Bragança Paulista, V. 17, pp. 101-112. (1999).

GOMES, A.; MENDES, A. **Suporte à Aprendizagem da Programação com o Ambiente SICAS.** Actas do V Congresso Ibero-americano de Informática Educativa, Viña del Mar, Chile. (2000).

KURNIA, Andy; LIM, Andrew; CHEANG, Brenda. **Online Judge. Computer & Education**, (2001).

KOILE, K.; SINGER, D. **Improving learning in CS1 via tablet-PC-based in-class assessment.** ICER '06: Proceedings of the Second International Workshop on Compunting Education Research (pp. 119-126). ACM. Canterbury, UK. (2006)

VICTORINO, Luis; ELIA, Marcos; GOMES, Ângelo; CASTRO, Marcos de; & BASTOS, César. **"Laboratório Virtual de Atividades Didáticas"** (2009).
KAY, D. G.; SCOTT, T.; ISAACSON, P.; REEK, K. A. **"Automated Grading Assistance for Student Programs".** ACM SIGCSE Bulletin, vol 26, is. 1, p. 381-382. (1994).

KJOLLERSTROM, B.; MARTENSSON, M. **"Assessment: The Key to Changing the Way We Learn**". CAL-Laborate. (1999).

MASETTO, Marcos. **Didática: A aula Como Centro**. São Paulo: Editora FTD S. A, (1997).

MARQUES, M.; OCHOA, S; QUISPE, A. **A Systematic Mapping Study on Practical Approaches to Teaching Software Engineering**. Frontiers in Education Conference. Madrid: IEEE. (2014).

MANNILA, L. **Novices' Progress in Introductory Programming Courses.** (2007).

MORATORI, P. B. "**Por que utilizar jogos Educativos no Processo de Ensino Aprendizagem?",** Universidade Federal do Rio de Janeiro - Instituto de Matemática - Núcleo de Computação Eletrônica - Informática da Educação. (2003).

MENEZES, N. N. C. **Introdução à Programação com Python: Algoritmos e lógica de programação para iniciantes**. (2014).

MOREIRA, M. P; FAVERO, E. L. **Um Ambiente para Ensino de Programação com *Feedback* Automático de Exercícios,** Programa de Pós-Graduação em Ciência da Computação (PPGCC) – Centro de Ciências Exatas e Naturais (CCEN) – Universidade Federal do Pará (UFPA) Rua Augusto Corrêa, No. 1 – 66075-110 – Belém – PA – Brasil. (2009).

MAIA, M. de C; MENDONÇA, A. L; GOES, Paulo. **Metodologia de Ensino e Avaliação de Aprendizagem.** (2005).

NOURI, Jalal; OQVIST, Martina. **Coding by hand or on the computer? Evaluating the effect of assessment mode on performance of students learning programming (2018).**

NOVAK, J. D. **Meaningful learning: The essential factor for conceptual change in limited or inappropriate propositional hierarchies leading to empowerment of learners.** (2002).

NETO, W. C. B.; SCHUVARTZ, A. A. **Ferramenta Computacional de Apoio ao Processo de Ensino-Aprendizagem dos Fundamentos de Programação de Computadores,** (2007).

NUNES, D. J., YAMAGUTI, M. H. E NUNES, I. **Refinement of student competences of the software engineering course**. IX Forum on Education in Software Engineering, p. 143-146, (2016).

OLIVEIRA, A. C; **Gamificação da Educação.** Universidade do Estado da Bahia. (2015).

PRIETCH, S. S; PAZETO, T. A. **Estudo sobre a Evasão em um Curso de Licenciatura em Informática e Considerações para Melhorias,** Curso de Licenciatura Plena em Informática – Universidade Federal de Mato Grosso (UFMT) (2010).

PRIOR, J. C. **"Online assessment of SQL query formulation skills"**. In Proceedings of the Fifth Australasian Conference on Computing Education. Adelaide, Australia. (2003).

PIERSON, W. C. et al. **WEB-based Animation of Data Structures Using JAWAA.** (1998).

PEREIRA, P. S.; et. al **Análise do *Scratch* como ferramenta de Auxílio ao Ensino de Programação de Computadores.** Instituto Federal de Educação, Ciência e Tecnologia do Ceará, Fortaleza – CE. Disponível em <www.abenge.org.br/CobengeAnteriores/2012/artigos/104281.pdf>. Acesso em 18 Dezembro 2021. (2012**).**

RAABE, A. L. A; SILVA, J. M. C, da. **Um Ambiente para Atendimento as Dificuldades de Aprendizagem de Algoritmos**, Ciência da Computação – Universidade do Vale do Itajaí (UNIVALI) (2005).

RAJALA, T., KAILA, E., LINDE'N, R., KURVINEN, E., LOKKILA, E., LAAKSO, M.-J., et al. **Automatically assessed electronic exams in programming courses.** In Proceedings of the Australasian computer science week multiconference (2016).

ROCHA, P. S. **Ensino e Aprendizagem de Programação: Análise da Aplicação de Proposta Metodológica Baseada no Sistema Personalizado de Ensino.** (2010).

ROBINS, A.; ROUNTREE, J.; ROUNTREE, N. **Learning and Teaching Programming: A Review and Discussion.** (2003).

SAIKKONEN R., MALMI L., & KORHONEN, A. **Fully automatic assessment of programming exercises.** In: Proc. 6th Annu. Conf. on Innovation and Technology in Computer Science Education, (Canterbury, 2001) pp. 133–136.

SIMPSON, I. **Developers in demand — tech talent doomsday.** Clearcode. Retrieved September 19, 2016 from: http://clearcode.cc/2016/01/developers-in-demand-tech-talent-doomsday/. (2016).

SHULL, F.; WANGENHEIM, C. G. V. **To Game or Not to Game?,** IEEE Software. (2009).
SALVADOR, C.C. **Aprendizagem escolar e construção do conhecimento na escola.** Porto Alegre, Artes Médicas, (1994).

SCHIMIGUEL, J. et al. **Desenvolvimento de Simulações para o Aprendizado em cursos na Web**. In: 3rd INTERNATIONAL CONFERENCE ON ENGINEERING AND COMPUTER EDUCATION.(2003).

STERN, L. et al. **A Strategy for Managing Content Complexity in Algorithm Animation.** In: SIGCSE BULLETIN – THE 4° ANNUAL SISCSE/SIGCUE CONFERENCE ON INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION.(1999).

TOBAR, Carlos M. et al. **Uma Arquitetura de Ambiente Colaborativo para o Aprendizado de Programação.** XII Simpósio Brasileiro de Informática na Educação, Vitória – ES, (2001).

TAROUCO, L. M. R; ROLAND, L. C; FABRE, Marie-Christine J. M; KONRATH, M. L. P. "**JOGOS EDUCACIONAIS".** CINTED/UFRGS (2004).

VAN HEUR, R. **Fears of software skills shortage in Germany and the Netherlands**. Computer Weekly, TechTarget. Retrieved September 19, 2016 from http://www.computerweekly.com/news/4500269840/Fears-of-software-skills-shortage-in-Germany-and-the-Netherlands. (2016).

VAN MERRIENBOER, J.J.G.; KRAMMER, H.P.M. **Instructional strategies and tactics for the design of introductory computer programming courses in high school.** (1987).

APÊNDICE A

Título do Projeto: Projeto SERDE (Software Engineering: Research, Development, and Education).
Coordenador do Projeto: Julio Cezar Costa Furtado
Instituição: Universidade Federal do Amapá
Email de Contato: furtado@unifap.br
Telefone de Contato: (96) 98114-7118
Título da Pesquisa: Papel ou Ide: Uma Análise Sobre as Formas de Avaliação em Turmas Iniciantes de Programação.
Tipo de Pesquisa: Experimento em Sala de Aula

1) Elabore um programa em C que calcule o que deve ser pago por um produto considerando o preço normal de etiqueta e a escolha da condição de pagamento. Utilize os códigos abaixo para ler qual a condição de pagamento escolhida e efetuar o cálculo adequado:
    1. À vista em dinheiro ou cheque, recebe 10% de desconto
    2. À vista no cartão de crédito, recebe 15% de desconto
    3. Em duas vezes, preço normal de etiqueta sem juros
    4. Em duas vezes, preço normal de etiqueta mais juros de 10

2) Escreva um programa em C que leia o número de identificação, as 3 notas obtidas por um aluno nas 3 verificações e a média dos exercícios que fazem parte da avaliação, e calcule a média de aproveitamento, usando fórmula:

$$MA = (nota1 + nota2 * 2 + nota3 * 3 + ME)/7$$

A atribuição dos conceitos obedece a tabela abaixo. O algoritmo deve escrever o número do aluno, suas notas, a média dos exercícios, a média de aproveitamento, o conceito corresponde e a mensagem 'Aprovado' se o conceito for A, B ou C, e 'Reprovado' se o conceito for D ou E. Abaixo é apresentado os intervalos de média para obtenção de cada conceito:

1. MA >= 90, Conceito A
2. MA >= 75 e MA < 90, Conceito B
3. MA >= 60 e MA < 75, Conceito C
4. MA >= 40 e MA < 60, Conceito D
5. MA < 40, Conceito E

APÊNDICE B

IDE					
ID	Q1	Q2	Média	Soma	Escala
1	90	40	65,00	130,00	65,00%
2	100	30	65,00	130,00	65,00%
3	70	70	70,00	140,00	70,00%
4	15	0	7,50	15,00	7,50%
5	25	0	12,50	25,00	12,50%
6	100	20	60,00	120,00	60,00%
7	85	0	42,50	85,00	42,50%
8	0	60	30,00	60,00	30,00%
9	20	0	10,00	20,00	10,00%
10	40	0	20,00	40,00	20,00%
11	45	0	22,50	45,00	22,50%
12	20	0	10,00	20,00	10,00%
13	45	95	70,00	140,00	70,00%
14	95	0	47,50	95,00	47,50%
15	25	10	17,50	35,00	17,50%
16	25	10	17,50	35,00	17,50%
17	30	0	15,00	30,00	15,00%
18	30	0	15,00	30,00	15,00%
19	40	70	55,00	110,00	55,00%
20	0	25	12,50	25,00	12,50%
Média	45,00	21,50	33,25	66,50	66,50%
Moda	25	0	65	130	65,00%
PAPEL					
ID	Q1	Q2	Média	Soma	Escala
1	85	85	85,00	170,00	85,00%
2	75	75	75,00	150,00	75,00%
3	60	20	40,00	80,00	40,00%
4	70	100	85,00	170,00	85,00%
5	55	0	27,50	55,00	27,50%
6	55	0	27,50	55,00	27,50%
7	100	0	50,00	100,00	50,00%
8	90	25	57,50	115,00	57,50%
9	65	0	32,50	65,00	32,50%
10	20	10	15,00	30,00	15,00%
11	95	30	62,50	125,00	62,50%
12	15	10	12,50	25,00	12,50%
13	0	0	0,00	0,00	0,00%
14	5	25	15,00	30,00	15,00%
15	0	0	0,00	0,00	0,00%
16	0	0	0,00	0,00	0,00%
17	0	0	0,00	0,00	0,00%
18	0	0	0,00	0,00	0,00%
19	0	0	0,00	0,00	0,00%
Média	41,58	20,00	30,79	61,58	61,58%
Moda	0	0	0	0	0,00%

teste Mann-Whitney U

O valor de U é 167 .

Explicação dos Resultados

Como você pode, sem dúvida, ver, esta calculadora cospe muitas informações. A maior parte é auto-explicativa, mas há algumas coisas que vale a pena observar.

Primeiro, não há uma maneira padrão para o teste U de Mann-Whitney lidar com classificações empatadas, o que significa que, se seus dados tiverem classificações empatadas, você obterá um resultado diferente para U , dependendo do pacote de estatísticas que usar (para uma discussão de algumas das questões que isso levanta, veja este artigo , por exemplo).

Em segundo lugar, onde o número de pontuações (ou seja, o valor de N) em cada amostra é 10 ou mais, você pode assumir que sua distribuição amostral é aproximadamente normal. Isso significa que você pode usar uma razão Z para calcular o valor de p.

Amostra 1	Amostra 2	*Valores S1*	*Classificações S1*	*Valores S2*	*Classificações S2*	
65.00	85.00	7.5	7	0	3.5	
65.00	75.00	10	8.5	0	3.5	
70.00	40.00	10	8.5	0	3.5	
7.50	85.00	12.5	11	0	3.5	
12.50	27.50	12.5	11	0	3.5	
60.00	27.50	15	14.5	0	3.5	
42.50	50.00	15	14.5	12.5	11	
30.00	57.50	17.5	17.5	15	14.5	
10.00	32.50	17.5	17.5	15	14.5	
20.00	15.00	20	19	27.5	21.5	
22.50	62.50	22.5	20	27.5	21.5	
10.00	12.50	30	23	32.5	24	
70.00	0.00	42.5	26	40	25	
47.50	15.00	47.5	27	50	28	
17.50	0.00	55	29	57.5	30	
17.50	0.00	60	31	62.5	32	
15.00	0.00	65	33.5	75	37	
15.00	0.00	65	33.5	85	38.5	
55.00	0.00	70	35.5	85	38.5	
12.50		70	35.5			

Detalhes do resultado

Amostra 1
Soma das classificações: 423
Média das classificações: 21,15
Soma esperada das classificações: 400
Média esperada das classificações: 20
Valor
U : 167 Valor U
esperado : 190

Amostra 2
Soma das classificações: 357
Média das classificações: 18,79
Esperado soma das classificações: 380
Média esperada das classificações: 20
Valor
U : 213 Valor U
esperado : 190

Amostra 1 e 2
Soma combinada das classificações:
780
Média das classificações: 20
Desvio padrão: 35,5903

Resultado 1 - Valor U

O valor U é 167 . O valor crítico de U em $p < 0,05$ é 130. Portanto, o resultado não é significativo em $p < 0,05$.

Resultado 2 - Razão Z

O Z -Score é 0,6322 . O valor- p é 0,26435. O resultado não é significativo em $p < 0,05$.