Sistema de Gerenciamento de Mineradora Espacial

Willian Charantola da Costa

Universidade Federal de Mato Grosso do SUL, Campus Três Lagoas(UFMS-CPTL) - Três Lagoas-MS

w_charantola@ufms.com.br

Abstract. *This work presents the development of a Space Mining Management System, created as a practical application of data structure concepts using the C programming language. The system integrates a circular queue of astronauts, sorted linked lists of asteroids and missions, and multiple search, insertion, and update routines designed to organize and control fictional space-mining operations. Modules were implemented for registering, viewing, launching, and closing missions, along with file-persistence mechanisms for saving and loading system data. The proposed solution demonstrates the effective use of both static and dynamic data structures, together with good practices in modularization, memory management, and secure pointer handling. This paper describes the system's architecture, design decisions, and main challenges encountered during development.*

Resumo. *Este trabalho apresenta o desenvolvimento de um Sistema de Gerenciamento de Mineradora Espacial, construído como aplicação prática dos conteúdos da disciplina de Algoritmos e Programação II. O sistema integra diferentes componentes — incluindo uma fila circular de astronautas, listas encadeadas ordenadas de asteroides e missões, além de mecanismos de busca, inserção e atualização — com foco na organização e controle de operações espaciais. Foram implementados módulos para cadastro, consulta, lançamento e encerramento de missões, juntamente com rotinas de persistência em arquivos para armazenamento dos dados. A solução busca demonstrar, em um cenário fictício, a aplicação eficiente de estruturas de dados dinâmicas e estáticas, bem como boas práticas de modularização, encapsulamento e manipulação segura de memória. Este artigo descreve o funcionamento geral do sistema, suas decisões de projeto e os principais desafios enfrentados durante o desenvolvimento.*

1. Tema do cenário problema

Um tema que ganha cada vez mais relevância no cenário global é a crescente dependência da humanidade de minérios de terras raras, essenciais para a produção de tecnologias avançadas — desde equipamentos militares até processadores que impulsionam sistemas de inteligência artificial e smartphones. Um dos grandes desafios atuais é a concentração desses recursos em poucos países, especialmente a China, responsável por cerca de 80% da produção mundial. Essa centralização cria vulnerabilidades estratégicas, econômicas e geopolíticas, estimulando a busca por alternativas sustentáveis e seguras.

Nesse contexto, a exploração de asteroides surge como uma proposta promissora. Estudos conduzidos por agências espaciais, como a NASA e a ESA, indicam que muitos asteroides — sobretudo os do tipo metálico — possuem altas concentrações de elementos valiosos, incluindo platina, níquel, cobalto e diversas terras raras. Diferentemente da mineração terrestre, que exige grandes extensões de território e causa impactos ambientais significativos, a mineração espacial poderia oferecer acesso direto a depósitos quase puros desses minerais, formados há bilhões de anos durante os estágios iniciais do Sistema Solar.

A possibilidade de capturar, pousar ou redirecionar pequenos asteroides também já é discutida no âmbito da engenharia espacial, sendo considerada uma fronteira tecnológica capaz de redefinir cadeias produtivas inteiras. Empresas privadas e projetos internacionais de exploração já estudam a viabilidade de missões automatizadas para extração e transporte de minérios, tornando a mineração espacial um campo emergente de pesquisa multidisciplinar.

Inserido nesse cenário de inovação, este trabalho apresenta o **Sistema de Gerenciamento de Mineradora Espacial**, um programa implementado em linguagem C que simula as operações de uma empresa fictícia dedicada à mineração de asteroides. O sistema permite cadastrar asteroides, administrar missões espaciais e gerenciar uma fila circular de astronautas disponíveis para lançamento. Para isso, integra listas encadeadas, filas circulares e persistência de dados em arquivos texto, demonstrando a combinação prática entre estruturas de dados estáticas e dinâmicas.

Além disso, o projeto explora modularização, abstração, tratamento de erros, validação de entradas e manipulação direta de memória por meio de ponteiros. Assim, o sistema não apenas constrói uma narrativa ficcional inspirada em tendências reais da exploração espacial, mas também funciona como um exercício prático que consolida conceitos fundamentais da disciplina e reforça o domínio de estruturas de dados e organização de programas em C.

2. Proposta e TADs para a solução

2.1. Dados utilizados

Tendo em vista o problema apresentado, foram abstraídas 4 entidades principais que regem o sistema. Sendo elas os asteroides que serão explorados, os minérios encontrados em cada asteroide, os astronautas que serão enviados para realizar a mineração e as missões que serão realizadas. Abaixo são descritas todas as entidades utilizadas e os dados que cada uma possui.

Asteroide

Tabela 1. Especificação dos dados da entidade asteróide

Dado	Tipo	Observação
Código	int	Código único utilizado para identificação de cada asteroide
Distância	float	Distância em Km do asteroide em relação a terra
Diâmetro	int	Diâmetro em metros do asteroide
Quantidade de minérios	int	Quantidade de minérios diferentes que foram encontrados durante uma missão
Status	char	Status de ocupação do asteroide, podendo ser “Livre” ou “Ocupado” quando uma missão está naquele asteroide.
Minérios	minerio	Lista encadeada de minérios que foram encontrados em cada asteroide

Minérios

Tabela 2. Especificação dos dados da entidade minérios

Dado	Tipo	Observação
Código	int	Código único utilizado para identificação de cada minério
Nome	char	Nome descritivo de cada minério
Quantidade	int	Quantidade em toneladas de cada minério

Astronauta

Tabela 3. Especificação dos dados da entidade astronauta

Dado	Tipo	Observação
Código	int	Código único utilizado para identificação de cada astronauta
Nome	char	Nome de cada astronauta
Especialidade	char	Especialidade de cada astronauta, como “Analista”, “Biologia”, “Geologia”.

Missão

Tabela 4. Especificação dos dados da entidade missão

Dado	Tipo	Observação
Código	int	Código único utilizado para identificação de cada missão
Nome	char	Nome destinado a cada missão
Status	char	Situação atual de cada missão podendo ser “Montagem”, “Lançada” ou “Encerrada”.
Código do Asteroide	int	Código do asteroide para o qual a missão foi enviada
Prioridade	int	Prioridade dada a aquela missão e utilizada para ordenar as missões que serão lançadas primeiro
Duração	int	Duração em dias que aquela missão irá durar
Astronautas	astronauta	Vetor com 3 astronautas que serão enviados em cada missão

2.2 Estruturas de Dados Utilizadas

Após a definição das entidades do sistema, tornou-se necessário selecionar as estruturas de dados mais adequadas para representar cada uma delas, considerando as funcionalidades desejadas e os requisitos operacionais do projeto. A seguir, são apresentadas as TADs utilizadas, acompanhadas das respectivas justificativas.

Lista encadeada de asteroides

Dada a natureza potencialmente ilimitada do número de asteroides a serem cadastrados, optou-se por utilizar uma **lista encadeada simples**, que não impõe restrições quanto ao crescimento dinâmico da estrutura. Além disso, por não haver necessidade de ordenação prévia dos asteroides, cada novo elemento é inserido no **início da lista**, garantindo uma operação de inserção com complexidade **O(1)**.

Lista encadeada de minérios

Assim como os asteroides, a quantidade e variedade de minérios presentes em cada corpo celeste são imprevisíveis. Por esse motivo, cada nó da lista encadeada de asteroides contém um ponteiro para outra **lista encadeada**, responsável por armazenar os minérios associados àquele asteroide. Essa abordagem possibilita que cada asteroide possua um conjunto próprio e dinâmico de minérios, sem limites pré-definidos.

Lista encadeada ordenada de missões

As missões espaciais, que podem estar planejadas, em execução ou concluídas, são armazenadas em uma **lista encadeada ordenada**. Como não se deseja impor limites à quantidade de missões cadastradas, a lista deve ser capaz de crescer dinamicamente. Diferentemente das estruturas anteriores, esta lista é **ordenada com base no campo de prioridade**, fornecido pelo usuário no momento do cadastro. Essa ordenação facilita o gerenciamento das operações espaciais, garantindo que missões mais relevantes sejam acessadas antes das demais.

Fila circular de astronautas

A representação da equipe de astronautas utiliza um **vetor estático de 40 posições**, modelado como uma **fila circular**. Essa estrutura simula a limitação real de recursos humanos disponíveis para o lançamento de missões. A manipulação da fila é realizada por meio de índices de início e fim, permitindo inserções e remoções eficientes, além de maximizar o aproveitamento do espaço no vetor sem a necessidade de realocação.

2.3 Funcionalidades Implementadas

As funções implementadas no projeto foram divididas seguindo a sua interação com as entidades e TADS citadas anteriormente dentro de arquivos .C, como astronauta.c, asteroide.c e missoes.c . Desta forma, funções que interagem unicamente

com uma das entidades em específico estão implementadas juntas.

Além disso, algumas funções interagem com mais de uma ou até todas as TADS utilizadas. Essas funções foram chamadas de funções gerais e estão juntas no arquivo funcoes_gerais.c.

Abaixo são citadas os principais grupos de funções utilizados no projeto.

2.3.1 Funções para persistência de dados

Com o objetivo de assegurar a persistência das informações e facilitar a realização de testes, foram implementadas funções responsáveis pela leitura e escrita de dados em arquivos específicos localizados no diretório do projeto. Dessa forma, todas as TADS utilizadas pelo sistema têm seu conteúdo registrado em arquivos texto, permitindo que as informações sejam preservadas entre diferentes execuções do programa.

Os dados são armazenados nos arquivos **astronautas.txt**, **asteroides.txt** e **missoes.txt**. Ao iniciar o sistema, a primeira operação realizada é a tentativa de leitura desses arquivos, a fim de reconstruir o estado interno das estruturas de dados. Caso algum arquivo não exista — por exemplo, na primeira execução do sistema — o programa prossegue normalmente e o arquivo correspondente é criado automaticamente ao final da execução, garantindo a persistência para os próximos acessos.

2.3.2. Funções de manipulação básica das TADS

Para possibilitar a correta manipulação das estruturas de dados utilizadas no sistema, foram desenvolvidas funções responsáveis pelas operações fundamentais de inserção, remoção, atualização e consulta. Essas funções constituem a base lógica do projeto, garantindo que cada entidade seja tratada de forma organizada e consistente, preservando a integridade das TADs ao longo da execução.

As funções de inserção permitem incluir novos elementos nas estruturas de forma segura e eficiente. No caso dos asteroides e minérios, a inserção ocorre no início das listas encadeadas, o que assegura complexidade constante na operação. Já as missões são inseridas de maneira ordenada, de acordo com a prioridade definida pelo usuário, preservando a estrutura lógica da lista. A fila circular de astronautas, por sua vez, utiliza um vetor de tamanho fixo, no qual o gerenciamento dos índices de início e fim permite inserções sucessivas sem desperdício de espaço.

As operações de remoção possibilitam excluir elementos das TADs quando necessário, mantendo as estruturas íntegras. Em listas encadeadas, isso exige o ajuste adequado dos ponteiros e a liberação da memória correspondente, evitando vazamentos. Na fila circular, a remoção segue o comportamento típico de filas, retirando o elemento da posição de início e avançando o índice de forma circular.

As funções de atualização desempenham um papel essencial ao permitir alterações pontuais nas informações armazenadas, como ajustes de diâmetro e distância

de asteroides, modificação do status de missões ou eventuais correções em dados de astronautas. Essas operações são realizadas sem reconstruir as estruturas, preservando os elementos existentes e garantindo consistência.

Por fim, as funções de consulta permitem ao sistema localizar e exibir elementos de maneira clara e precisa. A busca por asteroides, minérios e missões percorre as listas encadeadas conforme necessário, enquanto a visualização da fila de astronautas respeita a organização circular da estrutura. Essas consultas fornecem ao usuário uma visão acessível dos dados armazenados, contribuindo para o gerenciamento eficiente das operações espaciais simuladas.

2.3.3 Funções Gerais

Embora também sejam responsáveis pela manipulação das TADs utilizadas no sistema, algumas funções se destacam por operarem em um nível superior às demais, uma vez que envolvem simultaneamente todas as estruturas de dados do projeto. Essas funções desempenham papel central na lógica do sistema, coordenando a interação entre asteroides, missões e astronautas de forma integrada.

Um dos principais exemplos é a função de lançamento de missão, que realiza consultas e operações conjuntas sobre a lista de asteroides, a fila circular de astronautas e a lista encadeada de missões. A partir dos dados fornecidos pelo usuário, essa função associa três astronautas à missão, registra o código do asteroide de destino e atualiza o status das entidades envolvidas — o asteroide passa a ser marcado como "*Ocupado"*, enquanto a missão tem seu status alterado para "*Lançada"*. Essa interação coordenada demonstra a importância dessa função no fluxo operacional do sistema.

Outra função de destaque é a de encerramento de missões, responsável por identificar a missão de maior prioridade que se encontra com status "*Lançada"*. Uma vez localizada, essa função devolve os astronautas atribuídos à missão para a fila circular, liberando-os para futuras operações. Em seguida, atualiza o status do asteroide associado para "*Livre"* e marca a missão como "*Encerrada"*, concluindo assim o ciclo operacional iniciado no lançamento.

3. Especificação das entradas e saídas

3.1 Entradas do Sistema

O arquivo utiliza arquivos específicos de entrada no formato .txt contendo dados estruturados no formato das funções de leitura de arquivo de cada uma das entidades. Abaixo são apresentados os formatos de cada arquivo.

3.1.1 Cadastro de Asteroides

O arquivo asteroides.txt contém os dados referentes aos asteroides cadastrados no sistema, seguindo o formato abaixo:

Código 1. Especificação da organização dos dados no arquivo asteroides.txt

```C/C++
CodigoAsteroide;Distancia;Diametro;Status;QtdMinerios
CodigoMinerio;NomeMinerio;QuantidadeMinerio
CodigoMinerio;NomeMinerio;QuantidadeMinerio
CodigoMinerio;NomeMinerio;QuantidadeMinerio
...
```

Os campos possuem a seguinte definição:

- **CodigoAsteroide:** identificador único do asteroide (inteiro).
- **Distancia:** distância do asteroide em relação à Terra (float).
- **Diametro:** diâmetro aproximado do asteroide em quilômetros (inteiro).
- **Status:** indica a situação atual do asteroide no sistema, podendo assumir valores como *Livre* ou *Ocupado* (string).
- **QtdMinerios:** número total de minérios associados ao asteroide (inteiro).
- **CodigoMinerio:** identificador único de cada minério encontrado no asteroide (inteiro).
- **NomeMinerio:** nome do minério (string, sem espaços).
- **QuantidadeMinerio:** quantidade disponível do minério (inteiro).

Cada asteroide pode possuir zero ou mais minérios associados. Assim, após a linha principal contendo os dados do asteroide, seguem tantas linhas adicionais quanto indicado no campo **QtdMinerios**, cada uma descrevendo um minério encontrado naquele corpo celeste. Esse formato permite que o sistema reconstrua corretamente a lista encadeada de minérios pertencente a cada asteroide, garantindo consistência durante a carga inicial dos dados.

3.1.2 Cadastro de Astronautas

O arquivo astronautas.txt contém os dados dos astronautas cadastrados no sistema, no seguinte formato:

Código 2. Especificação da organização dos dados no arquivo astronautas.txt

```C/C++
CodigoAstronauta;Nome;Especialidade
CodigoAstronauta;Nome;Especialidade
CodigoAstronauta;Nome;Especialidade
```

Os campos possuem a seguinte definição:

- **CodigoAstronauta:** identificador numérico único do astronauta (inteiro).
- **Nome:** nome do astronauta (string, sem espaços, com até 49 caracteres). No contexto do arquivo, o nome é gravado como uma única palavra, de forma compatível com a leitura realizada pela função de carga.
- **Especialidade:** área de atuação do astronauta (string, sem espaços, com até 29 caracteres), como por exemplo piloto, biologo, engenheiro, programador, entre outras.

3.1.3 Cadastro de Missões

O arquivo **missoes.txt** armazena todas as informações referentes às missões espaciais registradas no sistema. Cada missão é descrita inicialmente por uma linha contendo seus atributos principais, seguida por um bloco de linhas que representa a tripulação associada à missão. O formato do arquivo é o seguinte:

Código 3. Especificação da organização dos dados no arquivo missoes.txt

```
C/C++

CodigoMissao;NomeMissao;Status;Prioridade;DuracaoDias
CodigoAstronauta NomeAstronauta Especialidade
CodigoAstronauta NomeAstronauta Especialidade
CodigoAstronauta NomeAstronauta Especialidade
...
```

Os campos possuem a seguinte definição:

- **CodigoMissao:** identificador numérico único da missão (inteiro).
- **NomeMissao:** nome da missão (string sem espaços, devido ao formato de leitura adotado).
- **Status:** estado atual da missão, podendo assumir valores como *planejada*, *lancada* ou *encerrada*.
- **Prioridade:** valor inteiro que determina a ordem de execução da missão, utilizado para manter a lista de missões ordenada.
- **DuracaoDias:** duração prevista da missão, expressa em dias (inteiro).
- **CodigoAstronauta:** identificador único do astronauta (inteiro).
- **NomeAstronauta:** nome do astronauta (string sem espaços).
- **Especialidade:** área de atuação, como *piloto*, *biologo*, *engenheiro*, entre outras (string sem espaços).

3.2 Saídas do Sistema

Após a execução do sistema, os dados de todas as TADS utilizadas são salvos em arquivos com os nomes e formatos já citados anteriormente para leitura. Sendo eles astronautas.txt, asteroides.txt e missoes.txt.

4. Casos de Teste

Para validar o correto funcionamento do sistema de gerenciamento da Mineradora Espacial, foram elaborados três conjuntos de testes agrupados na pasta “TestesOperacionais”. Cada conjunto contém arquivos de entrada com diferentes configurações e situações possíveis no contexto de mineração de asteroides, permitindo analisar o comportamento do sistema em cenários variados.

A seguir, descrevem-se detalhadamente as três pastas de teste, o conteúdo de cada uma e os resultados esperados.

4.1 Teste 1 – Carga Inicial Simples

A primeira pasta contém três arquivos de entrada:

- **astronautas.txt**
  Contém *5 astronautas*, permitindo verificar o correto preenchimento da fila circular.
- **asteroides.txt**
  Contém *3 asteroides livres*, cada um com minérios variados.
- **missoes.txt**
  Contém *1 missão planejada* (sem tripulação designada ainda).

Processamento esperado:

O sistema deve carregar corretamente os três arquivos, reconstruindo:

- a fila circular com 5 astronautas
- a lista de asteroides com suas listas de minérios
- a lista de missões com ordenação de prioridade preservada

Nenhum lançamento é realizado neste teste.

Resultados esperados:

- As três listas são exibidas corretamente quando solicitadas.
- Nenhum erro de leitura deve ocorrer.
- O sistema deve estar pronto para lançamento de missões.

4.2 Teste 2 – Lançamento de Missão com Sucesso

Arquivos de entrada:

- **astronautas.txt**
  Contém *10 astronautas*.

- **asteroides.txt**
  Contém *2 esteróides livres*, um deles contendo minérios raros.
- **missoes.txt**
  Contém *1 missão planejada* com alta prioridade.

Processamento esperado:

O usuário solicita o **lançamento da missão**, que deve:

- selecionar os **3 primeiros astronautas da fila**
- alterar o status do asteroide escolhido para **Ocupado**
- alterar o status da missão para **Lançada**

Resultados esperados:

- A fila circular deve agora conter **7 astronautas restantes**.
- O asteroide alvo deve estar com status **Ocupado**.
- A missão deve constar como **Lançada**, com tripulantes preenchidos corretamente.
- Todas as atualizações devem aparecer no arquivo ao salvar.

4.3 Teste 3 – Encerramento de Missão

Arquivos de entrada:

- **astronautas.txt**
  Contém *5 astronautas*.
- **asteroides.txt**
  Contém *1 asteroide ocupado*.
- **missoes.txt**
  Contém *1 missão já lançada*, contendo 3 tripulantes.

Processamento esperado:

O usuário executa **encerrar missão**, o que deve:

- devolver os 3 astronautas para a fila circular
- mudar o status do asteroide para **Livre**
- registrar a missão como **Encerrada**

Resultados esperados:

- A fila circular deve conter novamente *todos os 5 astronautas*.
- O asteroide deve retornar ao status **Livre**.
- A missão deve constar como **Encerrada**.
- Ao salvar, o arquivo de missões deve registrar o novo status.

5. Avaliação e análise da solução proposta (incluindo eficiência)

Nesta seção são avaliadas as estruturas de dados utilizadas no sistema de gerenciamento da mineradora espacial, considerando sua eficiência e adequação às operações realizadas durante o ciclo de execução do programa.

5.1 Lista Encadeada – Asteroides

A lista encadeada de asteroides armazena todos os corpos celestes cadastrados no sistema. Essa estrutura foi escolhida pela sua flexibilidade: permite crescimento dinâmico e não impõe limite ao número de asteroides que podem ser inseridos, o que está alinhado ao contexto fictício do projeto. Como não há necessidade de ordenação, novos asteroides são sempre inseridos no início da lista, portanto a complexidade é de O(n).

Já em relação a busca e a remoção, a complexidade de pior caso também é O(n), tendo em vista que a função irá percorrer a lista inteira.

5.2 Lista Encadeada Interna – Minérios por Asteroide

Cada asteroide possui uma lista encadeada de minérios associados, permitindo que quantidades variadas de recursos sejam representadas de maneira dinâmica. A ausência de limitação fixa para o número de minérios reforça a adequação dessa estrutura ao contexto espacial simulado. Assim como na lista principal, os minérios são inseridos no início da lista, o que garante complexidade O(1) para inserção. Já as operações de busca de um minério específico ou exclusão de um deles apresentam complexidade O(n), uma vez que o algoritmo precisa percorrer a lista até encontrar o nó correspondente.

5.3 Lista Encadeada Ordenada – Missões

A lista encadeada de missões contém todas as missões espaciais cadastradas no sistema e mantém seus elementos ordenados segundo o campo de prioridade. Essa escolha garante que a missão mais prioritária esteja sempre acessível de forma lógica e estruturada. No entanto, essa ordenação implica que a inserção de novas missões exija a busca pela posição correta na lista, resultando em uma complexidade de O(n). Da mesma forma, operações de busca e remoção também possuem complexidade O(n), já que o sistema precisa percorrer sequencialmente os nós até encontrar o elemento desejado.

5.4 Fila Circular – Astronautas

A fila circular de astronautas representa os recursos humanos disponíveis para lançamento. Implementada sobre um vetor de tamanho fixo, ela garante que as operações de entrada e saída sejam realizadas sem deslocamento de elementos, preservando o uso eficiente das posições do vetor por meio de aritmética modular. Tanto a inserção quanto a remoção de astronautas possuem complexidade O(1), já que consistem apenas na atualização dos índices que controlam o início e o fim da fila. As verificações de fila cheia ou vazia também são O(1), uma vez que dependem exclusivamente da comparação entre os índices.

6. Autoavaliação do processo de desenvolvimento

Estou extremamente satisfeito com o resultado deste trabalho, pois considero que ele possibilitou a aplicação prática de diversos conceitos estudados ao longo da disciplina de Algoritmos e Programação II. A implementação das estruturas de dados, o uso de ponteiros, a manipulação de arquivos e a organização modular do código contribuíram significativamente para ampliar minha compreensão sobre a lógica de construção de sistemas, bem como sobre o desenvolvimento de soluções estruturadas e eficientes em linguagem C.

Além do aprendizado técnico, sinto-me particularmente realizado com meu desempenho pessoal. Mesmo diante das limitações de tempo e das múltiplas demandas enfrentadas ao longo do semestre, consegui me organizar, revisar os conteúdos necessários e desenvolver o projeto integralmente por conta própria. Essa experiência reforçou minha autonomia, disciplina e capacidade de solucionar problemas, consolidando não apenas conhecimentos teóricos, mas também competências fundamentais para minha formação acadêmica.