Introdução

Este trabalho prático, desenvolvido no âmbito da cadeira de Laboratórios de Informática III, tem como objetivo principal a construção de um programa estruturado (em vários módulos) em C para análise de dados de um sistema de streaming de música. O projeto tem foco na consolidação de algumas habilidades de programação em C - sobretudo nas boas práticas de modularização, mas também no encapsulamento e gestão da memória. O sistema, ao explorar uma grande quantidade de dados, como informações sobre músicas, artistas e users, proporciona uma experiência prática na manipulação e consulta de grandes volumes de dados.

A implementação do projeto está dividida em duas fases, sendo este relatório sobre a primeira. Esta fase foca no desenvolvimento e validação de funcionalidades básicas, incluindo o processamento e validação de dados a partir de arquivos CSV e a implementação de um conjunto de 3 queries. Cada query permite ao sistema responder a perguntas específicas sobre os dados processados, como a listagem de informações de users, identificação dos artistas com maior discografia e popularidade de géneros musicais em diferentes faixas etárias. Todas estas funcionalidades foram projetadas de forma modular .

A organização do projeto baseou-se em princípios de modularidade. Dividimos o sistema em módulos para leitura/tratamento de dados, queries e operações sobre estruturas de dados. Esta abordagem permite encapsular as operações de leitura e manipulação de dados em um único ficheiro, facilitando a reutilização de código e simplificando o processo de desenvolvimento e manutenção. Além disso, a estrutura modular possibilita que o programa seja executado em dois modos distintos (nesta fase): o modo principal, para armazenamento de dados e execução geral de queries e o modo de testes, que inclui validações automatizadas de cada query.

Para garantir a qualidade e a estabilidade do programa, demos especial atenção à gestão de memória e à prevenção de leaks, utilizando o gdb e o valgrind para nos ajudar a perceber onde poderiam estar os problemas relacionados à memória. Também demos atenção à validação dos dados: garantimos que os dados estavam de acordo com os formatos definidos (datas, e-mails, listas e tipos de subscrição). Dados inválidos foram colocados em arquivos de erro, facilitando a perceção de entradas incorretas.

A realização desta fase do projeto foi uma grande oportunidade para melhorar alguns aspetos da programação em C, especialmente a organização modular mas também a boa gestão de memória. Os desafios enfrentados ao longo do desenvolvimento desta fase, sobretudo aqueles relacionados à eficiência das estruturas de dados, à qualidade do código e à eliminação dos memory leaks, certamente contribuíram para o nosso desenvolvimento como programadores.

Sistema

A arquitetura do sistema foi projetada com o objetivo de garantir modularidade, separando as tarefas em módulos diferentes que encapsulam cada etapa do processamento e consulta de dados. Este design modular não só facilita a manutenção e o aperfeiçoamento do código, mas também permite que futuras extensões e otimizações sejam incorporadas mais facilmente - de forma organizada e isolada.

1. Arquitetura Geral

O sistema é composto por um programa principal que realiza o processamento de dados e execução de queries, e um programa testes que automatiza a verificação da correção dos outputs. Abaixo estão descritos os módulos implementados e as suas respetivas responsabilidades, seguidos de um diagrama que mostra esta organização.

Programa Principal

1. **Inputs**: O sistema recebe dois arquivos principais como entrada: o Dataset, contendo os dados a serem processados (users, músicas, artistas), e o input.txt, que especifica as queries a serem executadas. Estes inputs são o ponto de partida para o fluxo de dados no sistema.
2. **Validação**: A primeira etapa do processo envolve o módulo de validação, que verifica a integridade dos dados do dataset. Dados incorretos ou inválidos são identificados e exportados para um arquivo errors.csv.
3. **Gestor de Dados**: Após a validação, os dados são organizados em três gestores específicos: **Gestor Artists**, **Gestor Musics** e **Gestor Users**. Cada um destes gestores encapsula as operações e dados relevantes ao seu domínio, mantendo as responsabilidades bem definidas. Para centralizar as interações entre os gestores é utilizado um **Gestor de Gestores.**
4. **Interpretação e Execução de Queries**: Depois de identificadas as queries a serem executadas é chamado o **Gestor Queries**. Este módulo coordena a execução das três queries principais do sistema, cada uma focada em responder perguntas específicas sobre o conjunto de dados:
    - **Query 1**: Listagem de informações específicas de users.
    - **Query 2**: Identificação dos artistas com maior discografia.
    - **Query 3**: Análise da popularidade de géneros musicais em diferentes faixas etárias.

    As respostas das queries são armazenadas nos ficheiros output1.txt, output2.txt, etc., dependendo do que foi especificado no input.txt.

Programa Testes

O programa de testes foi desenvolvido para automatizar a verificação da correção do sistema. Ele compara os outputs gerados (output1.txt, output2.txt, ...) com os resultados esperados fornecidos, garantindo que o sistema responde corretamente às queries. Este processo de comparação gera um resultado final, que indica o sucesso ou falha dos testes.

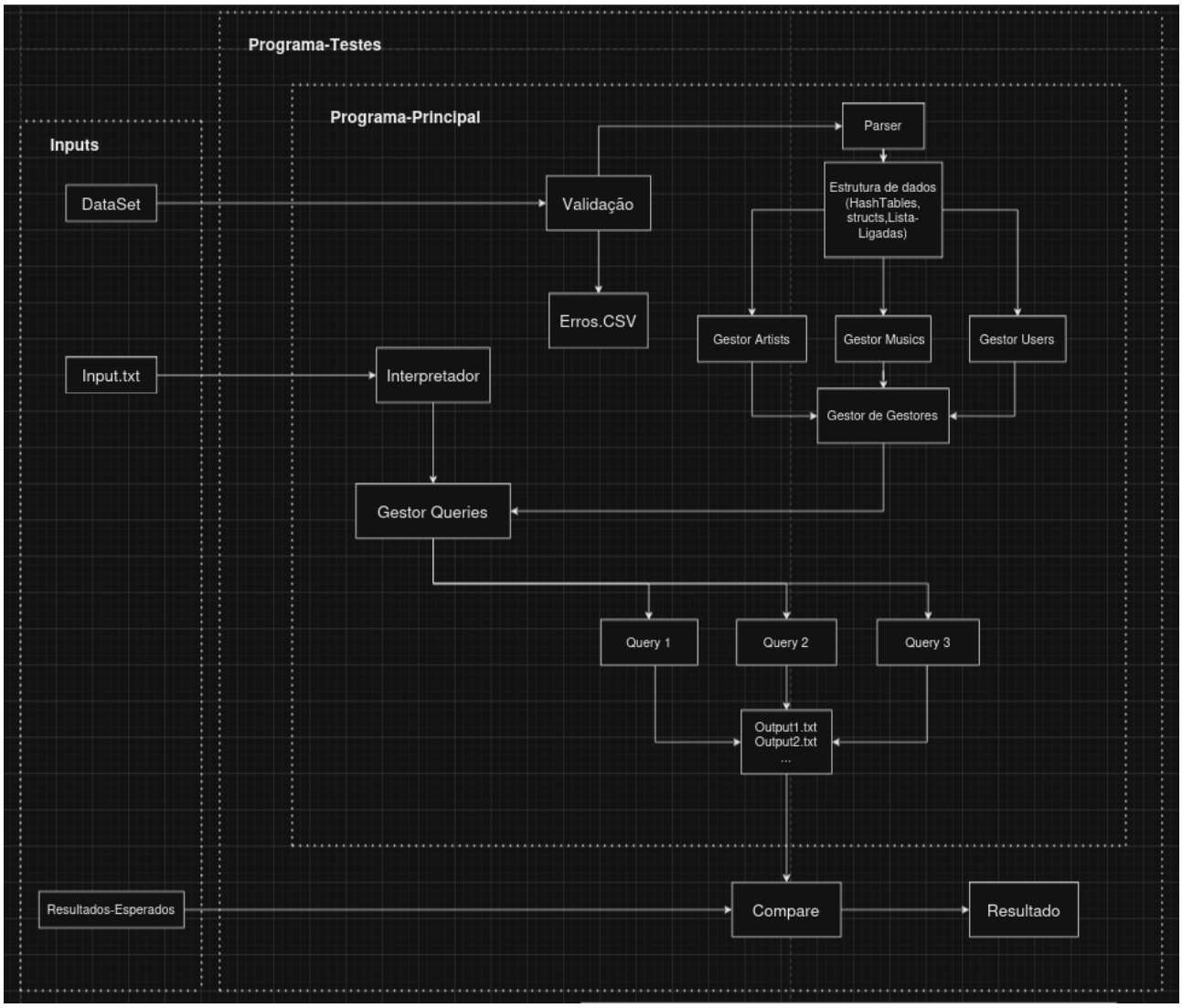
3. Justificativas das Escolhas de Modularidade

A separação do sistema em módulos foi essencial para a clareza e organização do código. Abaixo, destacamos as justificativas para a escolha desta arquitetura modular:

1. **Manutenção**: Cada módulo foi projetado para ser relativamente independente, o que facilita a identificação e a correção de problemas. Por exemplo, se houver um problema num dado módulo, ele pode ser corrigido diretamente no mesmo sem afetar outros módulos. Além disso, a presença do **Gestor de Gestores** centraliza o controlo sobre os dados, havendo uma maior eficiência no acesso e manipulação das informações.
2. **Escalabilidade**: O sistema foi projetado de maneira que novos tipos de queries e funcionalidades possam ser adicionados em módulos específicos sem causar interferências nas outras partes do sistema.
3. **Gestão de Memória**: A separação dos módulos também facilita a gestão de memória, uma vez que a alocação e a libertação podem ser controladas de maneira precisa dentro de cada módulo. Isto contribui para evitar leaks de memória e simplifica a análise e a resolução de problemas com ferramentas como o valgrind.

4. Diagrama da Arquitetura

Abaixo está o diagrama que representa a organização dos módulos:



Discussão

A Fase 1 deste projeto envolveu a construção de um programa modular para a análise de dados de um sistema de streaming de música. O desenvolvimento focou principalmente na implementação de um sistema robusto para a validação e processamento de dados, bem como na execução de queries específicas em um tempo relativamente pequeno e não gastando muita memória. Esta secção analisa criticamente as decisões de design e implementação, discute o desempenho do sistema com base nos testes realizados e aborda os desafios enfrentados ao longo do processo.

Modularização e Organização do Sistema

Optamos por uma arquitetura modular para garantir a separação de responsabilidades e facilitar a manutenção e expansão futura do sistema. Cada módulo foi projetado para desempenhar uma função bem definida:

- O **Programa Principal** controla a execução geral, gerindo a ordem de operações e a integração entre módulos.
- O **Módulo da Validação** cuida da identificação dos erros, exportando os dados inválidos para o arquivo errors.csv. Isto garante que somente dados limpos e consistentes avancem para o restante do sistema.
- O **Módulo de Gestores** armazena as informações de maneira eficiente e organizada, dividindo os dados entre os gestores de artistas, músicas e users. A presença do **Gestor de Gestores** centraliza o acesso e facilita a comunicação entre os diversos componentes. Foram usadas hashtables (estrutura que consideramos mais eficiente) para guardar todos os dados mas depois também listas ligadas para as queries 2 e 3.
- O **Módulo de Queries** executa tarefas específicas (Queries 1, 2 e 3), respondendo a perguntas com base nos dados armazenados. Este módulo é gerenciado pelo **Gestor Queries**, que controla as operações de acordo com a especificação das consultas no input.txt.
- O **Módulo de Testes** automatiza a verificação dos resultados das queries, comparando os outputs com os resultados esperados e reportando o sucesso ou falha de cada teste.

Esta estrutura modular mostrou-se vantajosa, pois permitiu que cada módulo fosse implementado e testado de forma independente. A modularidade também facilita a expansão do sistema para incluir novas funcionalidades ou queries na próxima fase. A organização modular proporcionou clareza e eficiência no desenvolvimento do projeto, permitindo que cada integrante do grupo se concentrasse em um módulo específico.

Análise de Desempenho

Para avaliar o desempenho do sistema, realizamos testes nas três queries nas nossas três maquinas com hardwares variados. O ambiente de testes incluiu:

- **Hardware utilizado**: Máquina 1: Asus M515UA 16 GB RAM proc. AMD Ryzen 7; Máquina 2: Asus Vivobook 16 GB Ram proc. AMD Ryzen 7; Máquina 3: Asus M515DA 8 GB RAM proc. AMD Ryzen 5
- **Número de repetições**: Cada query foi executada 5 vezes para garantir a consistência dos resultados e medir o tempo médio de execução.
- **Programa-Testes**: Utilizamos o programa-testes para verificar a precisão e a eficiência de cada query, comparando os outputs gerados com os resultados esperados.

Os testes indicam que o desempenho do sistema é bom para o volume de dados utilizado nesta fase. A estrutura modular e as técnicas de validação de dados evitaram sobrecarga desnecessária, e o uso eficiente das estruturas de dados garantiu uma boa performance para as consultas realizadas. Os tempos médios de execução de cada query estão documentados na tabela seguinte:

	Média Máquina 1 (s)	**Média Máquina 2 (s)**	**Média Máquina 3 (s)**
Query 1	0,000369	0,000574	0,001015
Query 2	0,905353	0,979764	1,206447
Query 3	2,604783	2,532566	3,302406
Total	7,047437	6,157356	7,669117

Máquina 1 - Pedro

Máquina 2 - Francisco

Máquina 3 - Simão

Resultados dos Testes e Justificativa dos Desempenhos

Os testes evidenciaram que o desempenho e a precisão das respostas variam conforme a complexidade da query e a carga de dados. Por exemplo:

- **Query 1**: Listagem de informações dos usuários; apresentou o desempenho mais rápido, pois exige apenas acesso direto aos dados previamente organizados na hashtable.
- **Query 2**: A identificação dos artistas com maior discografia exigiu operações de busca um pouco mais complexas, resultando num tempo de execução ligeiramente superior ao da query 1. Este comportamento era esperado, dado que o algoritmo precisa percorrer e comparar dados de forma mais intensiva.
- **Query 3**: Popularidade dos géneros de música por faixa etária. Esta é a query mais complexa (é necessários utilizar dados referentes a duas entidades – users e músicas) pelo que foi a que demorou mais tempo a executar. No entanto, consideramos que esta query apresenta um bom desempenho. A estrutura de dados utilizada permitiu uma análise eficaz dos géneros, distribuindo os dados numa lista ligada para permitir acesso repartido por cada faixa etária (funciona como um histograma).

Os resultados foram consistentes em diferentes hardwares, indicando que a implementação é otimizada para o volume de dados.

Desafios e Lições Aprendidas

Um dos principais desafios desta fase foi a garantia de uma gestão eficaz da memória, especialmente ao lidar com grandes volumes de dados. A utilização do gdb e do valgrind foi essencial para identificar e corrigir leaks de memória, assegurando que o sistema mantivesse a estabilidade durante as execuções. Além disso, a necessidade da validação e tratamento de dados antes de os guardar revelou-se fundamental para a precisão das queries, sendo que dados inválidos poderiam comprometer os resultados.

Para gastar menos memória não guardamos a letra das músicas (musicas) e a descrição e o dinheiro auferido por cada vez que uma das música do artista é reproduzida (artistas) nas suas respetivas hashtables uma vez que esta informação não tinha utilidade.

A implementação modular permitiu um desenvolvimento colaborativo e mais eficiente, além de facilitar a identificação e resolução de problemas em cada componente específico.

Em resumo, o sistema desenvolvido para a Fase 1 alcançou os objetivos propostos, apresentando uma estrutura sólida e um bom desempenho. O foco nas técnicas de validação de dados e na modularidade do sistema proporciona uma base robusta para a próxima fase, onde pretendemos implementar melhorias no desempenho.

Esta secção discute as escolhas feitas e apresenta um panorama do desempenho do sistema, além de abordar os desafios e lições aprendidas.

Conclusão

Nesta primeira fase do projeto desenvolvemos um programa modular em C para análise de dados de um sistema de streaming de música, aplicando técnicas fundamentais de programação e manipulação de dados. Através deste trabalho, consolidamos práticas de modularização e validação de dados, que foram essenciais para a implementação de um sistema capaz de processar e consultar grandes volumes de informações.

A principal lição aprendida foi a importância de uma organização modular e da separação de tarefas em diferentes componentes do sistema. Esta abordagem tornou o desenvolvimento mais eficiente e facilitou a manutenção e testes de cada módulo de forma independente, além de facilitar futuras extensões.

A experiência/utilização de ferramentas como o gdb e o valgrind para a gestão de memória foi também muito importante, pois permitiu/facilitou a identificação e resolução de leaks de memória. Este processo ajudou-nos a consolidar os conceitos de alocação e libertação de memória, algo essencial em C, especialmente ao lidar com grandes volumes de dados.

Achamos que o nosso sistema tenha apresentado um bom desempenho nesta Fase 1.

Em resumo, esta fase proporcionou-nos uma experiência prática na programação modular em C e também na manipulação de dados de grande dimensão. Depois de todos os desafios que enfrentamos na realização desta fase do projeto acreditamos que crescemos como programadores!