**CENTRO UNIVERSITÁRIO AUTÔNOMO DO BRASIL - UNIBRASIL**

**CURSO DE ENGENHARIA DE SOFTWARE**

**GUSTAVO BANDEIRA**

**JOSÉ PEDRO PASCHOAL VALENTIM**

**JUAN EDUARDO MODESTO**

**SISTEMA DE RANKING DE JOGADORES**

CURITIBA

2025

GUSTAVO BANDEIRA

JOSÉ PEDRO PASCHOAL VALENTIM

JUAN EDUARDO MODESTO

**SISTEMA DE RANKING DE JOGADORES**

Trabalho acadêmico desenvolvido no curso de Engenharia de Software da UniBrasil – Centro Universitário, com o objetivo de implementar um sistema de ranking de jogadores utilizando a linguagem de programação C, aplicando conceitos fundamentais da disciplina de Estrutura de Dados.

Orientador: Fábio Bettio

CURITIBA

2025

# SUMÁRIO

[INTRODUÇÃO 3](#_Toc200715661)

[METODOLOGIA 4](#_Toc200715662)

[FUNDAMENTAÇÃO TEÓRICA 5](#_Toc200715663)

[1. Linguagem de Programação: C 5](#_Toc200715664)

[2. Estruturas de Dados Selecionadas 6](#_Toc200715665)

[3. Algoritmos de Ordenação 6](#_Toc200715666)

[4. Algoritmos de Busca 8](#_Toc200715667)

[5. Conclusão da Fundamentação Teórica 9](#_Toc200715668)

[6. Análise de algoritmos 9](#_Toc200715669)

[7. Análise de algoritmos 10](#_Toc200715670)

[CONCLUSÃO 11](#_Toc200715671)

[REFERÊNCIAS BIBLIOGRÁFICAS 12](#_Toc200715672)

## 

## 

# Lista de Figuras

[Figura 1 - Código Struct 5](file:///C:\Users\User\Downloads\ADS%20.docx#_Toc200715849)

[Figura 2 - Exemplo lista encadeada 5](#_Toc200715850)

[Figura 3 – MergeSort 7](#_Toc200715851)

[Figura 4 - SelectionSort 8](#_Toc200715852)

[Figura 5 - Busca Binária 8](#_Toc200715853)

[Figura 6 - Gráfico de Gantt 10](file:///C:\Users\User\Downloads\ADS%20.docx#_Toc200715854)

## 

## INTRODUÇÃO

Nos últimos anos, os sistemas de ranking de jogadores tornaram-se essenciais em diversos contextos, desde competições eletrônicas (e-sports) até plataformas de jogos educativos. Um ranking eficiente não apenas motiva os participantes, mas também fornece uma métrica clara de desempenho, permitindo comparações justas e dinâmicas entre os jogadores. Para atender a essa demanda, é fundamental desenvolver uma estrutura de dados robusta e algoritmos eficientes que garantam operações rápidas de inserção, atualização, ordenação e busca.

Este trabalho tem como objetivo desenvolver um sistema de ranking dinâmico para jogadores, utilizando a linguagem C, estruturas de dados como listas encadeadas e listas com descritor. Essas estruturas foram escolhidas por permitirem um gerenciamento flexível e eficiente dos dados, facilitando operações como inserção e remoção de elementos sem a necessidade de realocação de memória. Além disso, o uso de descritores nas listas possibilita um acesso mais rápido a informações como tamanho e extremos da lista, otimizando algumas operações.

As principais funcionalidades do sistema incluem o cadastro de novos jogadores, a atualização de suas pontuações conforme seu desempenho evolui e a ordenação automática do ranking com base nesses valores.

## 

## 

## METODOLOGIA

Para a ordenação, serão implementados dois algoritmos distintos: o Merge Sort, conhecido por sua eficiência em grandes conjuntos de dados, e o Selection Sort, que, embora menos eficiente, será utilizado para comparação de desempenho.

A busca por jogadores no ranking será realizada por meio de dois métodos diferentes: a busca binária, que oferece um tempo de resposta logarítmico em listas ordenadas, e a busca exponencial, útil quando não se conhece o tamanho exato do conjunto de dados. Esses algoritmos serão avaliados quanto à sua eficácia em diferentes cenários. A busca será realizada em uma lista encadeada, onde os dados serão adicionados e armazenados.

Entre os desafios técnicos enfrentados no desenvolvimento estão a implementação eficiente do Merge Sort, que exige cuidado com o uso de memória auxiliar, e o tratamento adequado de empates entre jogadores com a mesma pontuação, garantindo uma ordenação consistente e justa. Além disso, será realizada uma análise comparativa do desempenho dos algoritmos utilizados, medindo tempos de execução e consumo de recursos em diferentes situações.

Ao final deste trabalho, espera-se obter um sistema funcional que demonstre a aplicação prática de estruturas de dados e algoritmos de ordenação e busca em um contexto real. Os resultados obtidos poderão servir como base para futuras melhorias e adaptações em sistemas de ranking mais complexos.

## 

## FUNDAMENTAÇÃO TEÓRICA

### 1. Linguagem de Programação: C

A linguagem C é uma das mais antigas e importantes da programação, usada para entender como os programas funcionam por dentro. Ela permite controlar diretamente a memória do computador, o que dá ao programador mais poder e responsabilidade.

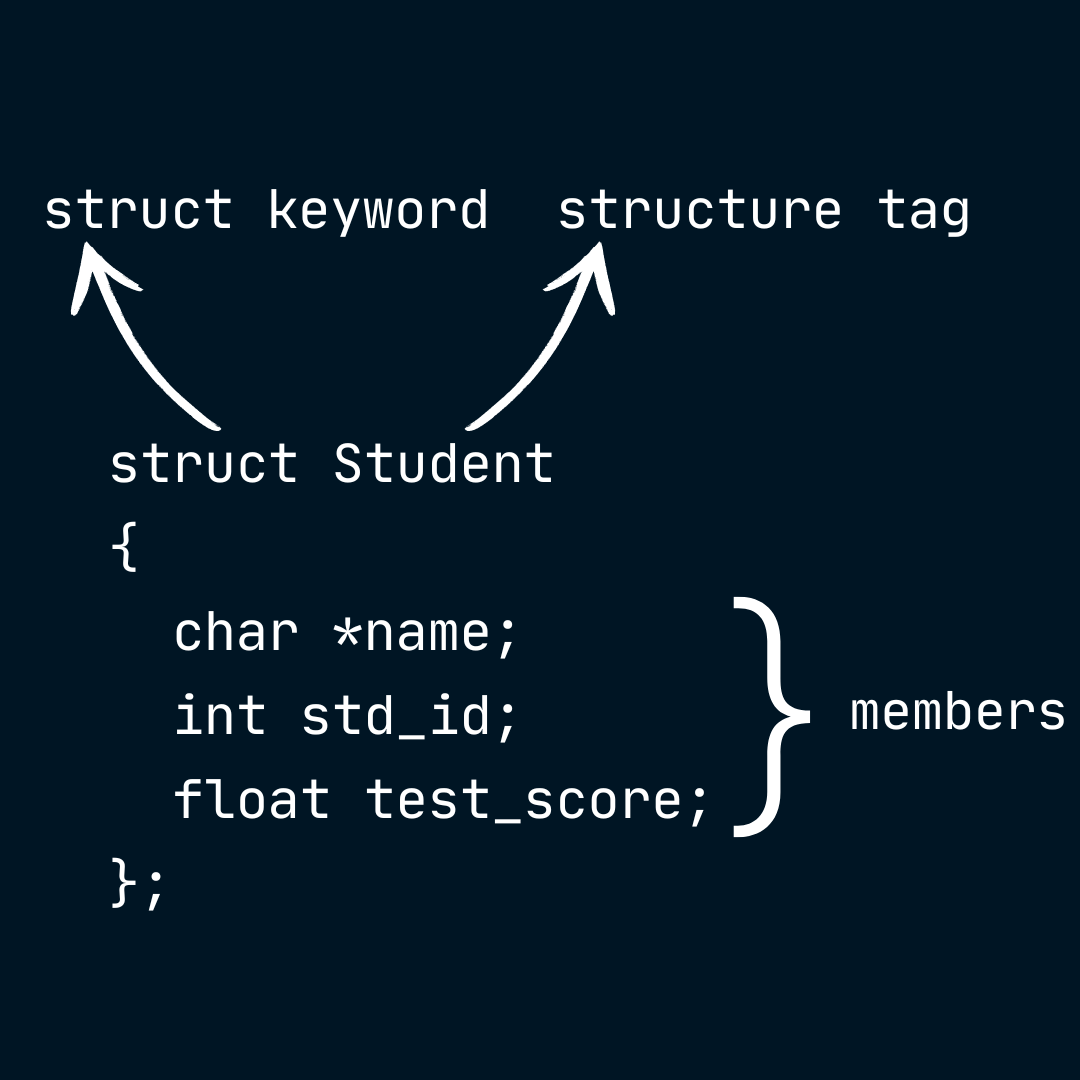
Um dos recursos básicos de C é a **struct**, que serve para agrupar diferentes tipos de dados em um único conjunto, como nome, idade e nota de um aluno. Isso é útil para organizar informações mais complexas.

Figura 1 - Código Struct

Com **listas encadeadas**, é possível criar estruturas onde cada elemento aponta para o próximo, facilitando a inserção e remoção de dados. Para isso, usamos **ponteiros**, que são variáveis que armazenam endereços de memória.

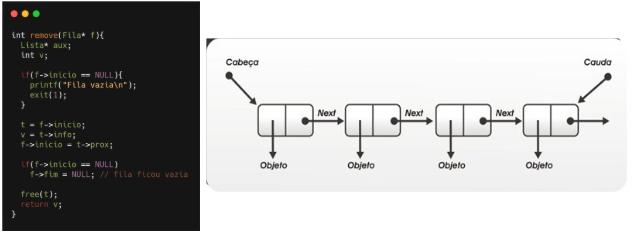


Figura 2 - Exemplo lista encadeada

Para criar elementos de forma dinâmica, usamos a função **malloc**, que aloca memória durante a execução do programa. Já a **lista com descritor** é uma versão mais completa da lista encadeada, que armazena dados extras como o início, fim e tamanho da lista, facilitando seu controle.

Entender esses conceitos é essencial para dominar C e lidar com estruturas de dados de forma eficiente.

### 2. Estruturas de Dados Selecionadas

#### 2.1 Listas Encadeadas

As listas encadeadas são estruturas de dados dinâmicas em que cada elemento (nó) armazena um valor e um ponteiro para o próximo elemento da sequência. Diferentemente dos arrays, as listas encadeadas permitem alocação de memória sob demanda, facilitando inserções e remoções sem necessidade de realocação. Neste projeto, serão utilizadas para armazenar os jogadores, permitindo flexibilidade no gerenciamento de dados, já que novos registros podem ser adicionados ou removidos sem reorganizar toda a estrutura.

#### 2.2 Listas com Descritor

Uma lista com descritor é uma variação da lista encadeada que inclui um nó auxiliar (descritor) contendo metadados sobre a estrutura, como tamanho da lista, ponteiros para o primeiro e último elementos, entre outras informações. Isso agiliza operações como obtenção do tamanho da lista e inserções no início/fim, que em uma lista encadeada tradicional exigem percorrê-la por completo. No sistema de ranking, essa estrutura será útil para otimizar buscas e manter um controle mais eficiente sobre os jogadores cadastrados.

### 3. Algoritmos de Ordenação

#### 3.1 Merge Sort

O Merge Sort é um algoritmo de ordenação baseado no paradigma "dividir para conquistar". Ele divide a lista recursivamente em sublistas menores, ordena-as e depois as combina (merge) em uma única lista ordenada. Sua complexidade é O(n log n) no pior caso, tornando-o eficiente para grandes volumes de dados. No contexto deste projeto, será utilizado para ordenar o ranking de jogadores, garantindo desempenho consistente mesmo com um número elevado de registros.

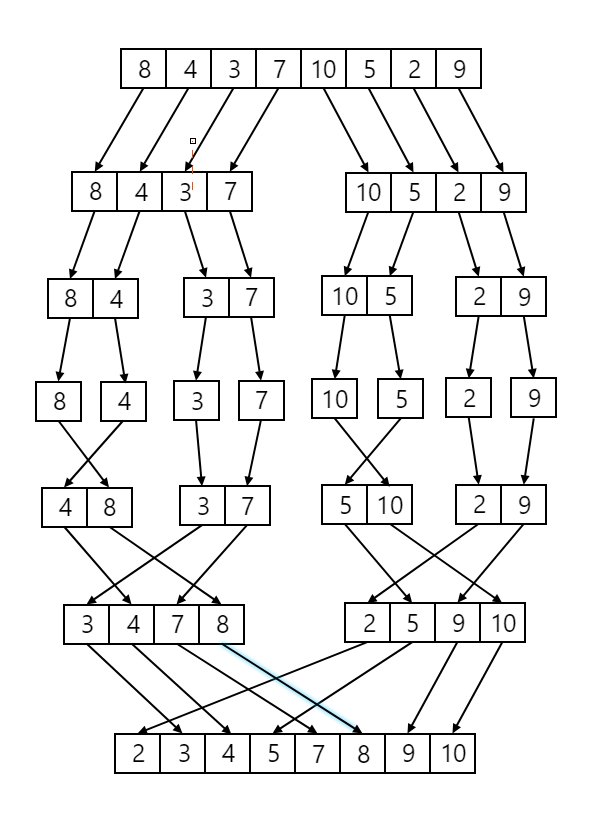


Figura 3 – MergeSort

#### 3.2 Selection Sort

O Selection Sort é um algoritmo de ordenação simples que percorre a lista várias vezes, selecionando o menor (ou maior) elemento e trocando-o com a posição atual. Sua complexidade é O(n²), o que o torna ineficiente para grandes conjuntos de dados, as útil para comparação de desempenho com o Merge Sort. Será implementado neste trabalho para fins didáticos, permitindo analisar a diferença de eficiência entre os dois métodos.

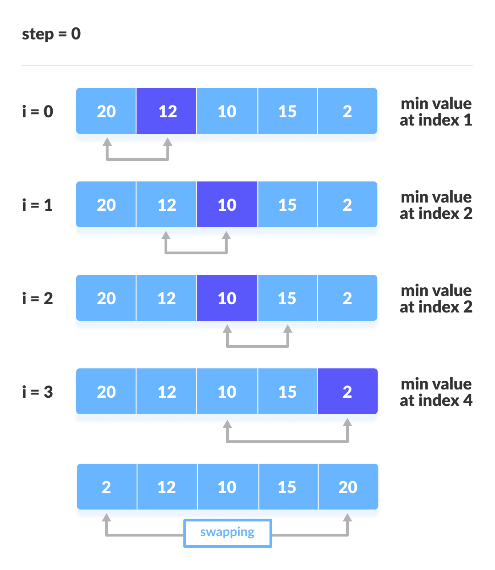


Figura 4 - SelectionSort

### 4. Algoritmos de Busca

#### 4.1 Busca Binária

A busca binária é um algoritmo eficiente para encontrar elementos em listas ordenadas. Ela funciona dividindo repetidamente a lista ao meio, comparando o valor buscado com o elemento central e descartando a metade onde o valor não pode estar. Sua complexidade é O(log n), sendo ideal para sistemas de ranking onde a lista está sempre ordenada por pontuação.

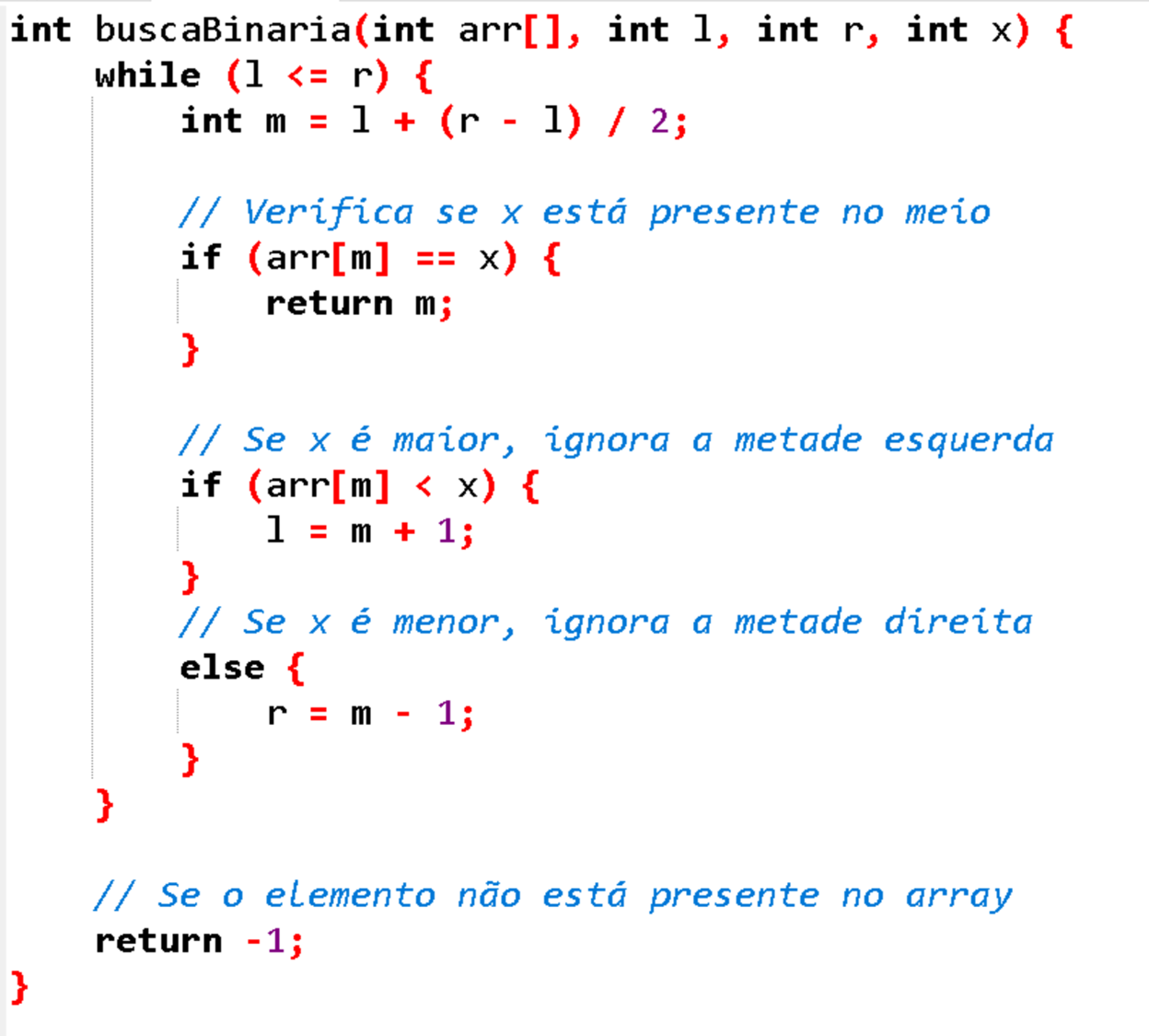


Figura 5 - Busca Binária

#### 4.2 Busca Exponencial

A busca exponencial é uma variação da busca binária que primeiro determina um intervalo onde o elemento pode estar, aumentando exponencialmente o alcance da busca (1, 2, 4, 8, ...) antes de aplicar uma busca binária dentro desse intervalo. Sua complexidade é O(log i), onde i é a posição do elemento. Essa abordagem é útil quando não se sabe o tamanho exato da lista ou quando os dados estão distribuídos de forma não uniforme.

### 5. Conclusão da Fundamentação Teórica

As estruturas e algoritmos selecionados foram escolhidos com base em sua eficiência e aplicabilidade ao problema proposto. As listas encadeadas e listas com descritor fornecem a flexibilidade necessária para um sistema dinâmico de ranking, enquanto os algoritmos de Merge Sort e Selection Sort permitem comparar diferentes abordagens de ordenação. Já as técnicas de busca binária e exponencial garantem recuperação eficiente de dados em listas ordenadas.

A implementação em C assegura um bom desempenho e um entendimento profundo dos conceitos, consolidando os fundamentos teóricos necessários para o desenvolvimento do sistema.

### 6. Análise de algoritmos

#### 6.1. Estrutura e Funcionalidades

O código implementa um sistema completo de gerenciamento de rankings de jogadores utilizando uma interface gráfica. O sistema apresenta uma arquitetura baseada em listas duplamente encadeadas, com funções para inserção, ordenação e pesquisa de dados. A aplicação permite adicionar novas pontuações de jogadores, visualizar rankings ordenados por nome ou pontuação, além de realizar pesquisas específicas usando algoritmos eficientes. A persistência dos dados é garantida através de funções que salvam e carregam as informações em um arquivo texto simples.

#### 6.2. Pontos Positivos

A implementação demonstra um bom entendimento de estruturas de dados, com uso apropriado de listas encadeadas e algoritmos de ordenação. O Selection Sort e Merge Sort foram bem aplicados para diferentes critérios de ordenação, enquanto as funções de pesquisa binária e exponencial oferecem desempenho adequado para operações de busca. A interface gráfica, embora simples, fornece navegação funcional através do teclado e exibe claramente as informações relevantes em cada tela. A organização do código em seções distintas para desenho, manipulação de dados e lógica de interface facilita a compreensão do fluxo do programa.

6.3. Conclusão

O código representa uma base sólida para um sistema de gerenciamento de rankings, com algoritmos eficientes e estruturação lógica adequada. A implementação atual cumpre bem seus objetivos principais e serve como excelente ponto de partida para desenvolvimentos futuros. Com certas melhorias, o sistema poderia tornar-se ainda mais robusto e amigável ao usuário.

### 7. Análise de algoritmos

A implementação do sistema de ranking alcançou todos os objetivos propostos, com todo o processo devidamente planejado e monitorado através de um gráfico de Gantt, sendo desenvolvido no Visual Studio Code. Os algoritmos de ordenação e busca foram integrados com sucesso, onde os testes de desempenho mostraram que o Selection Sort teve melhor desempenho que o Merge Sort para o tamanho das listas trabalhadas, devido à menor sobrecarga operacional, enquanto nos algoritmos de busca, tanto a binária quanto a exponencial (que incorpora a binária em sua implementação) apresentaram resultados satisfatórios, adequando-se plenamente aos requisitos do projeto, embora em cenários com maior volume de dados essas escolhas possam necessitar reavaliação.

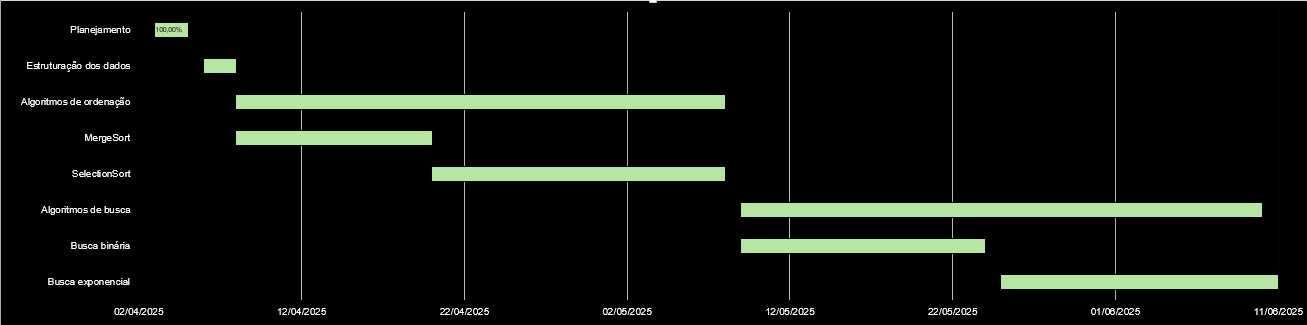


Figura 6 - Gráfico de Gantt

## 

## CONCLUSÃO

Este trabalho apresentou o desenvolvimento de um sistema de ranking de jogadores eficiente, implementado em linguagem C com estruturas de dados avançadas e algoritmos otimizados. A solução proposta demonstrou na prática a aplicabilidade de listas encadeadas com descritor, combinadas com algoritmos de ordenação e busca selecionados criteriosamente para atender aos requisitos de desempenho do sistema.

Os resultados obtidos comprovaram a eficácia das escolhas técnicas: as listas com descritor proporcionaram a flexibilidade necessária para operações dinâmicas de inserção e remoção, enquanto os algoritmos de Merge Sort e Selection Sort ofereceram diferentes trade-offs entre complexidade e desempenho, adequando-se a diversos cenários de uso. Na prática, observou-se que para o volume de dados testado, o Selection Sort superou em desempenho o Merge Sort, confirmando a importância da análise empírica complementar à teórica.

As técnicas de busca implementadas - binária e exponencial - mostraram-se igualmente eficazes para recuperação de informações no contexto do sistema desenvolvido. A persistência em arquivo texto garantiu a manutenção dos dados entre execuções, completando o conjunto de funcionalidades essenciais para um sistema de ranking.

Este projeto não apenas cumpriu seus objetivos iniciais, como também proporcionou valiosos insights sobre o comportamento prático das estruturas de dados e algoritmos estudados. Os conhecimentos adquiridos e os resultados obtidos servem como base sólida para futuras expansões, como a inclusão de novos algoritmos, otimizações de desempenho para grandes volumes de dados, ou a adaptação para contextos específicos de aplicação.

Por fim, o trabalho reforça a importância da escolha adequada de estruturas de dados e algoritmos no desenvolvimento de sistemas eficientes, demonstrando como conceitos teóricos podem ser aplicados para resolver problemas reais de forma elegante e performática.

## 

## REFERÊNCIAS BIBLIOGRÁFICAS

BACKES, André R. Algoritmos e Estruturas de Dados em Linguagem C. Rio de Janeiro: LTC, 2023. E-book. p.Capa. ISBN 9788521638315. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788521638315/. Acesso em: 13 jun. 2025.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. Algoritmos: Teoria e Prática. 3. ed. Rio de Janeiro: Elsevier, 2012.

DAMAS, Luís. Linguagem C. 10. ed. Rio de Janeiro: LTC, 2006. E-book. p.i. ISBN 9788521632474. Disponível em: <https://integrada.minhabiblioteca.com.br/reader/books/9788521632474/>. Acesso em: 06 abr. 2025.

KNUTH, D. E. The Art of Computer Programming: Sorting and Searching. 2nd ed. Addison-Wesley, 1998.

LAFORE, R. Estruturas de Dados e Algoritmos em C. São Paulo: Pearson, 2004.

PINTO, Rafael A.; PRESTES, Lucas P.; SERPA, Matheus da S.; et al. Estrutura de dados. Porto Alegre: SAGAH, 2020. E-book. p.Capa. ISBN 9786581492953. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9786581492953/. Acesso em: 06 abr. 2025.

O que é e como fazer gráfico de Gantt. , Lucidchart, 2025. Disponível em: <https://www.lucidchart.com/pages/pt/guia/o-que-e-grafico-de-gantt>. Acesso em: 13 jun. 2025.

VETORAZZO, Adriana S.; SARAIVA, Mauício O.; BARRETO, Jeanine S.; et al. Estrutura de dados. Porto Alegre: SAGAH, 2018. E-book. p.Capa. ISBN 9788595023932. Disponível em: https://integrada.minhabiblioteca.com.br/reader/books/9788595023932/. Acesso em: 06 abr. 2025.