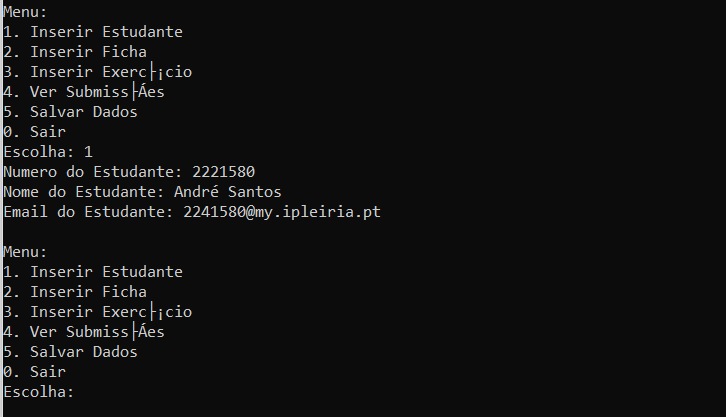
|  |  |  |  |
| --- | --- | --- | --- |
|  | | Curso de TeSP em  Programação de Sistemas de Informação  Fundamentos de Programação (FP) | |
|  |  | |  |
| **Ano Letivo 2024/2025** |  | | **1º Ano, 1º Semestre** |
|  | | | |
| **Projeto de FP** | | | |



**Relatório de acompanhamento do**

**Projeto de Fundamentos de Programação**

|  |  |
| --- | --- |
|  |  |
| **Grupo:** PL1 | **Docente: José Vítor** |
| **Nº 2241580** | André Santos |
| **Nº 2241582** | Rodrigo da Costa |
|  |  |

ÍNDICE

[1 Introdução 3](#_Toc187181947)

[2 Constantes e variáveis principais 4](#_Toc187181948)

[3 Funcionalidades implementadas e respetivas funções 5](#_Toc187181949)

[4 Validação de dados de entrada 9](#_Toc187181950)

[5 Realização de testes e descrição dos respetivos cenários 11](#_Toc187181951)

[6 Conclusão 12](#_Toc187181952)

# Introdução

Este projeto foi realizado no contexto da unidade curricular de Fundamentos de Programação, proposto pelo Professor José Vítor. A tarefa consiste na criação de um protótipo de aplicação, desenvolvido em linguagem C, que permita a gestão de informações relacionadas às submissões de exercícios das fichas das aulas práticas.

A aplicação deverá incluir funcionalidades como:

1. **Registo e consulta de dados**:
   * Dados dos estudantes;
   * Dados das fichas de exercícios;
   * Dados dos exercícios;
   * Dados das submissões.
2. **Estatísticas**:
   * Total de submissões realizadas por estudante;
   * Média das classificações nas submissões de cada estudante;
   * Percentagem de exercícios resolvidos em cada ficha por estudante.
3. **Gestão de dados**:
   * Armazenamento e leitura de dados da aplicação em ficheiros binários.

# Constantes e variáveis principais

No desenvolvimento do sistema, criamos constantes para limitar o uso de recursos. A constante TAMANHO\_STRING\_LONGA (80) define o tamanho máximo para strings extensas, como nomes de estudantes e fichas, enquanto TAMANHO\_STRING\_CURTA (20) é usada para informações mais breves.

Também definimos limites como MAXIMO\_EXERCICIOS (100), MAXIMO\_ESTUDANTES (100), MAXIMO\_EXERCICIOS\_POR\_FICHAS (10), MAXIMO\_FICHAS (10) e MAXIMO\_SUBMISSOES (100), que controlam a quantidade máxima de registos no sistema. Esta definição tem como objetivo simplificar alterações futuras, otimizar o uso de memória e manter a consistência ao longo do programa.

Estas foram as variáveis que implementamos na função main:

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

# Funcionalidades implementadas e respetivas funções

Para fazer a devida implementação do nosso projeto tivemos de criar várias funções que desenvolviam várias funcionalidades sendo elas:

**Registar e consultar dados dos estudantes**

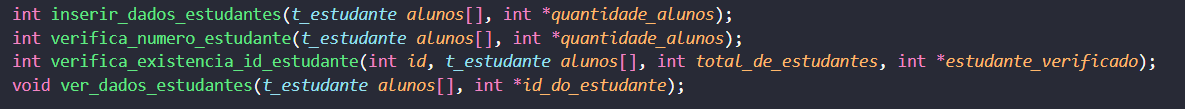
Para Implementar esta funcionalidade precisamos de criar duas funções principais, uma para o utilizador inserir os dados dos estudantes “Inserir\_dados\_estudantes” e outra para o utilizador conseguir ler os dados que foram inseridos “ver\_dados\_estudantes”, além disso criamos outras duas funções para fazer validações, “verifica\_numero\_estudante” que tem como objetivo verificar se aquele numero de estudante existe e a função “verifica\_existencia\_id\_estudante” que tem como objetivo verificar se aquele id de estudante já existe (Fig.1).

Fig.1 – Funções desenvolvidas

**Registar e consultar dados das fichas de exercícios**

Esta funcionalidade é idêntica precisamos de criar duas funções principais, uma para o utilizador inserir os dados das fichas “Inserir\_dados\_fichas” e outra para o utilizador conseguir ler os dados que foram inseridos “ver\_dados\_fichas”, além disso criamos outras três funções para fazer validações, “insira\_verifica\_nome\_ficha” que tem como objetivo verificar se aquele nome da ficha é único, a função “insira\_verifica\_exercicios\_da\_ficha” que tem como objetivo garantir que o numero de exercícios por ficha esteja entre o limite aceite e a ultima “insira\_verifica\_data” que tem como objetivo verificar se a data é valida (Fig.2).

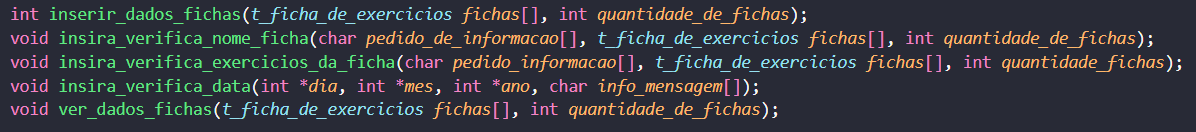
****

Fig.2 – Funções desenvolvidas

**Registar e consultar dados dos exercícios**

A funcionalidade de registar e consultar dados dos exercícios foi implementada com várias funções que garantem a integridade dos dados inseridos. A função “inserir\_dados\_exercicios” permite ao utilizador adicionar novos exercícios a uma ficha de exercícios, validando a quantidade de exercícios e a ligação correta com a ficha associada. Para garantir a consistência dos dados, a função “insira\_verifica\_id\_ficha” valida se o ID da ficha associada ao exercício existe, garantindo que o exercício está corretamente vinculado a uma ficha válida. A função “ler\_dificuldade\_exercicio” solicita e armazena a dificuldade do exercício, enquanto a função “ler\_tipo\_solucao\_exercicio” permite ao utilizador escolher o tipo de solução para o exercício. A função “verifica\_existencia\_id\_exercicios” assegura que o ID do exercício inserido ainda não foi utilizado, evitando duplicações. Finalmente, a função “ver\_dados\_exercicios” permite consultar as informações de um exercício específico, exibindo os detalhes conforme o ID fornecido.

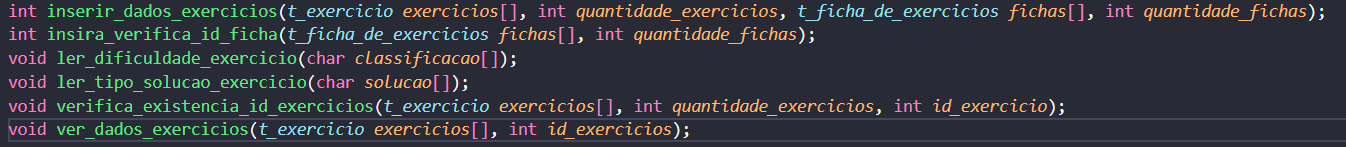


Fig.3 – Funções desenvolvidas

**Registar e consultar dados das submissões**

A funcionalidade de gestão de submissões foi implementada com várias funções para garantir o registo e consulta adequados das mesmas. A função “inserir\_submissao” permite que um estudante faça a submissão de um exercício, associando a ficha e o exercício correspondentes, além de validar a quantidade de alunos e exercícios existentes no sistema. A função “exibir\_submissoes” exibe todas as submissões registadas, proporcionando uma visão geral das atividades realizadas pelos estudantes. A variável “quantidade\_submissoes” mantém o controle do número de submissões realizadas, e o array “submissoes” armazena as submissões até o limite definido por “MAXIMO\_SUBMISSOES”, garantindo que o sistema não ultrapasse o número máximo de entradas.

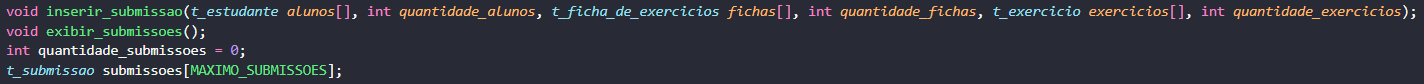


Fig.4 – Funções desenvolvidas

**Registar e consultar os dados das submissões**

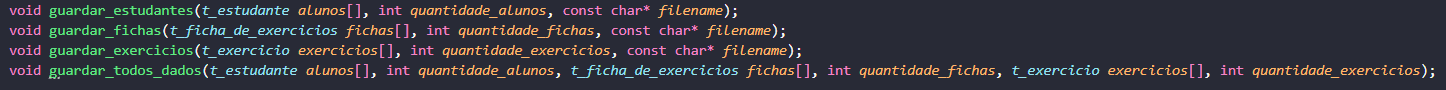
As funções de guardar dados foram implementadas para permitir a persistência de informações no sistema. A função “guardar\_estudantes” salva os dados dos estudantes em um arquivo especificado pelo nome do arquivo passado como argumento, garantindo que as informações dos alunos sejam armazenadas de forma segura. A função “guardar\_fichas” realiza a mesma operação, mas para os dados das fichas de exercícios, permitindo a recuperação das fichas sempre que necessário. Da mesma forma, “guardar\_exercicios” armazena os dados dos exercícios em um arquivo, garantindo que as informações relacionadas aos exercícios sejam preservadas. Por fim, a função “guardar\_todos\_dados” serve para salvar todos os dados (estudantes, fichas e exercícios) em seus respetivos arquivos, permitindo a recuperação completa das informações do sistema de forma centralizada e eficiente.

Fig.5 – Funções desenvolvidas

**Guardar Ficheiros Binários**

As funções de guardar dados são responsáveis por permitir a persistência das informações dentro do sistema. A função guardar\_estudantes tem o objetivo de salvar os dados dos estudantes em um arquivo, cujo nome é passado como parâmetro. Dessa forma, ela assegura que as informações dos alunos sejam armazenadas de maneira confiável para futura recuperação. A função guardar\_fichas, por sua vez, realiza um processo similar, mas com os dados das fichas de exercícios. Ela possibilita que as fichas sejam armazenadas em arquivos específicos e recuperadas quando necessário. Da mesma forma, a função guardar\_exercicios garante que os dados dos exercícios sejam armazenados em um arquivo, preservando a integridade dessas informações ao longo do tempo. Para facilitar o gerenciamento dos dados, a função guardar\_todos\_dados tem a tarefa de salvar todas as informações (estudantes, fichas e exercícios) nos arquivos apropriados. Com isso, o sistema assegura uma maneira eficiente e centralizada de gerenciar os dados, possibilitando tanto a persistência quanto a recuperação de informações.

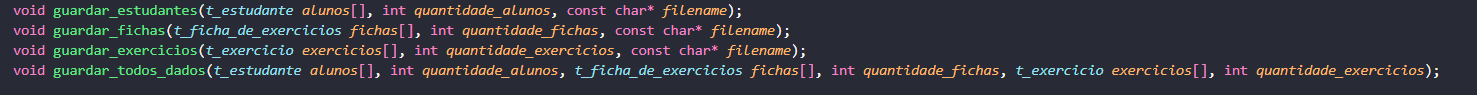
****

Fig.6 – Funções desenvolvidas

**Ler Ficheiros Binários**

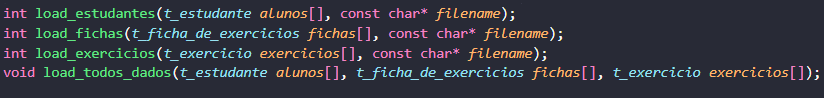
As funções de carregamento de dados têm como objetivo restaurar as informações salvas anteriormente a partir dos arquivos. A função load\_estudantes é responsável por carregar os dados dos estudantes a partir de um arquivo específico, utilizando o nome do arquivo passado como argumento. Isso permite que as informações dos alunos sejam recuperadas e utilizadas no sistema. A função load\_fichas realiza uma tarefa semelhante, mas para as fichas de exercícios. Ela recupera os dados das fichas salvas em arquivos, possibilitando a reentrada dessas informações no sistema quando necessário. De forma análoga, a função load\_exercicios recupera os dados dos exercícios a partir de um arquivo, assegurando que todas as informações relacionadas aos exercícios sejam restauradas corretamente. A função load\_todos\_dados, por sua vez, carrega todos os dados do sistema (estudantes, fichas e exercícios) de uma vez, de maneira centralizada, garantindo que todas as informações do sistema sejam recuperadas de forma eficiente e consistente.

Fig.7 – Funções desenvolvidas

# Validação de dados de entrada

Para fazer as validações de dados de entrada criamos várias funções, neste tópico irei abordar todas as validações de dados que criamos durante o projeto.

**Verificação de Número do Estudante:**

O código verifica se o número do estudante já existe na lista de estudantes. Se existir, uma mensagem é exibida.

**Verificação de Existência de ID do Estudante:**

A função verifica\_existencia\_id\_estudante verifica se um ID único de estudante existe na lista de estudantes.

**Verificação de Nome da Ficha:**

A função insira\_verifica\_nome\_ficha verifica se o nome da ficha já existe antes de permitir a inserção de uma nova ficha.

**Verificação do Número de Exercícios:**

A função insira\_verifica\_exercicios\_da\_ficha garante que o número de exercícios inserido não exceda o limite máximo definido (MAXIMO\_EXERCICIOS\_POR\_FICHAS).

**Validação de Data:**

A função insira\_verifica\_data valida a data inserida, garantindo que o mês e o dia sejam válidos, e que o ano seja positivo. Também considera anos bissextos.

**Verificação de ID da Ficha:**

A função insira\_verifica\_id\_ficha verifica se o ID da ficha inserido é válido e se já existe.

Verificação de Existência de ID de Exercício:

A função verifica\_existencia\_id\_exercicios verifica se um ID de exercício já existe na lista de exercícios.

**Leitura de Dificuldade e Tipo de Solução:**

As funções ler\_dificuldade\_exercicio e ler\_tipo\_solucao\_exercicio garantem que o usuário insira uma opção válida (1 ou 2) para a dificuldade e tipo de solução.

**Verificação de Existência de Estudante:**

Ao inserir uma nova submissão, o código verifica se o estudante existe antes de prosseguir.

**Verificação de Existência de Ficha e Exercício:**

O código verifica se a ficha e o exercício inseridos existem antes de registar a submissão.

**Validação de Data de Submissão:**

A data de submissão é inserida em um formato específico (dd/mm/aaaa), mas não há validação explícita para garantir que a data seja válida no momento da inserção.

**Verificação de Abertura de Arquivo:**

Antes de escrever em um arquivo, o código verifica se o arquivo foi aberto corretamente. Se não, uma mensagem de erro é exibida.

# Realização de testes e descrição dos respetivos cenários

# Conclusão

Durante o desenvolvimento deste projeto, diversas coisas correram bem e funcionaram de maneira eficiente, permitindo a criação de um sistema funcional. A estrutura de dados foi bem definida, o que nos facilitou a implementação das funções . A implementação das funções de correu bem e isso garantiu que as informações fossem corretamente salvas e carregadas quando fosse necessário. A validação de dados, como a verificação dos números de exercícios e a garantia da unicidade de fichas e estudantes, foi implementada com eficácia.

Acreditamos que algumas coisas poderiam ter sido melhoradas, como a otimização do código, achamos que talvez tenhamos desenvolvido mais código do que o necessário para certas funcionalidades, o que deixa o programa mais pesado.

O fator tempo, foi tambem uma surpresa no nosso trabalho, parecia que tudo estava bem até começarem a aparecer erros cujo demoraram bastante tempo para serem resolvidos, e com isto perdemos tempo que poderíamos ter usado para implementar novas funcionalidades e otimizar o programa.

Tivemos tambem um grande problema no uso dos ponteiros, algumas funções pediam e outras não o que complicou bastante a realização do projeto.

Para futuros projetos, temos de criar novas práticas como a prática de ir sempre testando o código a cada mudança, acho que foi a coisa que mais nos prejudicou durante o trabalho.

Devido aos erros que surgiram o novo projeto ficou com alguns erros e não conseguimos implementar o menu de estatísticas, apesar de termos feito as funções