UNIVERSIDADE ESTADUAL DA PARAÍBA

CAMPUS VII – GOVERNADOR ANTÔNIO MARIZ

CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

WANDERLEY PEREIRA DE SOUSA

COMPUTAÇÃO DESPLUGADA NO APOIO AO ENSINO DE ESTRUTURA DE DADOS

PATOS – PB

2019

WANDERLEY PEREIRA DE SOUSA

COMPUTAÇÃO DESPLUGADA NO APOIO AO ENSINO DE ESTRUTURA DE DADOS

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Estadual da Paraíba, como requisito Parcial à obtenção do título de Graduação em Bacharelado em Ciência da Computação.

Área de concentração: Computação e Educação

Orientador: Prof. Esp. Fábio Júnior Francisco da Silva

PATOS – PB

2019

S725c Sousa, Wanderley Pereira de.

Computação desplugada no apoio ao ensino de estrutura de dados [manuscrito] / Wanderley Pereira de Sousa. - 2019.

54 p. : il. colorido.

Digitado.

Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências Exatas e Sociais Aplicadas , 2019.

"Orientação : Prof. Esp. Fábio Júnior Francisco da Silva , Coordenação do Curso de Computação - CCEA."

1. Computação desplugada. 2. Ensino de estrutura de dados. 3. Listas encadeadas. I. Título

21. ed. CDD 004

Elaborada por José E. da S. Eugênio - CRB - 15/591 BSC7/UEPB

Wanderley Pereira de Sousa

COMPUTAÇÃO DESPLUGADA NO APOIAO AO ENSINO DE ESTRUTURA DE DADOS

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciências da Computação da Universidade Estadual da Paraíba, em cumprimento à exigência para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 27/11/2019

BANCA EXAMINADORA

Fábio Júnior Francisco da Silva

Prof. Esp. Fábio Júnior F. da Silva
(Orientador)

Jefferson Felipe Silva de Lima

Prof. Me. Jefferson Felipe Silva de Lima
(Examinador)

Ingrid Morgane M. de Lucena

Prof. Me. Ingrid Morgane M. de Lucena
(Examinadora)

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me prover a capacidade e a força necessária para sempre persistir e perseverar diante das adversidades, à Ele dedico conclusão deste importante ciclo em minha vida.

A toda minha família por me apoiar e sempre acreditar no meu potencial, em especial a minha mãe e meu pai pelos ensinamentos e valores que me ensinaram ao longo de minha vida e por não medirem esforços para me ajudar durante esta jornada. As minhas Wandikerlla e Weslania por sempre me apoiarem. A minha namorada Rubea por sempre estar ao meu lado em todos os momentos, pelo incentivo e pelo apoio durante todo o processo, sempre acreditando em mim, não me deixando desanimar nas dificuldades.

Agradeço ao meu orientador Prof. Esp. Fábio Júnior por toda sua dedicação e por todos os ensinamentos passados durante a condução desta pesquisa. Seu incentivo e sua paciência foram fundamentais para a conclusão deste trabalho.

A todos os meus amigos e colegas que estiveram comigo durante toda a graduação, em especial a Valter Gomes e Theogenes Nunes pela amizade e companheirismo compartilhados durante essa jornada.

A todos os professores do Curso de Ciência da Computação da UEPB Campus VII, que contribuíram para minha formação profissional, pelos conhecimentos e experiências compartilhadas e pelas amizades adquiridas.

Por fim, a todos que apoiaram, acreditaram em mim e contribuíram de forma direta ou indireta para a conclusão desta jornada, meus mais sinceros agradecimentos.

RESUMO

Nos cursos relacionados a área de computação, estudantes têm enfrentado dificuldades nas disciplinas que requerem habilidades como abstração, raciocínio lógico e desenvolvimento de algoritmos. Estas dificuldades têm causado um mal aproveitamento dos discentes nestas disciplinas, além de altos índices de evasão. Isso tem tornado o processo de ensino e aprendizagem de disciplinas como estas, um desafio, tanto para os professores quanto para os estudantes. Abordagens e metodologias lúdicas de ensino de programação e estrutura de dados têm sido criadas com o objetivo de amenizar os problemas enfrentados pelos estudantes em contato com estes conteúdos, percebeu-se, portanto, uma oportunidade de explorar propostas lúdicas de ensino para conceitos computacionais que ainda não foram bem explorados pela comunidade acadêmica. Diante disso, este trabalho tem como objetivo apresentar uma abordagem didática para apoiar o ensino de estrutura de dados do tipo lista. A abordagem tem como base metodologias ativas, como a computação desplugada, e utiliza materiais manipuláveis e dinâmicas de grupo para representar em sala de aula o funcionamento de uma lista dinâmica simplesmente encadeada e suas operações. Ela tem como diferencial a utilização de materiais de baixo custo e de fácil confecção, além de elementos presentes em sala de aula e no cotidiano dos estudantes. Espera-se que com a utilização de metodologias como esta, seja possível fazer com que os estudantes possam desenvolver as capacidades inerentes ao aprendizado destes conteúdos e participem de forma mais efetiva no processo de aquisição de conhecimento.

Palavras-chave: Computação desplugada. Ensino de estrutura de dados. Listas encadeadas.

ABSTRACT

In courses related to computing area, students have faced difficulties in subjects that require skills such as abstraction, logical reasoning and algorithm development. These difficulties have caused students to make poor use of these subjects, as well as high dropout rates. This has made the process of teaching and learning such subjects challenging for both teachers and students. Playful approaches and methodologies of programming and data structure teaching have been created in order to alleviate the problems faced by students in contact with these contents, therefore, it was perceived an opportunity to explore playful teaching proposals for computational concepts that still not well explored by the academic community. Given this, this paper aims to develop a didactic approach to support the teaching of list-type data structure. The approach is based on active methodologies, such as unplugged computing, and utilizes manipulable materials and group dynamics to represent in the classroom the functioning of a simply chained dynamic list and its operations. Its differential is the use of low cost and easy to make materials, in addition to elements present in the classroom and students' daily life It is hoped that with the use of methodologies such as this, it will be possible for students to develop the skills inherent in learning these contents and to participate more effectively in the process of knowledge acquisition.

Keywords: Unplugged computing. Data structure teaching. Chained lists.

LISTA DE FIGURAS

LISTA DE ABREVIATRAS E SIGLAS

AUX	Auxiliar
CL	Cabeça de Lista
ED	Estrutura de Dados
FIFO	*Last In First Out*
LIFO	*Last In First Out*
MEC	Ministério da Educação
PROX	Próximo
UEPB	Universidade Estadual da Paraíba

SUMÁRIO

1 INTRODUÇÃO

Diante da crescente influência que o avanço tecnológico tem exercido nos mais variados contextos socioeconômicos, a Ciência da Computação vem a cada dia ganhado visibilidade e importância em nossa sociedade. De modo que o profissional que deseja atuar de forma efetiva em sua área, necessita possuir algum nível de conhecimento e habilidade relacionadas a computação. Além disso, a busca por profissionais capazes de desenvolver soluções tecnológicas tem se tornado cada vez maior no mercado atual, fazendo com que os cursos de tecnologia ou graduação na área de computação, tenham um importante papel na formação destes profissionais. Dentre os cursos reconhecidos MEC (2016) podemos destacar os cursos de Ciência da Computação, Engenharia da Computação, Engenharia de Software e Sistemas de Informação. Nesse contexto, uma habilidade muito requerida é o domínio da programação de computadores.

O ensino de tópicos como algoritmos, lógica de programação e estrutura de dados (ED), estão presentes desde os estágios iniciais dos cursos da área de tecnologia, sendo estes uma base importante e indispensável na formação destes profissionais. Segundo Silva (2017), aprender programação melhora o desenvolvimento do raciocínio lógico e a capacidade de abstração, apoiando no processo de obtenção de habilidades como resolução de problemas e noções de causa e efeito. Isso faz com que este tipo de aprendizado se torne importante, não só para os estudantes dos cursos de tecnologia, mas também para todos os tipos de pessoas, pois desenvolve habilidades que podem auxiliar na resolução de problemas dos mais diversos contextos sociais e econômicos.

Em países como Estados Unidos e Estónia, o ensino de programação já é adotado desde a educação básica. No Brasil, o aprendizado destes conceitos geralmente é reservado para os cursos de graduação da área ou similares, de forma que os estudantes ao adentrarem nestes cursos, passam por um processo de adaptação para poderem desenvolver as habilidades necessárias para o aprendizado destes conteúdos (SILVA et al., 2015).

Segundo Belizário et al. (2016), o desenvolvimento de algoritmos e programas está entre as competências mais difíceis de serem adquiridas, o que faz com que, muitas vezes, os estudantes em contato com estas disciplinas encontrem dificuldades em lidar com certas práticas e conhecimentos como abstração, lógica e a programação de computadores, fazendo com que, em muitos casos, essas dificuldades resultem em altos índices de evasão dos cursos de computação (MOREIRA; MONTEIRO, 2018). Diante disso, Santiago e Kronbauer (2017)

afirma que, cerca de 40% dessas desistências acontecem já nas matérias introdutórias da área, presentes nos primeiros semestres do curso. Para Barbosa et al. (2011), o fator motivacional é um dos principais responsáveis pelo mal aproveitamento dos estudantes nestas disciplinas, e dos altos índices de evasão nestes cursos. Santiago e Kronbauer (2017) argumenta que essa problemática tem sido alvo de discussão de diversos estudos recentes, que apontam uma preocupação com este elevado nível de desistências nestes cursos.

A disciplina de Estrutura de Dados (ED), faz parte da base curricular dos cursos da área de computação, de forma que, os conhecimentos sobre algoritmos e programação servem de base para o seu bom entendimento, desta forma, é seguro afirmar que as mesmas dificuldades enfrentadas pelos estudantes nos conteúdos de algoritmos e programação, tendem a permanecer na disciplina de ED. Com isso, o processo de ensino e aprendizagem de disciplinas como esta, torna-se um desafio, tanto para os professores quanto para os estudantes.

Abordagens e metodologias lúdicas de ensino de ED e programação, como as descritas em Santiago e Kronbauer (2017), Souza et al. (2011), Moreira e Monteiro (2018), Costa et al. (2017) e Voss et al. (2015), têm sido criadas com o objetivo de amenizar os problemas enfrentados pelos estudantes nestas disciplinas. Com base nessa problemática, percebe-se que na literatura cientifica existem conceitos computacionais ainda pouco explorados por este tipo de abordagem, desta forma, foi possível identificar uma oportunidade para o desenvolvimento de abordagem lúdica que possam apoiar o processo de ensino e aprendizagem dentro da disciplina de ED.

Com base nos trabalhos de Moreira e Monteiro (2018), Costa et al. (2017) e Martinhago et al. (2017), identificou-se que a adoção de metodologias ativas como a computação desplugada, apresentam-se como uma alternativa para atuar como facilitador no processo de ensino e aprendizagem, minimizando as dificuldades expostas anteriormente. A partir deste contexto, foi criada neste trabalho uma abordagem didática (dinâmica), que pode ser utilizada na disciplina de ED para auxiliar no ensino de listas, pilhas e filas. A proposta faz uso de materiais manipuláveis e dinâmicas de grupo, apoiado nos conceitos advindos da computação desplugada, para representar em sala de aula o funcionamento de uma estrutura de dados do tipo lista dinâmica simplesmente encadeada.

1.1 Objetivo geral

Desenvolver uma abordagem didática para apoiar o ensino de estrutura de dados do tipo lista, utilizando computação desplugada.

1.2 Objetivos específicos

Para alcançar o objetivo geral deste trabalho definiu-se os objetivos específicos:

a. Investigar, na literatura, trabalhos que abordam o ensino estrutura de dados de forma lúdica, utilizando metodologias ativas como a computação desplugada e uso de materiais manipuláveis;
b. Elaborar material manipulável, de baixo custo, o qual possa ser utilizado no ensino de listas na componente curricular Estrutura de Dados;
c. Desenvolver abordagem didática com o material elaborado para o ensino de estrutura de dados do tipo lista.

1.3 Metodologia

A presente seção tem como objetivo descrever os procedimentos metodológicos empregados no desenvolvimento desta pesquisa. Com base nos objetivos específicos definidos anteriormente, são descritos os seguintes processos metodológicos: revisão bibliográfica, definição e criação dos materiais manipuláveis, desenvolvimento da proposta e descrição das operações através de ilustrações e códigos em linguagem de programação C.

A revisão bibliográfica teve como objetivo inicial alcançar um entendimento dos conceitos relacionados ao tema abordado, servindo de base para a fundamentação teórica, além de oferecer os parâmetros para delimitar o objetivo desta pesquisa. Através da revisão sobre o ensino de estrutura de dados e programação foi possível identificar e selecionar estudos sobre o desenvolvimento e utilização de ferramentas e metodologias lúdicas aplicadas a estas áreas de conhecimento. Tais estudos serviram como base para o desenvolvimento da abordagem didática proposta nesta pesquisa. Para a execução da revisão bibliográfica foram utilizados os mecanismos de busca Google Scholar e o Portal de Periódicos CAPES.

A execução das atividades descritas na abordagem proposta conta com a utilização de materiais manipuláveis que, seguindo a recomendação de Bell et al. (2009), devem ser acessíveis e de baixo custo. Tendo isso em mente foi sugerido a utilização de materiais

comumente encontrados em sala de aula, como cadeiras, lápis e papel. Além disso, alguns dos artefatos a serem utilizados foram ilustrados utilizando um *software* para criação de apresentações PowerPoint.

Após a definição dos materiais a serem utilizados, foi descrito como representar uma lista simplesmente encadeada utilizando o espaço de sala de aula, juntamente com os materiais sugeridos. Foram escolhidas 7 operações que podem ser efetuadas em uma lista encadeada, e para cada uma dessas operações foi criada uma representação em forma de dinâmica para serem praticadas pelos estudantes. Para descrever o funcionamento de cada uma destas operações foi utilizado a linguagem de programação C, por ser comumente utilizadas na disciplina de ED, para codificar as funções que realizam as operações utilizadas nesta abordagem, os códigos das funções apresentadas tiveram como base o livro "Estrutura de Dados descomplicada em linguagem C" (BACKES, 2017), para codificação das funções foi utilizada a ferramenta gratuita Code::Blocks IDE. Para facilitar o entendimento da dinâmica também foram desenvolvidas ilustrações que representam a execução de cada uma das operações representadas pelos estudantes. Para criar as ilustrações foram utilizados os *softwares* PowerPoint e o editor de imagens Paint.

1.4 Organização do trabalho

Este trabalho apresenta cinco seções e está organizado da seguinte forma: a primeira seção fornece uma visão geral dos principais temas envolvidos neste trabalho, além de uma contextualização acerca do problema, justificativa, proposta, objetivos e metodologia desta pesquisa; a segunda seção fornece a base teórica necessária para o entendimento dos temas relacionados a pesquisa e apresentação de trabalhos relacionados; a terceira seção descreve a proposta de abordagem didática para o ensino de estrutura de dados do tipo lista; na quarta seção são apresentadas as considerações finais da pesquisa, suas dificuldades, contribuições e sugestões para trabalhos futuros, e por fim, as referências e apêndices.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção aborda os conceitos que nortearam o desenvolvimento desta pesquisa, dos quais se fazem necessários para o entendimento dos temas abordados. O mesmo possui as seguintes subseções: Computação; Didática; Computação desplugada; Estrutura de dados; O ensino de estrutura de dados; Listas encadeadas e Trabalhos relacionados.

2.1 Computação

No decorrer da história, a humanidade buscou meios de minimizar o esforço repetitivo e tedioso do trabalho, desenvolvendo máquinas que passaram a substituir os homens em determinadas tarefas (FONSECA FILHO, 2007). Segundo Brookshear (2003), a busca por uma máquina que executasse tarefas algorítmicas vem desde épocas remotas, como nas antigas civilizações grega e romana que utilizavam o ábaco como dispositivo computacional primitivo para fazer cálculos matemáticos.

A computação é uma ciência muito recente se comparada a outras áreas do conhecimento, apesar de boa parte de sua base lógica e matemática já existirem há muito mais tempo, como a álgebra booleana criada no século XIX por George Boole, sua fundamentação veio justamente do esforço de dezenas de pensadores de diferentes culturas, para encontrar as melhores formas de usar símbolos, o que viabilizou o desenvolvimento da ciência, matemática e lógica, e acabou fundamentando toda a computação. Desde a segunda guerra mundial o avanço da computação foi exponencial, fazendo surgir diversas tecnologias, conceitos e ideias nunca antes imaginadas, que acabaram se tornando revolucionárias (FONSECA FILHO, 2007).

A evolução da computação, segundo Denning (2005), pode ser dividida em três estágios, o das ferramentas, iniciando na década de 1940, dos métodos, nos anos 80 e dos processos fundamentais, com início nos anos 2000. Na década de 1940, iniciava-se o desenvolvimento dos primeiros computadores digitais eletrônicos, a computação era vista como ferramenta usada para resolver equações, quebrar códigos de criptografia, analisar dados, gerenciar processos de negócios, executar simulações, entre outros problemas, que antes eram considerados intratáveis. Na década de 1980 a computação passou a se tornar indispensável em diversos campos de estudo, passando de uma ferramenta usada para explorar o conhecimento existente, a uma ciência capaz de prover a descoberta de inovações, usando a computação como método principal. Nos anos 2000 em diante a computação avançou ainda mais, estando presente

em diversos setores da sociedade, sendo utilizada por cientistas como biólogos, astrônomos e economistas, para prover o avanço de seus campos de estudo. Assim, com a popularização do uso do computador e o desenvolvimento de novas tecnologias de hardware e software, a Ciência da Computação se tornou cada vez mais multidisciplinar e se faz presente nas mais diversas áreas do conhecimento humano, apoiando e automatizando atividades, e criando novos negócios e inovações tecnológicas.

Martins (2019), define a Ciência da Computação como o estudo de técnicas, metodologias e instrumentos computacionais com o objetivo de automatizar processos e desenvolver soluções com o uso de processamento de dados. Nesse contexto de desenvolvimento tecnológico um cientista da computação deve ser o profissional capaz de manter e contribuir para o avanço da tecnologia da informação e comunicação, bem como da computação, na medida em que consegue combinar a computação com outras ciências, dando a estas um tratamento computacional (MEC, 2016).

2.2 Didática

A educação ou prática educativa é um fenômeno social e universal que se caracteriza como atividade necessária à existência e desenvolvimento da humanidade. É através da prática educativa que se torna possível a inclusão de indivíduos na sociedade, através do compartilhamento de conhecimentos e experiências que os tornam capazes de interagir e atuar no meio em que vivem (LIBÂNEO, 2006).

Para Zabala (1998, p. 18), a prática educativa pode ser compreendida como: “um conjunto de atividades ordenadas, estruturadas e articuladas para a realização de certos objetivos educacionais”, já Libâneo (2006, p.17), ressalta a importância da prática educativa afirmando que “não há sociedade sem prática educativa nem prática educativa sem sociedade”, então podemos afirmar que a prática educativa pode ser vista como um dos pilares de sustentação de nossa sociedade.

A didática compreende o estudo dos objetivos, conteúdos, meios e as condições do processo de ensino como finalidade educacional e social (HAYDT, 2011). Borges e Alencar (2014), define a didática como o procedimento em que um conjunto de teorias e técnicas relacionadas à transmissão do conhecimento, auxiliam o educador a transmitir o mundo da experiência e da cultura ao educando, já Piletti (2004), trata a didática como uma disciplina técnica, que tem como objeto de estudo os aspectos práticos e operacionais das técnicas de ensino.

É por meio da troca de experiências e cultura que o educador juntamente dos seus recursos didáticos, têm importante papel no aprendizado crítico-reflexivo do estudante. O educador além de transmissor de conhecimento, também deve atuar como mediador, fazendo uso destes recursos para facilitar o aprendizado do estudante de forma ativa e motivadora (BORGES; ALENCAR, 2014). O professor tem o dever de garantir o processo de construção de conhecimento ao educando, sempre tendo em vista o objetivo de estimular o desejo dos estudantes para a aprendizagem. Para Freire (1996), o educador, na sua prática docente, tem o dever de reforçar a capacidade crítica do educando, sua curiosidade e sua insubmissão. Borges e Alencar (2014), argumenta que, muitas vezes o professor tem o domínio do conteúdo a ser apresentado, porém, encontra dificuldades no aspecto metodológico, de forma que o mesmo não consegue encontrar a forma adequada de abordá-lo, podendo assim prejudicar o processo de aprendizagem.

A utilização de métodos de ensino tradicionais tem como característica a resistência a inovações, onde o professor posiciona-se em sala de aula como o único detentor e transmissor do conhecimento, mantendo assim certa distância dos estudantes que são tratados como seres passivos. As avaliações são periódicas, geralmente envolvendo provas com o objetivo de avaliar o conhecimento dos estudantes de forma mais quantitativa do que qualitativa (ALMEIDA, 2015).

Ainda segundo Almeida (2015), o objetivo desses métodos é a transmissão de conteúdos bem definidos com farta variedade de noções, conceitos e informações, mas deixando de fora deste escopo o pensamento reflexivo.

No ensino de estrutura de dados este modelo geralmente segue uma típica sequência de passos dos quais se apresentam respectivamente: a apresentação da teoria, apresentação de exemplos práticos, aplicação de exercícios e desenvolvimento de projetos. Segundo Voss et al. (2015), este tipo de metodologia direcionada ao professor, faz com que haja uma grande diferença entre os conceitos abstraídos pelos estudantes e a prática adotada pelos mesmos. O mesmo ainda afirma que somente o uso de aulas expositivas pode se tornar desvantajoso para os estudantes, uma vez que os mesmos perdem a concentração muito rapidamente, prejudicando assim o processo de abstração necessário para a construção do conhecimento.

Segundo Libânio (2006), a função da didática, ao estudar o processo de ensino, não é somente dizer que os estudantes precisam dominar os conhecimentos, é necessário mostrar como fazê-lo, ou seja, o professor deve buscar objetivos e métodos seguros e eficazes para a assimilação dos conhecimentos.

Para Gemignani (2013), a complexidade dos problemas atuais enfrentados na vida profissional exige novas competências como colaboração, conhecimento interdisciplinar, habilidade para inovação e trabalho em equipe.

> Estas novas competências demonstram a necessidade de alterar a nossa percepção e a forma de nos relacionar com o mundo circundante, modificando a abordagem mecanicista, fragmentada, competitiva e hegemônica para uma abordagem sistêmica, holística, cooperadora e integradora (GEMIGNANI, 2013, p. 3).

Diante disso Borges e Alencar (2014) defende que para preparar os estudantes numa formação crítico social é preciso que os professores passem a utilizar metodologias ativas de aprendizagem como um recurso didático na prática docente.

As metodologias ativas buscam desenvolver processos de aprendizagem utilizando experiências reais ou simuladas, como forma de criar as condições de solucionar problemas e desafios de diferentes contextos sociais. Elas fazem da problematização um recurso didático de ensino e aprendizagem, despertando a curiosidade dos estudantes ao passo que os insere de forma ativa na teorização dos conteúdos, dando a liberdade para eles poderem inserir novos elementos ou novos pontos de vista ainda não considerados pelo professor, incentivando assim a criação de um senso de autonomia nos estudantes (BERBEL, 2011; BORGES; ALENCAR, 2014).

2.3 Computação Desplugada

A computação desplugada (*Computer Science Unplugged*) é um método de ensino que fornece maneiras de expor aos estudantes, de forma lúdica, os fundamentos da Ciência da Computação sem a utilização do computador, e sem a necessidade de aprofundar em detalhes técnicos. Pode ser aplicada a todas as idades, desde o ensino fundamental até o ensino superior, podendo ser usada para o ensino de computação em lugares remotos ou que possuam pouca infraestrutura em equipamentos de informática (BELL et al, 2009; BORDINI et al., 2016).

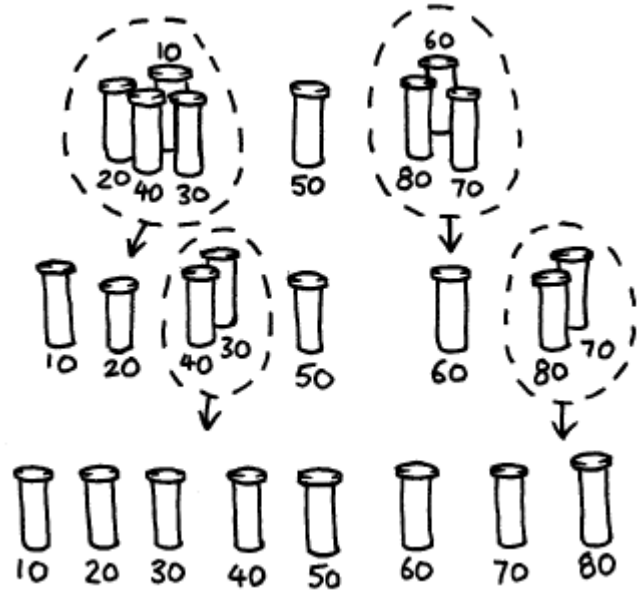
Entre outros objetivos, a metodologia busca tornar concreto vários conceitos abstratos da Ciência da Computação, tais como números binários, algoritmos, programação de computadores, compressão de dados, redes de computadores, bancos de dados, entre outros. Com isso, a metodologia possui como um de seus objetivos principais, desmistificar a Ciência da Computação, de forma que possa aumentar o interesse dos estudantes pela área, mostrando

como ela pode ser aplicada na resolução de diferentes tipos de problemas (MARTINHAGO et al, 2017).

O projeto conhecido como *CS Unplugged* teve início na Universidade de Canterbury em 1998, idealizado pelos professores Tim Bell, Jason Alexander, Isaac Freeman e Mick Grimley, que elaboraram o livro *"Computer Science Unplugged-off-line activities and games for all ages"*. A ideia foi bem aceita na comunidade acadêmica e passou a ser divulgado no site csunplugged.org, que disponibiliza gratuitamente o material do projeto em diversas línguas. O projeto ganhou boa visibilidade internacional com o apoio de grandes empresas como a Google e o Facebook.

A Figura 1 mostra uma ilustração de uma das atividades presentes no livro "Ensinando Ciência da Computação sem o uso do computador", disponibilizado no site do projeto. A atividade trata do tema algoritmos de ordenação e busca, utilizando recipientes com pesos diferentes para serem ordenados utilizando o algoritmo *quicksort*.

Figura 1 – Atividade dividir e conquistar



Fonte: Bell et al. (2011, p. 67).

Segundo o Bell et al. (2009), a iniciativa desplugada baseia-se em 10 princípios que fundamentam suas atividades, são esses:

1 O desenvolvimento das atividades não deve depender do uso de computadores;

2 Foco na demonstração dos conceitos;

3 Tornar as atividades sinestésicas, geralmente em larga escala e envolvendo trabalho em equipe, permitindo assim que os participantes possam descobrir as respostas sozinhos, fazendo com que se tornem mais autônomas no aprendizado;
4 As atividades devem ser divertidas e cativantes, com explicações breves sobre suas regras e materiais utilizados nos desafios. Geralmente as atividades envolvem algum tipo história, e os problemas são apresentados como parte dela;
5 Os materiais devem ser acessíveis e de baixo custo;
6 Os materiais são disponibilizados com uma licença que permite o compartilhamento e adaptação dos materiais;
7 As atividades possuem neutralidade de gênero;
8 As atividades geralmente têm um senso de história para manter a atenção dos estudantes;
9 As atividades são dispostas em módulos não sequenciais que podem ser utilizadas de forma independente das outras;
10 As atividades possuem caráter flexivo de forma que erros cometidos pelos participantes não devem impedir que os mesmos entendam os princípios envolvidos nas atividades.

Com base nestes princípios pode-se afirmar que a iniciativa desplugada tem como seus pilares, o ensino da Ciência Computação de forma simples e acessível para diversas faixas etárias do ensino. Ela incentiva a capacidade do estudante de resolver problemas individualmente ou em grupo, propondo atividades que sejam mais divertidas e lúdicas do que só algo para manter os participantes ocupados (BELL et al, 2009).

2.4 Estrutura de dados

Uma estrutura de dados nada mais é do que uma forma de armazenar e organizar informações na memória de um computador utilizando uma lógica específica. Segundo Cormen (2002), trata-se de uma maneira de armazenar e organizar os dados de forma que possa facilitar o acesso e as modificações. Backes (2017, p. 4) classifica as estruturas de dados como “um relacionamento lógico entre diferentes tipos de dados visando a resolução de determinado problema de forma lógica”.

As EDs são utilizadas nas mais diversas áreas de conhecimento e podem possuir diversos propósitos, assim sua utilização está sempre atrelada ao uso de algoritmos para manipulação destas estruturas. Segundo Ascêncio e Campos (2007), as EDs se diferenciam uma

das outras pela forma em que organiza e manipula seus dados, de forma que, não se pode separar as estruturas de dados dos algoritmos associados a ela.

As etapas para o desenvolvimento de um programa podem ser definidas como análise, algoritmo e codificação. Define-se um algoritmo como uma descrição de uma sequência finita de passos que devem ser seguidos para a realização de uma tarefa. Também podemos determina-lo como um procedimento computacional bem definido que possui como entrada algum valor ou conjunto de valores, para posteriormente produzir algum valor ou conjunto de valores como saída. Desta forma os processos que compõem o funcionamento de um algoritmo são: entrada de dados, processamento e saída dos dados (ASCENIO; CAMPOS, 2007; CORMEN, 2002).

Existem diferentes tipos de EDs, e sua eficiência pode variar quanto ao custo de processamento e o tempo de execução, de forma que, a escolha de uma deve depender do tipo de solução que se deseja implementar com ela. Entre as principais podemos destacar estruturas lineares como arranjos, listas, pilhas, filas e deques; e não lineares como arvores, grafos e tabelas *hash*. Algumas dessas estruturas possuem suas próprias variações como as listas que podem ser: circulares, simplesmente encadeadas, duplamente encadeadas; arvores: binárias, AVL, rubro-negra; entre outros.

2.4.1 O ensino de estrutura de dados

O ensino de estrutura de dados acontece em sua maioria no contexto das ciências exatas, pois nessa área encontra-se mais presente o uso do raciocínio lógico e matemático para a resolução de problemas. Esta disciplina faz parte da matriz curricular dos cursos da área computação, e possui como pré-requisito conhecimentos sobre algoritmos e programação. No projeto pedagógico do curso de Bacharelado em Ciência da Computação do Campus VII da Universidade Estadual da Paraíba (UEPB), a disciplina é dividida em Estrutura de Dados e Laboratório de Estrutura de Dados, sendo uma com enfoque teórico e a outra prático. Os assuntos abordados nas duas disciplinas tratam de abstração de dados, estruturas de dados lineares, árvores e suas generalizações, grafos, tabelas *hash*, ordenação e busca e introdução à complexidade de algoritmos (UNIVERSIDADE ESTADUAL DA PARAÍBA, 2016).

Segundo Santana et al. (2009), o processo de aprendizagem das EDs envolve um estudo teórico e prático dos conceitos abordados, no qual deve possibilitar ao estudante criar representações dos objetos e implementar rotinas capazes de atuar sobre essas representações.

Santos e Costa (2005) afirma que é necessário dar mais destaque a parte conceitual e comportamental das estruturas, para só depois partir para a parte prática.

A utilização de uma linguagem de programação para desenvolver EDs permite identificar diversas ações que segundo Valente et al. (1999), se mostram de grande importância na aquisição de novos conhecimentos para o estudante. Essas ações envolvem os processos de descrição da solução de um problema usando uma linguagem de programação, execução dessa descrição pelo computador, reflexão sobre o que foi produzido e depuração do resultado.

Essas atividades acontecem em um ciclo no qual exige que o estudante estruture seus conhecimentos para entender os conceitos envolvidos na solução do problema, defina estratégias para a aplicação desse conhecimento, e após aplicar a solução, reflita sobre seu resultado, que caso não saia como esperado, o leva a um processo de depuração, que é a busca de novas informações e estratégias para atingir a solução do problema, fazendo com que o ciclo se repita (VALENTE et al., 1999).

O ensino de ED e programação não se limita ao estudo das linguagens de programação, mas sim a aplicação de diversos tipos de conhecimentos que transformam um modelo de solução de um problema em um produto de software. Desta forma a natureza específica destas práticas se mostram diferentes de outras disciplinas, em que os métodos de estudos utilizados pelos estudantes, se baseiam em leitura e memorização de conceitos e formulas, de forma mecânica e pouco reflexiva, sendo que, o estudo de ED envolve além do domínio da programação, uma boa capacidade de abstração, resolução de problemas e raciocínio logico e matemático, exigindo dos estudantes uma compreensão mais aprofundada e reflexiva dos assuntos, e uma prática mais intensiva (GOMES et al, 2008; SANTOS; COSTA, 2006).

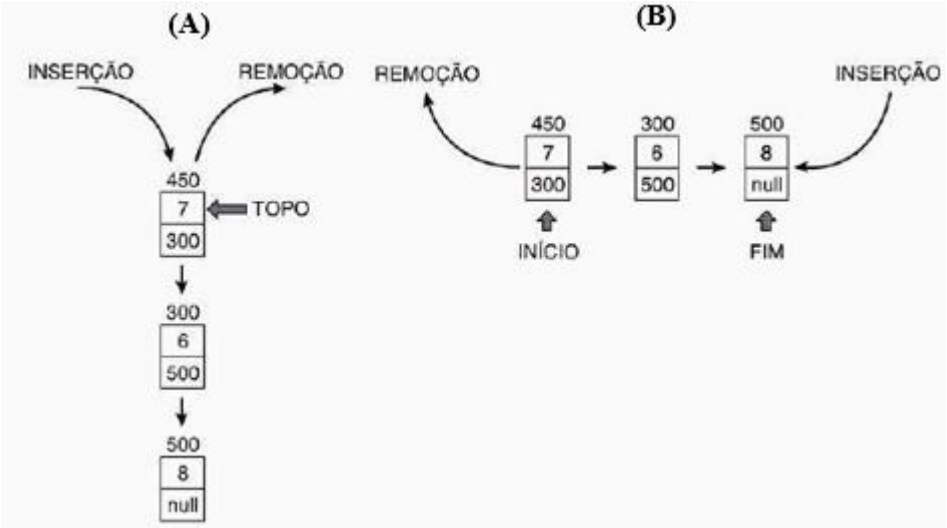
2.4.2 Listas encadeadas

A lista encadeada, de uma maneira simples, trata-se de um conjunto finito de itens do mesmo tipo organizado de uma forma linear (ASCENIO; CAMPOS, 2007). Este tipo de estrutura pode ser utilizado em diferentes aplicações, e se encaixar em diversos cenários de uso, como por exemplo, uma lista de convidados, de itens de estoque de uma empresa, de processos a serem executados por um processador, entre tantos outros.

Segundo Backes (2017), existem diferentes representações de uma lista, que podem variar de acordo com a forma que os elementos são inseridos ou removidos, o tipo de alocação de memória utilizada, e o tipo de acesso aos elementos.

Quanto a inserção e remoção, podemos classificar as listas como fila: os elementos são inseridos somente no final e acessados ou removidos no início da lista (Figura 2, B), seguindo o padrão FIFO (*First In First Out*); pilha: os elementos só podem ser inseridos, removidos ou acessados no final da lista (Figura 2, A), seguindo o padrão LIFO (*Last In First Out*); e listas convencionais: os elementos podem ser acessados, inseridos e removidos em qualquer lugar da lista (BACKES, 2017).

Figura 2 – Estrutura do tipo Pilha (A) e Fila (B)

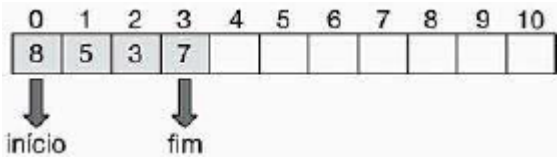


Fonte: Ascêncio e Campos (2007, p. 184, com adaptações).

Quanto a alocação de memória podemos utilizar de forma estática ou dinâmica, na alocação estática é necessário que no momento da implementação, seja definido o número máximo de elementos que a lista deve possuir, assim, os espaços de memória são alocados durante a compilação do programa; já na alocação dinâmica, não se faz necessária uma prévia definição do tamanho da lista, pois os espaços de memória são alocados a medida que novos elementos são armazenados na mesma (BACKES, 2017).

Independentemente do tipo de alocação, o acesso aos elementos da lista pode acontecer de forma sequencial ou encadeada. No acesso sequencial os elementos são armazenados de forma consecutiva na memória, a exemplo dos vetores, que podem fazer este acesso diretamente através do seu índice, como mostra a Figura 3. Já no acesso encadeado, os elementos são armazenados em áreas distintas da memória, isso faz com que seja necessário que eles armazenem, além de seu dado, o endereço da posição de memória que se encontra o próximo elemento, diferente de um vetor, para acessar um dado elemento na lista, é preciso percorrer todos os seus antecessores (BACKES, 2017).

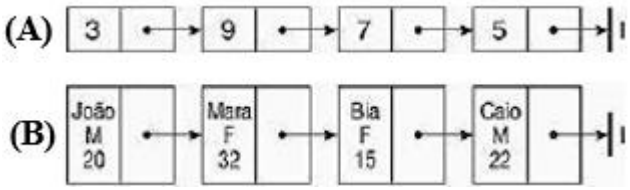
Figura 3 – Lista estática



Fonte: Ascêncio e Campos (2007, p. 106).

Com relação aos tipos de dados que uma lista pode armazenar, podemos classifica-los como homogêneos os dados primitivos como um número ou uma cadeia de caracteres, como mostra a Figura 4 (A), e heterogêneos os dados compostos, como um cadastro de funcionário ou uma lista telefônica, como mostra a Figura 4 (B) (ASCENIO; CAMPOS, 2007).

Figura 4 – Lista dinâmica homogênea (A) e heterogênea (B)



Fonte: Ascêncio e Campos (2007, p. 106, com adaptações).

É possível implementar tanto de forma dinâmica, quanto estática, diferentes tipos de listas, podendo estas serem: simplesmente encadeada, ordenada e não ordenada; duplamente encadeada, ordenada e não ordenada; e circular. Dentre os tipos de lista apresentados, destaca-se as seguintes operações básicas das quais são possíveis de serem executadas numa lista: criação da lista; inserir, remover e buscar elemento da lista; destruir a lista e obter informações sobre o tamanho da lista (ASCENIO; CAMPOS, 2007, BACKES, 2017).

Uma lista dinâmica e encadeada é caracterizada pela alocação dinâmica, e acesso encadeado aos elementos, também chamados de “nós”, assim uma porção de memória é alocado dinamicamente, ou seja, em tempo de execução, para cada elemento adicionado na lista, e posteriormente essa memória é liberada quando um elemento é excluído. Os nós contidos numa lista são na verdade ponteiros, ou seja, um elemento que armazena um endereço de memória para uma estrutura, que na lista simples, é formado por dois campos, um campo tem por finalidade armazenar a informação inserida na lista, e o outro nada mais é do que um ponteiro que armazena o endereço de um nó posterior, sendo que, nas listas duplas um ponteiro adicional é inserido a esta estrutura para armazenar o endereço de um elemento anterior. Além disso ainda

se faz necessário a utilização de um ponteiro para ponteiro, que tem apenas a função de armazenar o endereço do primeiro elemento da lista (BACKES, 2017).

2.5 Trabalhos Relacionados

Nesta seção são apresentados alguns estudos que possuem características relacionadas ao tema desta pesquisa. Dentre os trabalhos descritos, voltados para o ensino de estrutura de dados, podemos destacar os de Souza et al. (2011), Voss et al. (2015), Moreira e Monteiro (2018), Costa et al. (2017), Santiago e Kronbauer (2017); pela utilização de recursos como ferramentas computacionais destaca-se Souza et al. (2011) e Voss et al. (2015); e pela utilização de métodos desplugados são apresentados os estudos de Moreira e Monteiro (2018), Costa et al. (2017) e Martinhago et al. (2017).

A pesquisa de Santiago e Kronbauer (2017), publicada na Revista Brasileira de Informática na Educação, descreve a criação de um modelo lúdico para o ensino de algoritmos recursivos. Segundo o autor, seu principal objetivo é propor a criação e validação de uma ferramenta visual e ilustrativa para o ensino de conceitos relacionados a programação de computadores, por meio do uso de metáforas lúdicas. Ele utilizou de avaliações quantitativas e qualitativas do modelo proposto para concluir que houve um nível satisfatório de aceitação da proposta por parte dos estudantes, que as metáforas lúdicas possuem potencial metodológico para o ensino de conceitos de programação de computadores nas universidades, mas também revela que a visualização simultânea do código fonte é importante para uma melhor compreensão dos algoritmos, pois permite que os alunos relacionem as metáforas com os passos descritos em código textual.

Souza et al. (2011), apresenta uma ferramenta computacional para auxiliar o ensino de estrutura de dados. O Simulador Integrado para Estrutura de Dados (SIED), trata-se de uma plataforma *web* composta por 12 módulos e 49 submódulos, onde cada submódulo representa um assunto especifico de ED. Seu objetivo é fornecer um ambiente completo e compatível com o conteúdo visto na disciplina. Segundo o autor, a ferramenta foi avaliada de forma positiva por meio de um questionário presente na própria ferramenta. O trabalho foi publicado em novembro de 2011 nos Anais do SBIE em Aracajú.

Publicado nos Anais dos Workshops do Congresso Brasileiro de Informática na Educação (CBIE), no ano de 2015, a pesquisa de Voss et al. (2015) apresenta a proposta de um jogo educacional para o ensino de ED do tipo arvore. O autor aponta algumas dificuldades enfrentadas pelos estudantes em contato com estes conteúdos, e defende a utilização mais

intensiva de aulas práticas neste tipo de disciplina. O jogo educacional tem como objetivo fornecer um ambiente para que os estudantes possam praticar os conceitos adquiridos em sala de aula, visando solidificar a construção do conhecimento e possibilitar uma maior interação entre os estudantes. O jogo está em fase de desenvolvimento e tem como objetivos futuros expandir a plataforma para outras áreas da computação.

A pesquisa de Moreira e Monteiro (2018), realizada no Instituto Federal de Ciência e Tecnologia do Sertão Pernambucano, Campus Petrolina, em 2018, apresenta um relato de experiência de reformulação da disciplina de ED em um curso de computação, utilizando gamificação e computação desplugada, com objetivo de simplificar conceitos e motivar os estudantes, a fim de obter uma aprendizagem significativa. Segundo o autor, o uso de metodologias como está incentiva a criação de estratégias para o ensino de conteúdos considerados complexos e traz consigo melhorias no aprendizado dos estudantes.

O trabalho de conclusão de curso de Costa et al. (2017), feito na Universidade Federal da Paraíba (UFPB), na cidade de Rio Tinto em 2017, expõe um relato de experiência sobre aplicação de atividades desplugadas para estudantes da disciplina de ED, nos cursos de Licenciatura em Ciência da Computação e Bacharelado em Sistemas de Informação. O objetivo da pesquisa é utilizar a computação desplugada para apoiar a aprendizagem de algoritmos por comparação e de tabela *hash*. O autor descreve o processo de criação e aplicação das atividades desplugadas em sala de aula, que tratam sobre algoritmos de ordenação (inserção, seleção, mergesort e quicksort), busca binária e tabela *hash* com encadeamento. Através de análises quantitativas e qualitativas da proposta, feitas com base na aplicação de testes estatísticos, o autor concluiu que a computação desplugadaauxilia na compreensão da lógica de funcionamento dos algoritmos de ordenação por comparação e de tabela *hash*.

A pesquisa de Martinhago et al. (2017), realizada na Universidade Federal de Viçosa, Campus de Rio Paranaíba-MG, foi publicada no WEI-XXII Workshop sobre Educação em Computação no ano de 2017, descreve um relato de experiência, de desenvolvimento e aplicação de atividades desplugadas para a disciplina de Banco de Dados, do curso de Sistemas de Informação. O autor descreve o processo de criação e execução de atividades desplugadas baseadas nos conceitos de bancos de dados, abstração de dados, modelo relacional e consultas SQL, produzindo um material didático para cada uma destas atividades. Segundo o autor, os resultados apontam que o método não teve impacto significativo na aprendizagem dos conceitos, mas obteve resultados positivos e significativos com relação à satisfação dos estudantes.

Tabela 1 – Relação dos trabalhos relacionados

Trabalho	Contribuição
Santiago e Kronbauer (2017)	Ferramenta visual e ilustrativa para o ensino de conceitos relacionados a programação de computadores, por meio do uso de metáforas lúdicas.
Souza et al. (2011)	Ferramenta computacional para auxiliar o ensino de estrutura de dados.
Voss et al. (2015)	Proposta de um jogo educacional para o ensino de ED do tipo arvore.
Moreira e Monteiro (2018)	Reformulação da disciplina de ED em um curso de computação, utilizando gamificação e computação desplugada.
Costa et al. (2017)	Aplicação de atividades desplugadas para apoiar a aprendizagem de algoritmos por comparação e de tabela *hash* para estudantes da disciplina de ED.
Martinhago et al. (2017)	Desenvolvimento e aplicação de atividades desplugadas baseadas nos conceitos de bancos de dados, abstração de dados, modelo relacional e consultas SQL, para a disciplina de Banco de Dados.

Fonte: Elaborada pelo autor, 2019.

3 RESULTADOS E DISCUSSÕES

Esta seção tem por finalidade apresentar a proposta de abordagem didática, que através do uso de conceitos desplugados, busca apoiar o ensino de estrutura de dados do tipo lista. Aqui são descritos os recursos necessários para a aplicação da proposta em sala de aula, como também os materiais para a sua execução. As operações que podem ser realizadas sobre a estrutura de dados lista dinâmica são descritas através de código na linguagem de programação C e adaptadas para abordagem metodológica desenvolvida nesta pesquisa.

3.1 Descrição da proposta

A proposta descrita neste trabalho tem como base os conceitos sobre as EDs do tipo lista, sendo que, entre os tipos de listas apresentados, optou-se por trabalhar com a lista dinâmica e simplesmente encadeada, pelo fato deste tema ainda ser pouco explorado no âmbito da computação desplugada como abordagem didática, no contexto do ensino em cursos superiores.

Esta estrutura é caracterizada pela alocação dinâmica de memória durante as operações, e o acesso encadeado aos elementos, de modo que, cada um deles armazena o endereço do elemento subsequente. Dentre as operações definidas em Backes (2017), sobre este tipo de estrutura, foram escolhidas, para serem representadas na forma de dinâmica, operações básicas que se mostraram mais acessíveis para este formato. São listadas as seguintes operações: criar a lista; inserir elemento no início e no final da lista; remover elemento no início, no final e em uma posição qualquer da lista; e consultar elemento na lista pela posição.

Apesar desta proposta dar mais ênfase a aplicação das atividades utilizando o conceito de listas encadeadas convencionais, a partir da definição destas operações é possível trabalhar também com as estruturas do tipo fila, utilizando as operações de inserir no final e remover no início; e do tipo pilha com as operações de inserir e remover no final; além das operações de consultar elemento e criar a lista, que se aplicam as duas estruturas.

Buscou-se então uma maneira de representar essas operações no espaço de uma sala de aula, utilizando conceitos desplugados para criar uma dinâmica de grupo em que os estudantes pudessem participar de forma mais ativa no processo de aquisição destes conhecimentos.

A construção da dinâmica seguiu-se da seguinte forma: primeiro criou-se uma representação física para todos os elementos que compõem este tipo de estrutura, fazendo com

que a própria sala de aula se torne uma estrutura de dados, e os estudantes, os elementos que a compõem. O segundo passo foi criar, para cada uma das operações escolhidas, uma maneira de serem representadas pelos participantes da dinâmica, explicando passo a passo a forma como cada operação deve ser representada pelos mesmos. Para a aplicação da presente abordagem didática foram definidos os seguintes recursos:

- A sala de aula com as cadeiras organizadas em filas, alinhadas e identificadas por uma letra e um número. Como forma de identificar visualmente as cadeiras com seu respectivo endereço, pode ser utilizado notas adesivas, escrevendo nelas os endereços e colando em cada uma das cadeiras (Apêndice A). As cadeiras dispostas na sala de aula representam a memória do computador;
- O mediador (professor ou monitor da disciplina de Estrutura de Dados) que representa o Sistema Operacional e está rotulado como SO. O SO tem o papel de gerenciar as operações envolvidas na abordagem didática e indicar em que posição de memória (cadeira na sala) os elementos (estudantes escolhidos) serão armazenados;
- O dado é o conteúdo que cada elemento carrega, na dinâmica é representado por um papel, que pode ser uma nota adesiva, onde a informação que será armazenada é escrita pelo SO;
- O Cabeça de Lista (CL) é o elemento que indica o início da lista, ele armazena o endereço de memória do primeiro elemento da lista. O SO escolhe um estudante para ser o CL, que deverá anotar, utilizando lápis grafite e um cartão (Apêndice A), o endereço do primeiro elemento a ser armazenado;
- O "nó" é o elemento armazenado na lista, ele contém o dado e o endereço do nó adjacente a ele. Cada nó da lista é representado por um estudante;
- O parâmetro que indica o próximo elemento da lista, daqui por diante intitulado como *PROX*, faz parte da estrutura de cada nó da lista. Ele tem a função de armazenar o endereço do nó posterior de cada elemento. Os estudantes que representarem os nós armazenados utilizam lápis grafite e um cartão (Apêndice A), para guardar essa informação;
- Os elementos auxiliares (AUX) são utilizados para executar as operações que percorrem e acessam os nós da lista. Dependendo da operação, pode ser utilizado um ou dois auxiliares (AUX1 e AUX2). Como na maioria das operações, os elementos auxiliares estão constantemente mudando de posição, então não possuem local definido nas cadeiras da sala, já que eles são utilizados apenas durante as operações na

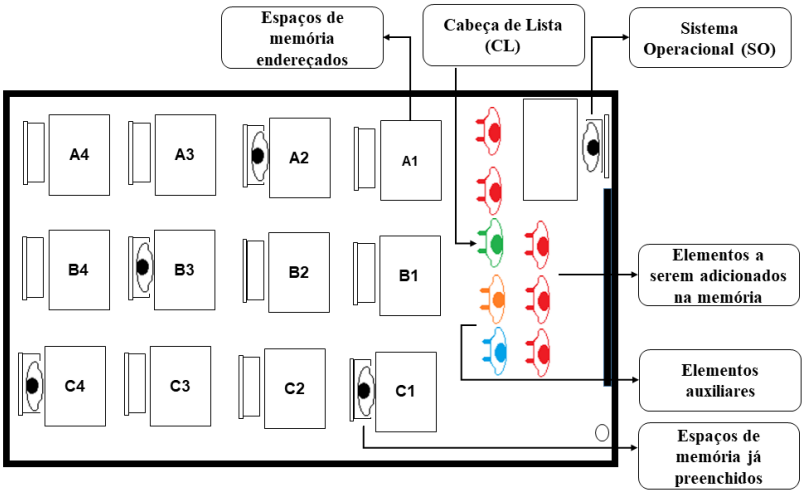
lista. Os mesmos devem fazer uso de um lápis grafite e um cartão (Apêndice A) para registrar as informações necessárias. Para cada auxiliar é escolhido um estudante para representa-lo.

Esta dinâmica deve ser praticada mediante a discussão teórica do assunto (estruturas de dados do tipo lista, filas, pilhas e suas operações). São necessários 8 estudantes para a realização das operações da lista, um número maior de participantes pode ser utilizado de acordo com o tamanho da turma, e o tempo disponível para realização da dinâmica, mas para uma boa execução recomenda-se participação de pelo menos 8 participantes.

O mediador (SO) da dinâmica deve iniciar com uma breve explicação dos conceitos e as operações requeridas para o entendimento e implementação de cada estrutura. Para representar os elementos da lista, são escolhidos pelo SO, estudantes da sala para se posicionar à frente da turma para dar início a dinâmica. O mediador deve explicar de forma clara aos estudantes a relação entre a sala de aula e a memória de um computador, onde cada cadeira é um endereço único de memória. Os espaços alocados na memória do computador para armazenar os nós da lista não, necessariamente, estão próximos uns aos outros. Dessa forma, a cadeira que o estudante (novo elemento da lista) vai ocupar pode ser qualquer uma, desde que não esteja ocupada por outro estudante (elemento). Para manter a lógica da ED, cada estudante (elemento da lista) terá consigo a referência do nó posterior a ele, sempre utilizando os cartões para registrar as informações necessárias.

Antes de iniciar a dinâmica é necessário preparar o cenário. A Figura 5 mostra como a sala deve ser organizada: as cadeiras alinhadas e identificadas; os estudantes que irão representar os elementos da lista a serem adicionados na memória, em pé na frente dos demais, que continuam sentados (representando posições de memória ocupadas no computador); um estudante, dos que estão na frente, representa a cabeça da lista (em verde); e dois estudantes representam os elementos auxiliares (azul e laranja), que são necessários para percorrer a lista, os demais estudantes (em vermelho) serão os elementos a serem inseridos na lista.

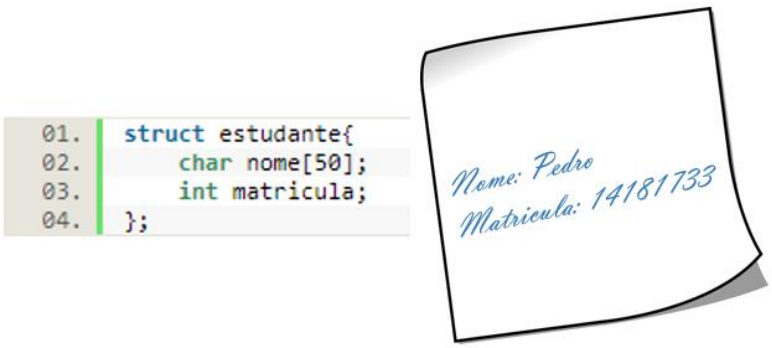
Figura 5 – Descrição do cenário



Fonte: Autoria própria, 2019

Antes de criar a lista, é necessário definir o tipo de dado que será armazenado, qualquer tipo de dado pode ser guardado numa lista, porém é necessário antes especificá-lo, então o mediador deve preparar os papéis com os dados a serem armazenados no momento da inserção. A Figura 6 mostra a criação de um tipo de dado heterogêneo feito na linguagem de programação C (a esquerda), e a representação de um dado criada para a dinâmica (a direita).

Figura 6 – Tipos de dados da lista



Fonte: Autoria própria, 2019

3.2 Material necessário

A quantidade de material necessário para a execução da abordagem pode variar de acordo com o número de participantes e a quantidade de cadeiras utilizadas. Os recursos utilizados nesta abordagem foram pensados de maneira que possa simplificar a execução e o

entendimento da mesma. Os materiais necessários para uma execução com 8 estudantes têm um custo estimado de R$ 25,00, e estão listados da seguinte forma:

- 8 lápis e borracha, para os participantes;
- 1 pincel para quadro branco, para o mediador;
- 1 folha de papel de tamanho A4, para confecção dos cartões;
- 1 bloco de notas adesivas de formato quadrado, para o endereçamento das cadeiras e a criação dos dados a serem armazenados.

3.3 Descrição das operações

Esta subseção descreve cada uma das operações escolhidas, demonstrando seu funcionamento através de código escrito na linguagem de programação C, e apresentando em seguida, uma descrição em forma de dinâmica para ser representada pelos estudantes em sala de aula.

3.3.1 Operação criar a lista

A criação de uma lista é feita por função e/ou operação em linguagem de programação em que se aloca memória para o elemento que guarda o endereço de memória do primeiro nó da lista (que de fato tem os dados). A Figura 7 mostra como uma lista é criada na linguagem de programação C. O valor NULO indica que CL (cabeça de lista *li) inicialmente não guarda o endereço de qualquer elemento (BACKES, 2017).

Figura 7 – Código da função que cria uma lista

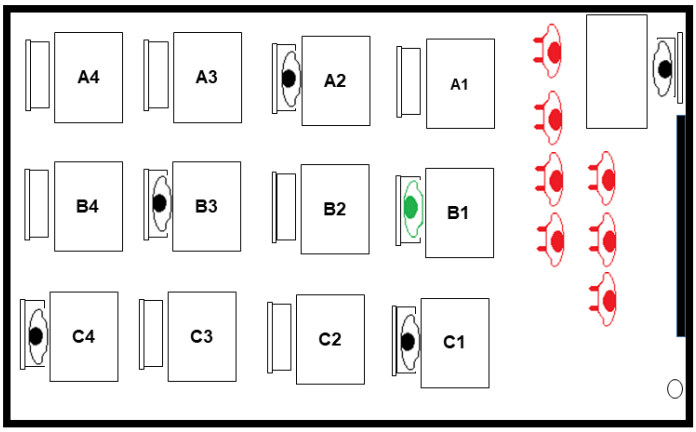
```
Lista* criar_lista(){
    Lista* li = (Lista*) malloc(sizeof(Lista));
    if(li != NULL)
        *li = NULL;
    return li;
}
```

Fonte: Autoria própria, 2019

A operação de criação da lista envolve os com materiais manipuláveis está ilustrada na Figura 8 e pode ser descrita da seguinte forma:

1. O SO escolhe um dos participantes para ser o CL e fala para ele o endereço da cadeira onde deve sentar-se (um espaço de memória é alocado pelo SO e o elemento é armazenado na memória).
2. O estudante CL senta-se na cadeira indicada (Figura 8, B1) e anota em seu cartão a referência (endereço) do próximo elemento da lista, que neste caso é NULO.

Figura 8 - Criação da lista



Fonte: Autoria própria, 2019

Esta é a operação que dá início a dinâmica, como mostra a imagem, podem existir posições de memória que já estejam ocupadas por outros estudantes, isto quer dizer que todos na sala, mesmo que não estejam fazendo um papel especifico, também estão participando e ajudando a dar mais sentido a sua execução, isto faz com que todos na sala se mantenham envolvidos na atividade. Após essa operação deve-se seguir com uma operação de inserção na lista.

3.3.2 Operação inserir elemento no início da lista

Para inserir um elemento no início da lista inicialmente aloca-se memória para o novo elemento, este passa a armazenar o mesmo endereço que o cabeça da lista esteja guardando, o próximo passo é fazer com que o elemento CL armazene o endereço do novo nó. Caso a lista esteja vazia, a referência que o novo elemento armazenou será igual a NULO (BACKES, 2017). A Figura 9 exibe como ocorre a inserção de um elemento no início da lista utilizando a linguagem de programação C.

Figura 9 – Código da função que insere no início da lista

```
int inserir_inicio(Lista* li, struct aluno al){
    if(li == NULL)
        return 0;
    Elemento* no;
    no = (Elemento*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = (*li);
    *li = no;
    return 1;
}
```

Fonte: Autoria própria, 2019

A operação inserir elemento no início da lista está ilustrada na Figura 10 com materiais manipuláveis e pode ser descrita da seguinte forma:

1. O SO escolhe um dos estudantes para ser inserido na lista, entrega a ele o dado que deve armazenar, somente o participante e o SO devem saber o conteúdo deste dado, e indica em que posição da memória este deve sentar-se (o SO aloca memória para um novo nó).
2. O estudante vai até o CL, fala que será o novo primeiro nó da lista, pega com ele o endereço do primeiro elemento e armazena essa informação em seu cartão no campo *PROX*, definindo então a ligação com o próximo elemento (Figura 10, B), caso a lista esteja vazia este endereço será NULO (Figura 10, A).
3. O novo nó informa ao CL o endereço em que ele será armazenado e o CL passa a guardar o endereço informado.
4. O estudante vai até o endereço que lhe foi passado pelo SO e senta-se (o novo nó foi armazenado na memória).

A Figura 10, em (A), mostra a inserção de um novo elemento, no caso em que a lista se encontra vazia, representado por um estudante sentado na cadeira na posição B2. Em (B), a lista contém 3 nós armazenados e o estudante que representa o novo elemento agora armazena a posição A1, onde encontra-se o seu próximo.

Figura 10 – Inserindo elemento numa lista vazia (A) e com elementos (B)

Fonte: Autoria própria, 2019

Uma preocupação no momento de descrever as operações em forma de dinâmica, foi conseguir manter a lógica e o máximo possível de semelhança com os códigos das funções apresentadas, mas devido à complexidade de algumas operações, e a existência de passos repetitivos ou semelhantes entre elas, tomamos a liberdade de omitir ou adaptar alguns passos do algoritmo que as compõe, como forma de diminuir a complexidade, tornando as atividades mais acessíveis e simples de serem executadas pelos estudantes.

Diferente da operação de criar a lista, onde os passos descritos representam bem todas as linhas de código ilustradas na Figura 7; nesta operação, alguns trechos de código vistos na Figura 9, como os condicionais das linhas 2 e 6, que tratam se a lista ou o elemento a ser adicionado são válidos, foram omitidos, com base nos critérios supracitados.

3.3.3 Operação inserir elemento no final da lista

A operação inserir no final da lista possui custo operacional maior do que inserção no início, já que para inserir no final é necessário percorrer todos os elementos da lista. Inicialmente é necessário alocar memória para o novo elemento e criar um elemento auxiliar para percorrer a lista até encontrar seu último nó, feito isso, o último nó passa a guardar o endereço do novo nó que passa ser o último da lista, e tem sua referência de próximo, o valor NULO. Para o caso em que a lista esteja vazia, muda-se somente a referência do CL, que passa a guardar o endereço do novo nó (BACKES, 2017). A Figura 11 exibe como ocorre a inserção de um elemento no final da lista utilizando a linguagem de programação C.

Figura 11 – Código da função que insere no final da lista

```
int inserir_final(Lista* li, struct aluno al){
    if(li == NULL)
        return 0;
    Elem *no;
    no = (Elem*) malloc(sizeof(Elem));
    if(no == NULL)
        return 0;
    no->dados = al;
    no->prox = NULL;
    if((*li) == NULL){
        *li = no;
    }else{
        Elem *aux;
        aux = *li;
        while(aux->prox != NULL){
            aux = aux->prox;
        }
        aux->prox = no;
    }
    return 1;
}
```

Fonte: Autoria própria, 2019

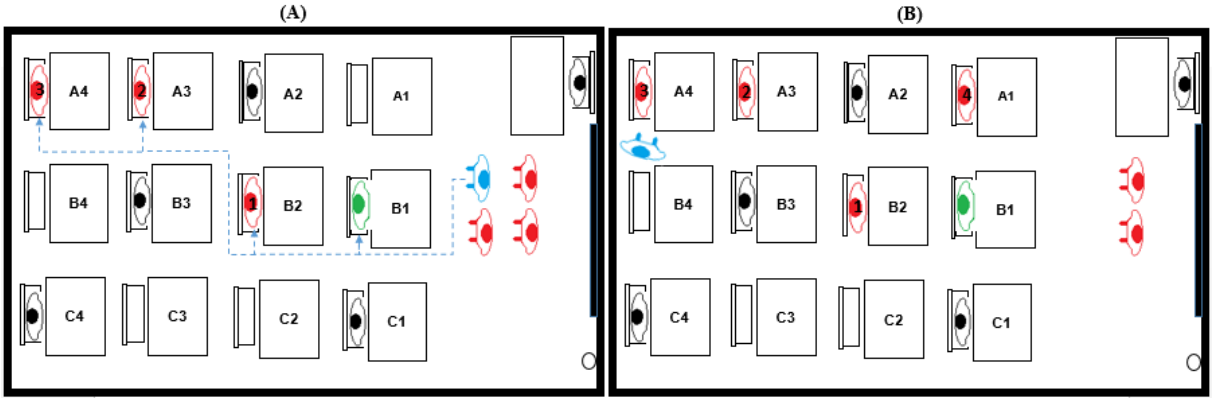
A operação inserir elemento no final da lista é ilustrada na Figura 12, com materiais manipuláveis, e pode-se descreve-la da seguinte forma:

1. O SO escolhe um dos estudantes para ser inserido no final da lista, entrega a ele o dado que deve armazenar e indica em que posição da memória este deve sentar-se, somente o SO e o participante devem saber o conteúdo do dado.
2. O participante vai até o CL e pergunta se a lista está vazia, ou seja, se o CL guarda o endereço NULO.
3. Neste momento observa-se as seguintes situações:
    a. Caso a lista esteja vazia:
        i. O estudante informa ao CL que será o novo primeiro nó da lista, pega com ele o endereço do primeiro elemento que tem o valor de NULO, e armazena essa informação em seu cartão no campo *PROX*.
        ii. O novo nó informa ao CL o endereço em que ele será armazenado e o CL passa a guardar o endereço informado.
        iii. O estudante vai até o endereço que lhe foi passado pelo SO e senta-se (o novo nó foi armazenado na memória).
    b. Caso a lista não esteja vazia, o estudante passa essa informação ao SO, que escolhe mais um participante que atuará como elemento auxiliar, AUX, e terá a função de percorrer a lista em busca do último nó.

4. O AUX vai até o CL e pega o endereço do primeiro nó da lista, ele anota em seu cartão e vai até este endereço (Figura 12, A).
5. O AUX chega para o nó que está no endereço que recebeu e pega o endereço do próximo nó, ele atualiza o endereço em seu cartão e segue para o próximo (Figura 12, A). Este processo repete-se até que o AUX encontre um nó que tenha um endereço NULO armazenado, ou seja, o último nó da lista.
6. O AUX fala para o último nó da lista guardar o endereço do novo nó (Figura 12, B).
7. O novo nó vai até o endereço que lhe foi passado pelo SO, senta-se, e registra seu *PROX* como NULO (o novo nó foi armazenado no final da lista).

A Figura 12 representa o caso de uma inserção no final, em uma lista com 3 elementos. Em (A), o estudante que representa o elemento auxiliar (em azul) percorre a lista desde o CL na posição B1, até encontrar o último elemento, em A4. Ao chegar no último nó, o auxiliar informa que um novo elemento foi adicionado na posição A1, então o elemento 3 da lista armazena a posição A4 como próximo, e o elemento 4, por ser o último deve armazenar o valor NULO.

Figura 12 – Inserindo elemento no fim em numa lista com 3 nós armazenados



Fonte: Autoria própria, 2019

A ação do auxiliar de percorrer a lista acontece na forma de um laço, pois possui passos que se repetem até satisfazer uma condição (chegar até o ultimo nó), é papel do moderador, cabe ao mediador que representa o SO, fazer com que isso seja claro para os estudantes. Também é possível notar que as posições em que os elementos são armazenados não precisão seguir uma sequência, fazendo uma analogia ao comportamento da lista na memória do computador, cabe ao SO fazer com que essa prática seja entendida pelos estudantes.

3.3.4 Operação remover elemento no início da lista

Assim como na operação de inserção no início, a remoção de elemento no início da lista é relativamente simples. Deve-se inicialmente verificar se a lista é válida, e se não está vazia. O próximo passo é fazer com que o cabeça da lista guarde o endereço do nó seguinte ao atual primeiro elemento da lista, caso só exista um elemento armazenado, então o CL armazena o valor NULO (BACKES, 2017). A Figura 13 exibe o código na linguagem de programação C, que mostra como ocorre a remoção de um elemento no início da lista.

Figura 13 – Código da função que remove no início da lista

```
int remover_inicio(Lista* li){
    if(li == NULL)
        return 0;
    if((*li) == NULL)
        return 0;

    Elem *no = *li;
    *li = no->prox;
    free(no);
    return 1;
}
```

Fonte: Autoria própria, 2019

A operação remover elemento no início da lista está ilustrada (Figura 14), com materiais manipuláveis, da seguinte forma:

1. O SO escolhe um participante para ser um elemento auxiliar, AUX.
2. O AUX vai até o CL e pega o endereço do primeiro nó da lista.
3. A partir deste ponto deve-se observar os seguintes casos:
    a. Se o valor do endereço for NULO, então a lista está vazia e a operação é encerrada.
    b. Caso o endereço seja diferente de NULO, o AUX vai até o nó indicado na referência, e anota o endereço do próximo elemento da lista para o qual este nó estaria apontando.
4. O estudante que representa o nó levanta-se liberando a posição de memória que estava ocupando, e volta para frente da turma.

5. O AUX volta para o CL e fala que ele agora deve armazenar o endereço que anotou (Figura 14, A). Se não existir outro nó além do primeiro, então o CL guardará o valor NULO (Figura 14, B).

Como pode ser observado na Figura 14, são simulados dois casos possíveis para esta operação, no caso (A), a lista possui dois elementos, o auxiliar (em azul) vai até o CL e pega o endereço do elemento que está armazenado na posição B2, vai até o local e pega o endereço do próximo elemento (B4) e informa ao estudante que ele será eliminado, em seguida volta para a posição B1 e informa que o CL agora deve armazenar o endereço B4. Em (B) o processo ocorre de maneira similar, com a diferença de que a lista agora se torna vazia, então o CL armazena o valor NULO em seu cartão.

Figura 14 – Removendo elemento no início da lista com 2 (A) e 1 (B) elemento (s)

Fonte: Autoria própria, 2019

O fato de um estudante ter sido eliminado da lista não impede que ele participe em operações posteriores. É importante observar que os estudantes que fazem o papel de elementos auxiliares têm uma participação mais ativa em boa parte das operações, sabendo disso, recomenda-se, entre uma operação e outra, fazer um revezamento dos papéis representados pelos estudantes, como forma de proporcionar uma experiência mais homogênea para todos, de forma que, por exemplo, um estudante que fez o papel de elemento auxiliar durante uma operação, pode ser o próximo elemento a ser adicionado na lista em uma operação posterior.

3.3.5 Operação remover elemento no final da lista

Assim como na operação de inserir no final da lista, para remover no final é necessário o uso de elementos auxiliares para percorrer a lista em busca do último nó, ou seja, aquele que

tem como referência de próximo o valor NULO. São utilizados dois elementos auxiliares para percorrer a lista, sendo que um deles permanece sempre um nó a frente do outro, para que ao chegar no último elemento da lista, o penúltimo elemento passe a armazenar o endereço do elemento seguinte ao último, que neste caso, será igual a NULO. Se houver somente um elemento na lista então simplesmente substitui o endereço que o CL tem armazenado, pelo do elemento da posição seguinte, o valor NULO (BACKES, 2017). A Figura 15 mostra o código, escrito na linguagem C, para remover elementos do final da lista.

Figura 15 – Código da função que remove no final da lista

```
int remover_final(Lista* li){
    if(li == NULL)
        return 0;
    if((*li) == NULL)
        return 0;

    Elem *ant, *no = *li;
    while(no->prox != NULL){
        ant = no;
        no = no->prox;
    }

    if(no == (*li))
        *li = no->prox;
    else
        ant->prox = no->prox;
    free(no);
    return 1;
}
```

Fonte: Autoria própria, 2019

A operação de remover elemento no final da lista é ilustrada (Figura 16), com materiais manipuláveis, da seguinte forma:

1. Para esta operação são necessários dois auxiliares para percorrerem a lista, AUX1 e AUX2. O SO escolhe dois participantes para ocuparem estes papéis.
2. O AUX1 vai até o CL, pega o endereço do primeiro nó da lista e armazena em seu cartão.
3. Com base nessa informação, o AUX1 deve observar as seguintes situações:
    a. Se o valor do endereço for igual a NULO, então a lista está vazia e a operação é encerrada.
    b. Caso contrário, o AUX1 vai até o endereço do primeiro nó da lista e pede o endereço armazenado em seu *PROX*, o AUX1 atualiza o endereço em seu cartão e faz a seguinte verificação:

i. Se o valor da variável *PROX* deste nó for igual a NULO, só existe um elemento na lista, então o nó levanta-se, liberando a posição de memória que estava ocupando, e volta para frente da turma, e o AUX volta para o CL e fala que ele agora deve armazenar o endereço que anotou, ou seja, o valor NULO, e a operação é encerrada.

ii. Caso o valor seja diferente de NULO, existe mais de um nó na lista, então o AUX1 chama o AUX2, fala para ele o endereço do nó em que ele se encontra e vai até o próximo nó (Figura 16, A).

4. Ao chegar no próximo nó, o AUX1 repete a mesma verificação feita no nó anterior, chama o AUX2 para o endereço do nó em que se encontra e passa para o próximo nó, este processo se repete até que o AUX1 encontre um nó do qual seu próximo seja igual a NULO, ou seja, o último elemento da lista (Figura 16, B).
5. O AUX1 remove o último nó da lista.
6. O AUX2 informa ao nó que seria o penúltimo da lista que agora este passa a ser o último, e que deve armazenar o valor NULO como próximo (Figura 16, B).

A figura 16 ilustra a remoção do último nó em uma lista com 4 elementos. Como descrito anteriormente os auxiliares, iniciando pelo AUX1, percorrem a lista a partir do CL em B1 até chegar a posição do último nó em A4, o AUX1 permanece sempre um nó a frente de AUX2 que para na posição A3 (A). Em (B), o AUX1 informa ao nó 4 que o mesmo será removido da lista e o AUX2 informa ao nó do endereço A3 que ele se tornará o último nó da lista e que deverá armazenar o valor NULO como referência de próximo.

Figura 16 – Removendo último elemento da em uma lista com mais de um elemento

Fonte: Autoria própria, 2019

Diferente das operações anteriores, esta operação possui um pouco mais de complexidade, pela necessidade de utilizar dois estudantes como elementos auxiliares, e ter que percorrer toda a lista. É importante frisar que não é aconselhável que todos os estudantes sejam armazenados na lista, já que para operações como esta, faz-se necessário que haja dois estudantes livres para poderem representar os auxiliares.

3.3.6 Operação remover determinado elemento da lista

A remoção de um determinado elemento numa lista dinâmica é feita através de uma função que, assim como na operação de remover no final, percorre a lista em busca do nó que se deseja remover, mas em vez de simplesmente percorrer a lista até encontrar o último nó, a função efetua a busca utilizando algum critério de comparação definido no início da operação, que neste caso utiliza-se o dado que o elemento carrega como critério de comparação. São utilizados dois elementos auxiliares para percorrerem a lista em busca do elemento se deseja remover, o qual pode estar no início, no meio, no final ou nem estar contido da lista (BACKES, 2017). A Figura 17 mostra uma função que remove determinado elemento da lista, escrito na linguagem C.

Figura 17 – Código da função que remove um determinado elemento da lista

```
int remove_qualquer(Lista* li, int dado){
    if(li == NULL)
        return 0;
    if((*li) == NULL)
        return 0;
    Elem *ant, *no = *li;
    while(no != NULL && no->dados.dado != dado){
        ant = no;
        no = no->prox;
    }
    if(no == NULL)
        return 0;

    if(no == *li)
        *li = no->prox;
    else
        ant->prox = no->prox;
    free(no);
    return 1;
}
```

Fonte: Autoria própria, 2019

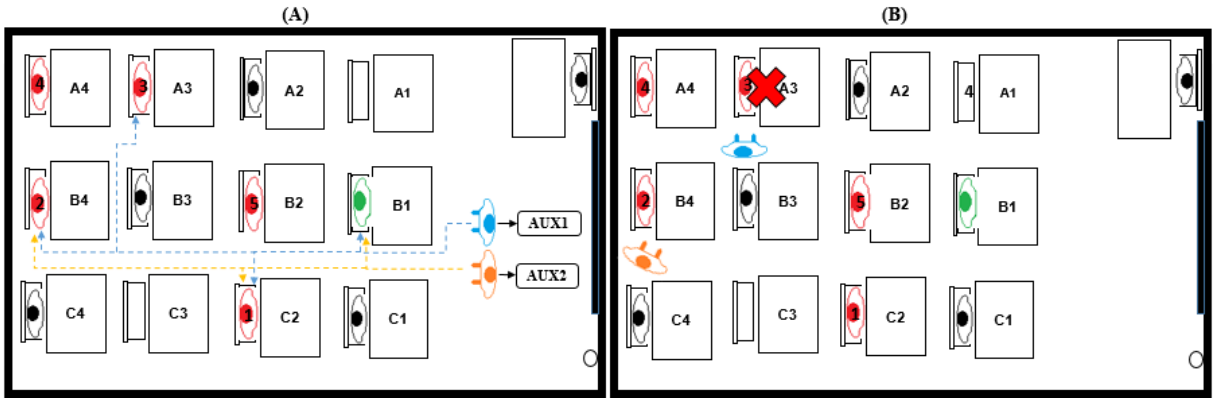
A operação de remover determinado elemento da lista está ilustrada (Figura 18), com materiais manipuláveis, da seguinte forma:

1. Para esta operação são necessários dois auxiliares para percorrerem a lista, AUX1 e AUX2. O SO escolhe dois participantes para ocuparem estes papéis.
2. O SO entrega para o AUX1 um papel com o dado que se deseja eliminar da lista.
3. O AUX1 vai até o CL, pega o endereço do primeiro nó da lista e armazena em seu cartão.
4. Neste ponto podemos ter as seguintes situações:
    a. Se o valor do endereço for NULO, então a lista está vazia e a operação é encerrada.
    b. Caso contrário, o AUX1 vai até o primeiro nó da lista e compara o dado que deve ser removido com o dado que o nó tem armazenado.
5. Se o dado não corresponder, o AUX1 fala para o AUX2 vir até o endereço do nó em que se encontra e pega com este nó o endereço para o próximo, caso ele exista, e anota em seu cartão.
6. O AUX1 vai ao próximo nó e repete o processo de comparação, caso não obtenha sucesso, o processo é repetido a partir do passo anterior. Esta execução se repete até o AUX1 encontrar um nó que possua correspondência, ou até alcançar o último nó da lista, e não obtiver nenhuma equivalência (Figura 18, A).
7. Se o AUX1 encontrar um nó que possua correspondência, então deve-se verificar:
    a. Se for o primeiro da lista, seu endereço é igual ao endereço indicado pelo CL, então:
        i. O nó levanta-se liberando a posição de memória que estava ocupando, e volta para frente da turma.
        ii. O AUX volta para o CL e fala que ele agora deve armazenar o endereço que anotou, que neste caso é o valor NULO, e a operação é encerrada.
    b. Se for em outra posição da lista, então o AUX1 fala para o AUX2 o endereço armazenado na variável *PROX* do nó que será eliminado (Figura 18, B).
8. O AUX2 informa ao nó em que se encontra, que agora ele deve armazenar o endereço que foi informado pelo AUX1.
9. O AUX1 remove o nó em que se encontra, o estudante que ocupa esta posição então levanta-se liberando a posição de memória que estava ocupando e volta para frente da turma encerrando a operação.

A figura 18 mostra a execução da operação de remoção de um determinado elemento em uma lista de tamanho 5. Em (A), os auxiliares devem percorrer a lista em busca do nó que possui o dado que se deseja eliminar. O AUX1 realiza as comparações em cada elemento e

permanece sempre um nó a frente do AUX2. Em (B), quando o nó que se deseja excluir é encontrado pelo AUX1, na posição A3, o mesmo informa para o AUX2 que o nó presente nesta posição tem como o seu endereço de próximo a posição A4, então o AUX2 informa ao nó na posição B4 que agora o nó posterior a ele está na posição A4, o nó 3 então é removido, liberando a posição de memória A3.

Figura 18 – Removendo um determinado nó em uma lista com 5 elementos



Fonte: Autoria própria, 2019

Esta operação de remoção se diferencia das anteriores, pelo fato de haver uma condição específica de exclusão, de forma que o elemento a ser excluído pode não estar contido na lista ou estar no início, final, ou entre dois elementos. Observa-se que esta operação de remoção apresenta algumas características vistas nas anteriores, mas com alguns passos a mais, principalmente no caso em que o elemento a ser removido se encontrar no meio da lista. Portanto, como forma de facilitar a execução desta, recomenda-se apresenta-la somente após os estudantes terem passado pelas operações de remover no início e no final da lista. Para que assim, com a experiência adquirida nas operações anteriores, os mesmos consigam executá-la mais facilmente, como também, poder observar melhor as características que estas operações têm em comum e de diferente.

3.3.7 Operação consultar determinado elemento da lista

A operação de consultar um determinado elemento na lista é basicamente uma função que percorre a lista para recuperar alguma informação armazenada em um determinado elemento, assim como na operação de remover um determinado elemento, ela é feita utilizando

algum critério de comparação obtido no início da operação. Em geral uma função de consulta numa lista dinâmica pode ser feita por posição ou por conteúdo (BACKES, 2017).

Na busca por posição, o algoritmo percorre a lista até encontrar um elemento na posição desejada, já na busca por conteúdo, é feita comparando o conteúdo armazenado no nó, com alguma informação de pesquisa, que pode ser, por exemplo, o número de matrícula de um estudante (BACKES, 2017). Para este trabalho, decidiu-se por representar uma consulta feita com base na posição. A seguir, a Figura 19 mostra o código escrito linguagem de programação C, com a função que faz a consulta de um determinado elemento dado sua posição.

Figura 19 – Código da função que consulta elemento da lista por posição

```
int consulta_posicao(Lista* li, int pos, struct aluno *al){
    if(li == NULL || pos <= 0)
        return 0;
    Elem *no = *li;
    int i = 1;
    while(no != NULL && i < pos){
        no = no->prox;
        i++;
    }
    if(no == NULL)
        return 0;
    else{
        *al = no->dados;
        return 1;
    }
}
```

Fonte: Autoria própria, 2019

A operação de consultar determinado elemento da lista está ilustrada (Figura 20), com materiais manipuláveis, da seguinte forma:

1. O SO escolhe um participante para ser um elemento auxiliar, AUX, e entrega um papel contendo o número da posição do elemento que o auxiliar deve encontrar.
2. O AUX cria em seu cartão um contador, que recebe inicialmente o valor 1, ele vai até o CL, pega o endereço do primeiro nó da lista e armazena-o.
3. Com esta informação o AUX deve observar as seguintes opções:
    a. Caso a lista esteja vazia, ou seja, o endereço passado pelo CL é igual a NULO, a operação é encerrada.
    b. Caso contrário, o AUX vai até o primeiro nó da lista e verifica se o valor do contador corresponde ao valor passado pelo SO.
4. Se o valor não corresponder, ele pega com o nó atual o endereço do próximo nó, caso ele exista, e anota em seu cartão.
5. O AUX vai até o próximo nó, incrementa o seu contador com +1 e repete o processo de comparação, caso não obtenha sucesso, repete-se o processo a partir do passo anterior.

Esse processo se repete até que AUX encontre o nó da posição desejada, ou até chegar ao último nó da lista, e não obtiver nenhuma equivalência, encerrando a operação.

6. Caso o AUX encontre o nó da posição desejada, ele pede ao aluno que representa este elemento para que revele para a turma o valor do dado que este possui.

A Figura 20 demonstra a operação de consultar um elemento pela posição em uma lista com 5 elementos. O SO passa para o AUX (em azul) o número da posição que se deseja pesquisar. Em (A), o AUX percorre a lista a partir do CL até a quantidade de nós necessária para chegar a posição escolhida, que neste caso seria o terceiro elemento na posição A4. Ao chegar na posição desejada, o AUX pede ao nó para revelar para a turma o dado que ele tem armazenado (B).

Figura 20 – Consultar um elemento pela posição

Fonte: Autoria própria, 2019

Assim como na operação de excluir determinado elemento, que faz uma busca na lista com base no dado que cada nó tem armazenado, esta operação também realiza uma busca, só que usando como parâmetro a posição do respectivo elemento na lista. Esta operação pode ser utilizada em diferentes momentos da dinâmica mas recomenda-se que haja no mínimo três elementos armazenados na lista para facilitar o entendimento dos estudantes.

4 CONSIDERAÇÕES FINAIS

O presente trabalho apresentou o desenvolvimento de uma abordagem didática para ser utilizada na disciplina de Estrutura de Dados como forma de apoio ao ensino de listas, pilhas e filas. A abordagem descrita possui atividades baseadas na computação desplugada com o objetivo de criar uma representação simples e lúdica do tema.

Através do trabalho de revisão bibliográfica realizado neste estudo, foi possível identificar o problema da pesquisa, que apontava para a existência de dificuldades enfrentadas por estudantes em disciplinas que lidam com programação, lógica e abstração. Através da análise dos estudos obtidos, foi possível concluir que a utilização de métodos desplugados mostra-se como uma alternativa para a amenizar problemas como baixo rendimento dos estudantes, falta de motivação e abandono de curso, o que vem por justificar o uso desta metodologia como base para a formulação das atividades que compõe a dinâmica apresentada nesta abordagem. Desta forma a execução da revisão bibliográfica e a elaboração da abordagem didática, juntamente com a criação dos materiais manipuláveis, concretizam os objetivos definidos no início da pesquisa.

Com base nos estudos apresentados sobre o tema, este trabalho contribui para a pesquisa e desenvolvimento de novas metodologias de ensino, como forma de aprimorar o processo de ensino e aprendizagem, para serem utilizados na formação de novos profissionais para um mercado que está em constante evolução. A abordagem didática tem como diferencial a utilização de elementos presentes em sala de aula e no cotidiano dos estudantes, e de materiais de baixo custo e fácil confecção. Uma característica positiva da abordagem é poder diminuir a complexidade das operações praticadas nas listas, tornando mais acessível seu entendimento aos estudantes, além de fazer com que eles possam criar uma relação lógica entre os códigos das operações em linguagem de programação e as atividades desenvolvidas durante a execução da dinâmica.

Este trabalho limitou-se a apresentar uma abordagem didática para ser utilizada em sala de aula como ferramenta auxiliar ao ensino de estrutura de dados do tipo lista simples e dinamicamente encadeada. A principal dificuldade encontrada foi conseguir traduzir, de forma clara e concisa, a mesma lógica presente nas funções descritas em código de linguagem de programação, para o formato de dinâmica de grupo.

Como sugestão para trabalhos futuros destaca-se: a aplicação da abordagem em uma turma da disciplina de Estrutura de Dados, seguindo o plano de aulas disponível no Apêndice

B; adicionar outros conteúdos relacionados como estruturas do tipo lista duplamente encadeada e lista circular, seguindo o modelo descrito; adaptar o modelo proposto para o ensino de algoritmos de ordenação e busca.

REFERÊNCIAS

ALMEIDA, H. M. A didática no ensino superior: práticas e desafios. **Revista Estação Científica**, Juiz de Fora, n. 14, jul. / dez. 2015. Disponível em: http://portal.estacio.br/docs%5Crevista_estacao_cientifica/07-14.pdf. Acesso em: 18 set. 2019.

UNIVERSIDADE ESTADUAL DA PARAÍBA. Projeto Pedagógico de Curso PPC: Computação (Bacharelado). **Universidade Estadual da Paraíba CCEA**, Núcleo docente estruturante, EDUEPB. Patos, 2016.

ASCÊNCIO, A.F.G.; CAMPOS, E.A.V. **Fundamentos da Programação de Computadores**. São Paulo: Editora Pearson Prentice Hall, 2007.

BACKES, A. **Estrutura de Dados descomplicada em linguagem C**. 1. ed. Rio de Janeiro: Elsevier, 2017.

BARBOSA, L. S.; FERNANDES, T. C. B.; CAMPOS, A. M. C. Takkou: uma Ferramenta Proposta ao Ensino de Algoritmos. **Workshop de Educação em Computação (WEI)**, Natal, 2011. Disponível em: http://www.dimap.ufrn.br/csbc2011/anais/eventos/contents/WEI/Wei_Secao_1_Artigo_3_Barbosa.pdf. Acesso em: 20 jun. 2019.

BELIZARIO, B. N.; OLIVEIRA, J. V. S; JUNIOR, O. P. A. **Ensino de Programação nas Escolas Públicas: criando novos gênios**. Campos dos Goytacazes: Cadernos de Extensão do Instituto Federal Fluminense, v.2, p. 169-180. 2016. Disponível em: <http://www.essentiaeditora.iff.edu.br/index.php/cadernos_de_extensao/article/view/7645/5272>. Acesso em: 12 nov. 2018.

BELL, T.; ALEXANDER, J.; FREEMAN, I.; GRIMLEY, M. Computer science unplugged: School students doing real computing without computers. **The New Zealand Journal of Applied Computing and Information Technology**, v. 13, n. 1, p. 20-29, 2009. Disponível em: http://grorichome.dyndns.org/oldsite/groricorssgoo/web/pdf/unplugged.pdf. Acesso em. 17 out. 2018.

BELL, T. WITTEN, I. H.; FELLOWS, M.; ADAMS, R.; MCKENZIE, J. **Ensinando Ciência da Computação sem o uso do computador**. Computer Science Unplugged ORG, 2011.

BERBEL, N. A. N. **As metodologias ativas e a promoção da autonomia de estudantes**. Ciências Sociais e Humanas, Londrina, v. 32, n. 1, p. 25-40, jan. /jun. 2011. Disponível em: http://www.proiac.uff.br/sites/default/files/documentos/berbel_2011.pdf. Acesso em: 20 ago. 2019.

BORDINI, A.; AVILA, C. M. O.; WEISSHAHN, Y.; CUNHA, M. M.; CAVALHEIRO, S. A. C.; FOSS, L.; AGUIAR, M. S.; REISER, R. H. S. Computação na educação básica no Brasil: o estado da arte. **Revista de Informática Teórica e Aplicada**, v. 23, n. 2, p. 210-238, 2016. Disponível em: https://www.seer.ufrgs.br/rita/article/view/RITA-VOL23-NR2-210. Acesso em. 17 out. 2018.

BORGES, T. S.; ALENCAR, G. Metodologias ativas na promoção da formação crítica do estudante: o uso das metodologias ativas como recurso didático na formação crítica do estudante do ensino superior. **Cairu em Revista**, n. 04, p. 119-143, jul. / ago. 2014. Disponível em: https://ufsj.edu.br/portal2-repositorio/File/napecco/Metodologias/Metodologias%20Ativas%20na%20Promocao%20da%20Formacao.pdf. Acesso em: 17 out. 2018.

BROOKSHEAR, J. G. **Ciência da Computação: Uma Visão Abrangente**. Bookman Editora, 2003.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Algoritmos - Teoria e Prática**. 2. ed. Rio de Janeiro: Campus. 2002.

COSTA, T. L. S.; SOUZA, F. V. C.; COSTA, W. E**. O uso de Computação Desplugada para apoiar a Aprendizagem de Algoritmos de Ordenação e Tabela Hash**. Trabalho de Conclusão de Curso - Universidade Federal da Paraíba (UFPB), Rio Tinto, 2017. Disponível em: https://repositorio.ufpb.br/jspui/bitstream/123456789/3344/1/TLSC14062017.pdf. Acesso em: 12 mai. 2019.

DENNING, P. J. Computing is a natural Science. **Communications of the ACM**, v. 48, n. 4, p. 27-31, abr. 2005. Disponível em: https://dl.acm.org/citation.cfm?id=1053309. Acesso em: 24 out. 2018.

FONSECA FILHO, C. **História da computação: O Caminho do Pensamento e da Tecnologia**. Porto Alegre: EDIPUCRS, 2007.

FREIRE, P. **Pedagogia da Autonomia - Saberes Necessários à Prática Educativa**. 25 ed. Editora Paz e Terra. Coleção Saberes, 1996.

GEMIGNANI, E. Y. M. Y. Formação de Professores e Metodologias Ativas de Ensino-Aprendizagem: Ensinar para a Compreensão. **Revista Fronteiras da Educação**, Recife, v. 1, n. 2, 2012. Disponível em: http://fronteirasdaeducacao.org/index.php/fronteiras/article/view/14/22. Acesso em: 17 fev. 2019.

GOMES, A.; HENRIQUES, J.; MENDES, A. Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. **Educação, Formação & Tecnologias**, v. 1, n. 1, p. 93-103, mai. 2008. Disponível em http://eft.educom.pt. Acesso em: 18/12/2018.

HAYDT, R. C. C. **Curso de didática geral**. 1 ed. São Paulo: Ática, 2011.

LIBÂNEO, J. C. **Didática**. São Paulo: Cortez Editora, 2006.

MARTINHAGO, A. Z.; SMARZARO, R.; LIMA, I.; GUIMARÃES, L. Computação Desplugada no Ensino de Bancos de Dados na Educação Superior. **WEI-XXII Workshop sobre Educação em Computação**, 2017. Disponível em: http://www. lbd. dcc. ufmg. br/colecoes/wei/2014/003. pdf. Acesso em. 17 out. 2018.

MARTINS, E. R. **Fundamentos da Ciência da Computação**. Ponta Grossa: Atena Editora, 2019. *E-book.*

MEC. **Diretrizes Curriculares Nacionais para os Cursos de Graduação em Computação (DCN16)**. Resolução CNE/CES nº 5, nov. 2016.

MOREIRA, J. A; MONTEIRO, W. M. O uso da computação desplugada em um contexto de gamificação para o ensino de estrutura de dados. **RENOTE**, v. 16, n. 2, 2018. Disponível em: https://www.seer.ufrgs.br/renote/article/view/89272. Acesso em: 03 ago. 2019.

PILETTI, C. **Didática geral**. 23. ed. São Paulo: Ática, 2004.

SANTANA, T. S.; RIBEIRO, N. C. S.; PRIETCH, S. S. A Utilização da Animação Digital no Processo de Ensino-Aprendizagem de Estrutura de Dados**. VIII Workshop de Educação e Informática (VII WEIBASE)**, p. 1-9, 2009. Disponível em: http://twixar.me/cpC1. Acesso em: 20 jun. 2019.

SANTIAGO, A. D.; KRONBAUER. Um modelo lúdico para o ensino de conceitos de programação de computadores. **Revista Brasileira de Informática na Educação – RBIE**, v.25, n.3. 2017. Disponível em: https://br-ie.org/pub/index.php/sbie/article/view/6723. Acesso em: 03 ago. 2019.

SANTOS, R. P.; COSTA, H. A. X. Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos iniciantes em Computação e Informática. **INFOCOMP**, v. 5, n. 1, p. 41-50, 2006. Disponível em: http://infocomp.dcc.ufla.br/index.php/INFOCOMP/article/view/121. Acesso em: 17 out. 2018.

SANTOS, R. P.; COSTA, H. A. X. TBC-AED: Um Software Gráfico para Apresentação de Algoritmos e Estruturas de Dados aos Iniciantes em Computação e Informática. **I Congresso de Computação do Sul do Mato Grosso (COMPSULMT)**, Rondonópolis, 2005. Disponível em: https://www.cos.ufrj.br/~rps/pub/completos/2005/COMPSULMT.pdf. Acesso em: 25 jun. 2019.

SILVA, J. C. **Ensino de programação para alunos do ensino básico: um levantamento das pesquisas realizadas no Brasil**. Trabalho de Conclusão de Curso, Universidade Federal da Paraíba, Rio Tinto, 2017. Disponível em: https://repositorio.ufpb.br/jspui/bitstream/123456789/3328/1/JCS14062017.pdf. Acesso em: 12 nov. 2018.

SILVA, T. R.; MEDEIROS, T. J.; LOPES, R.; ARANHA, E.; MEDEIROS, H. Ensino-aprendizagem de programação: uma revisão sistemática da literatura. **Revista Brasileira de Informática na Educação**, v. 23, n. 1, 2015. Disponível em: https://www.researchgate.net/profile/Thiago_Reis5/publication/281264303_Ensino-aprendizagem_de_programacao_uma_revisao_sistematica_da_literatura/links/55e46b1908ae6abe6e901f90.pdf. Acesso em: 17 fev. 2019.

SOUZA, C. C.; MEDEIROS, T. R; SOUSA, T. D. N.; GADELHA, R. N. S.; SILVA, E. L.; AZEVEDO, R. R.; COSTA, E. B. Um Ambiente Integrado de Simulação para Auxiliar o Processo de Ensino/Aprendizagem da Disciplina de Estrutura de Dados. **Anais do XXII**

SBIE - XVII WIE, Aracaju, nov. 2011. Disponível em: https://www.br-ie.org/pub/index.php/sbie/article/view/1607. Acesso em: 20 jun. 2019.

VALENTE, J. A. **O computador na sociedade do conhecimento**. Campinas: Unicamp/NIED, 1999.

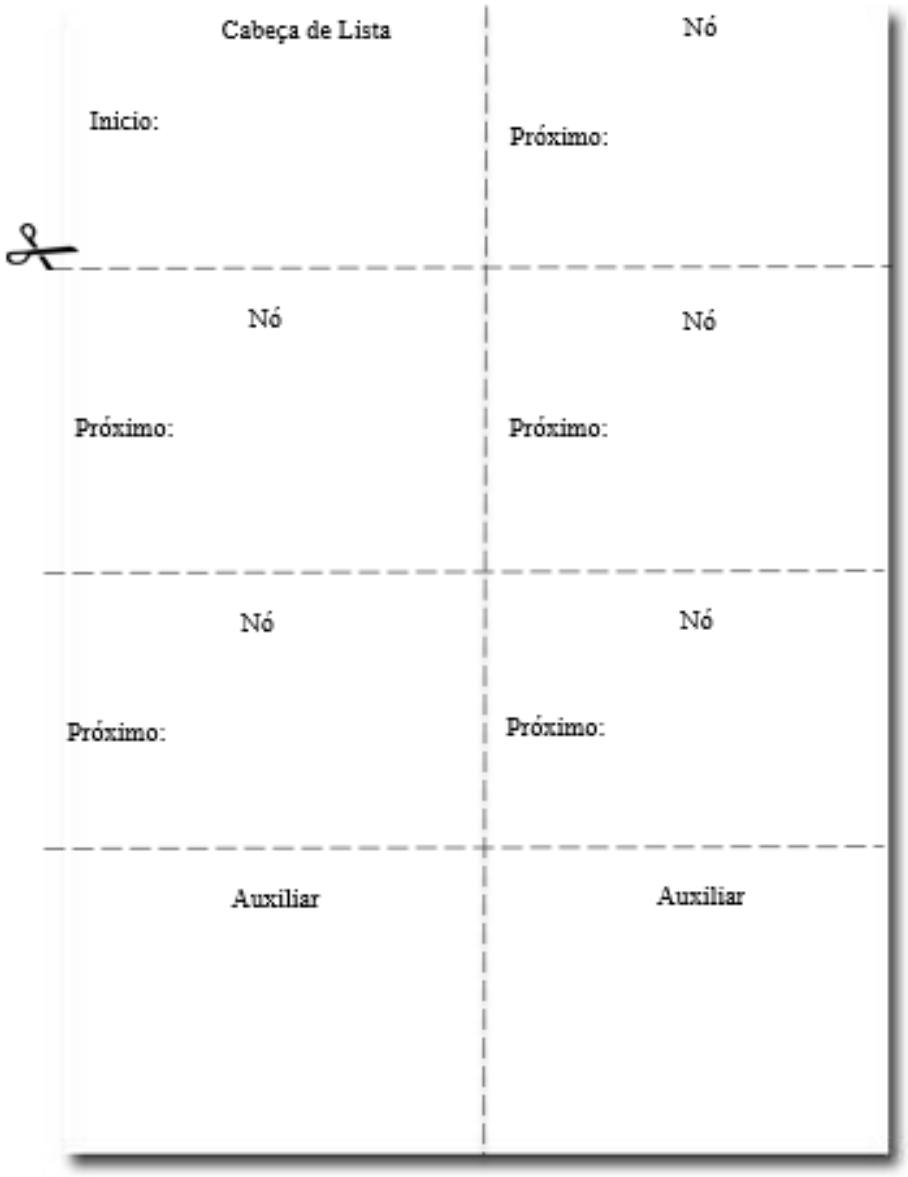
VOSS, G. B.; POZZEBON**,** R. C. B.; DENARDI, R. N.; FERREIRA, C. N. Proposta de um Jogo Educacional para o Ensino de Estrutura de Dados**. Anais dos Workshops do Congresso Brasileiro de Informática na Educação (CBIE)**, vol. 4, n. 1, p. 1184, 2015. Disponível em: https://br-ie.org/pub/index.php/wcbie/article/view/6271. Acesso em: 25 jun. 2019.

ZABALA, A. **A prática educativa: Como ensinar**. Porto Alegre: Artmed, 1998.

APÊNDICE A – MATERIAIS NECESSÁRIOS AO DESENVOLVIMENTO DA METODOLOGIA AUXILIAR PARA O ENSINO DE ESTRUTURA DE DADOS

A Figura 21 exibe o formato e as dimensões dos cartões utilizados pelos participantes durante a dinâmica. Este modelo deve ser utilizado para uma execução com 8 participantes, caso haja um número maior, deve-se utilizar uma folha adicional somente com o modelo referente aos nós.

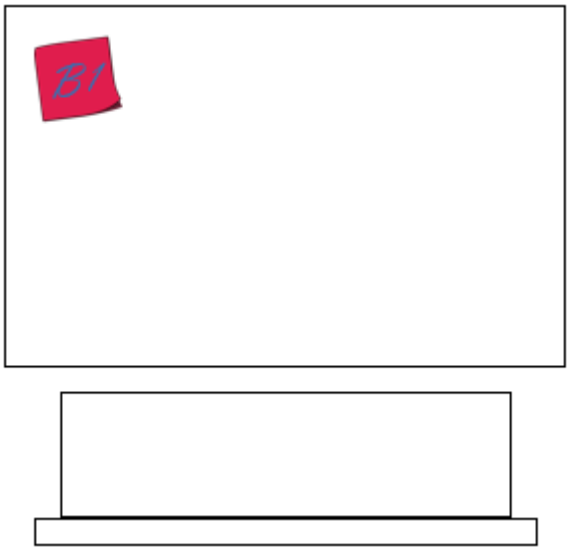
Figura 21 – Modelo dos cartões em folha A4



Fonte: Autoria própria, 2019

A Figura 22 mostra uma representação da maneira como as cadeiras devem ser identificadas na sala, utilizando os papeis adesivos, comumente conhecidos como *post-its*.

Figura 22 – Representação identificação das cadeiras



Fonte: Autoria própria, 2019

APÊNDICE B – PLANO DE AULA

ÁREA DO CONHECIMENTO/TEMA
Estrutura de Dados: “Listas Dinâmicas Encadeadas”.

PROFESSOR
Professor da disciplina

OBJETIVOS
GERAL • Capacitar o aluno para compreender o funcionamento de estruturas de dados do tipo listas, pilhas e filas.
ESPECÍFICOS • Apresentar os conceitos que envolvem estruturas de dados do tipo listas, pilhas e filas. Seu funcionamento e suas operações; • Mostrar o funcionamento destas estruturas em código de linguagem de programação; • Aplicar a abordagem didática envolvendo os conceitos apresentados.

CONTEÚDO
Estruturas de dados do tipo listas, pilhas, filas e suas operações.

METODOLOGIA
• Aula expositiva dialogada com explicação detalhada sobre a temática; • A aula contemplada pelo tema em questão utilizar-se-á de projeção com slides, pincel e quadro branco para demonstração dos conceitos e dos códigos em linguagem de programação; • Aplicação da abordagem didática em sala de aula, como forma de reforçar os conceitos apresentados, aliando a teoria à prática.

AVALIAÇÃO
• A avaliação da aprendizagem será verificada mediante aplicação de lista de exercício para que o estudante ponha em prática os conhecimentos abordados na aula.

REFERÊNCIAS
ASCÊNCIO, A.F.G.; CAMPOS, E.A.V. **Fundamentos da Programação de Computadores**. São Paulo: Editora Pearson Prentice Hall, 2007. BACKES, A. **Estrutura de Dados descomplicada em linguagem C**. 1. ed. Rio de Janeiro: Elsevier, 2017. CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Algoritmos - Teoria e Prática**. 2. ed. Rio de Janeiro: Campus. 2002.