Trabalho Prático G14

1.0

Gerado por Doxygen 1.9.8

Capítulo 1

Trabalho Prático G14

1.1 Trabalho Prático

1.1.1 Descrição

Licenciatura em Engenharia de Sistemas Informáticos 2023-24

Laboratórios de Informática e Programação Imperativa

O programa visa apoiar o bem-estar e cuidados nutricionais, abordando o problema do comportamento alimentar saudável. Funcionalidades incluem carregamento de dados de pacientes, dietas e planos nutricionais, apresentação de estatísticas de calorias, listagem de pacientes com comportamento fora do plano nutricional e análises diversas. Desenvolvido com base em métodos rigorosos de análise de problemas e as melhores práticas de programação imperativa em C.

1.1.2 Índice

1.1.3 Grupo

O grupo deste trabalho é o grupo *nº14*.

Número	Nome
28602	Enrique George Rodrigues
27965	Andre

1.1.4 Hierarquia de Ficheiros

- **assets**/
  - Pasta para imagens.
- **build**/
  - Pasta criada quando compila o programa.
- **doc**/
  - Documentação do código.
    - **v0.1**/
      - **html**/
        - Site para documentação da versão 0.1.
      - **latex**/
        - Ficheiros Latex e documentação em PDF da versão 0.1.
    - **v1.0**/
      - **Biblioteca**/
        - **html**/
          - Site para documentação da biblioteca da versão 1.0.
        - **latex**/
          - Ficheiros Latex e documentação em PDF da biblioteca da versão 1.0.
      - **Programa**/
        - **html**/
          - Site para documentação do programa da versão 1.0.
        - **latex**/
          - Ficheiros Latex e documentação em PDF do programa da versão 1.0.
- **examples**/
  - Pasta que contém ficheiros de dados de exemplo para testar o programa.
- **src**/
  - Código-fonte.
    - **GestorDadosIO**/
      - Biblioteca para importar e exportar dados.
    - **LESI_PI_TP_a28602**/
      - Código-fonte principal do programa.
- **README.md**
  - Ficheiro README do projeto.
- **Makefile**
  - Ficheiro para construir e compilar o projeto para Linux (Windows é compilado através do VS IDE).
- ∗∗.gitignore∗∗
  - Ficheiro de configuração Git para especificar os ficheiros e pastas para ignorar.
- ∗∗.gitattributes∗∗
  - Ficheiro de configuração Git para especificar atributos para caminhos.

1.1.5 Compilação

1.1.5.1 Usando Visual Studio IDE no Windows

1. Abra o projeto `src/GestorDadosIO.sln` no Visual Studio IDE.
2. Compile o projeto para gerar a biblioteca.
3. Os ficheiros compilados serão colocados em `build/lib`.
4. Abra o projeto `src/LESI_PI_TP_a28602.sln` no Visual Studio IDE.
5. Compile o projeto para gerar o programa.
6. Os ficheiros compilados, incluindo a DLL, serão colocados em `build/prog`.

1.1.5.2 Usando MakeFile no Linux

Certifique-se de ter o `make` instalado no seu sistema e execute o seguinte comando no terminal:
```
make
```
Isto irá automaticamente compilar a biblioteca e o programa e depois colocará os ficheiros resultantes nos diretórios `build/lib` e `build/prog`, respectivamente.
Se deseja limpar todos os ficheiros compilados, pode executar o seguinte comando:
```
make clean
```
Isto apagará todo o conteúdo da pasta `build`.

1.1.6 Como Executar o Programa Bem Estar

O programa Bem Estar oferece diversas opções para importar dados e executar operações. Abaixo estão as opções disponíveis:

1. **Importar Dados com Ficheiros:**
   - `-p <ficheiro.x>`: Importa dados de pacientes de um ficheiro de texto ou binário.
   - `-d <ficheiro.x>`: Importa dados de dietas de um ficheiro de texto ou binário.
   - `-l <ficheiro.x>`: Importa dados de planos de um ficheiro de texto ou binário.
   - `-tab`: Indica que o ficheiro está formatado com separação por tabulações.
   - `-bin`: Indica que o ficheiro é do tipo binário.
2. **Apagar Dados Guardados:**
   - `-apagarbd`: Apaga a base de dados (ficheiros binários).
3. **Ver Ajuda Através do Programa:**
   - `-ajuda`: Mostra como utilizar o programa.
4. **Ver Possíveis Erros do Programa:**
   - `-ajuda erros`: Mostra os possíveis erros do programa.

1.1.6.1 Exemplos de Utilização:

Dentro da pasta "examples" foram adicionados ficheiros de formato de texto, tab e binário. Estes ficheiros podem ser utilizados para testar o programa com dados.

- Importar dados de pacientes de um ficheiro de texto normal:
  ```
  ./bemestar -p ex1.txt
  ```
- Importar dados de dietas com separação por tabulações:
  ```
  ./bemestar -d ex2.txt -tab
  ```
- Importar dados de planos no formato binário:
  ```
  ./bemestar -l ex3.bin -bin
  ```
- Apagar a base de dados:
  ```
  ./bemestar -apagarbd
  ```

Lembre-se de ajustar os nomes dos ficheiros e as opções conforme necessário para a sua utilização específica.

1.1.7 Estado Atual dos Testes

Os testes de funcionalidades atualmente estão num estado de **SUCESSO** (ou **FALHA**). Abaixo, são apresentados os resultados dos testes realizados:

1.1.7.1 Teste de Compilação

Ambiente	Status
Windows	SUCESSO
Linux	SUCESSO

1.1.7.2 Teste de Importação de Dados

Importação de Pacientes

Tipo	Ambiente	Status
Texto	Windows	SUCESSO
Binário	Windows	SUCESSO
Tabulação	Windows	SUCESSO
Texto	Linux	SUCESSO
Binário	Linux	SUCESSO
Tabulação	Linux	SUCESSO

Importação de Dietas

Tipo	Ambiente	Status
Texto	Windows	SUCESSO
Binário	Windows	SUCESSO
Tabulação	Windows	SUCESSO
Texto	Linux	SUCESSO
Binário	Linux	SUCESSO
Tabulação	Linux	SUCESSO

Importação de Planos

Tipo	Ambiente	Status
Texto	Windows	SUCESSO
Binário	Windows	SUCESSO
Tabulação	Windows	SUCESSO
Texto	Linux	SUCESSO
Binário	Linux	SUCESSO
Tabulação	Linux	SUCESSO

1.1.7.3 Parâmetros e Menu

Funcionalidade	Status
Menu Ajuda	SUCESSO
Menu Erros	SUCESSO
Apagar Base Dados	SUCESSO
Funcionalidade 1	SUCESSO
Funcionalidade 2	SUCESSO

Funcionalidade	Status
Funcionalidade 3	SUCESSO
Funcionalidade 4	SUCESSO
Funcionalidade 5	SUCESSO
Funcionalidade 6	SUCESSO

1.1.8 Exemplos Detalhados de Testes

A seguir, são apresentados detalhes específicos de alguns dos testes realizados, incluindo exemplos e expectativas.

1.1.8.1 Teste de compilação da biblioteca/programa - Windows

Compilação da biblioteca IO:
```
Build started at 16:42...
1>------ Build started: Project: GestorDadosIO, Configuration: Debug x64 ------
1>LINK : C:\Users\user\Documents\GitHub\d-14\src\..\build\lib\GestorDadosIO.dll not found or not built by
      the last incremental link; performing full link
1>   Creating library C:\Users\user\Documents\GitHub\d-14\src\..\build\lib\GestorDadosIO.lib and object
      C:\Users\user\Documents\GitHub\d-14\src\..\build\lib\GestorDadosIO.exp
1>GestorDadosIO.vcxproj -> C:\Users\user\Documents\GitHub\d-14\build\lib\GestorDadosIO.dll
1>Done building project "GestorDadosIO.vcxproj".
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
========== Build completed at 16:42 and took 00.821 seconds ==========
```
Compilação do programa:
```
Build started at 16:42...
1>------ Build started: Project: LESI_PI_TP_a28602, Configuration: Debug x64 ------
1>Debug.c
1>main.c
1>Menu.c
1>OperacoesFicheiros.c
1>Utils.c
1>OperacoesIniciais.c
1>Generating Code...
1>LESI_PI_TP_a28602.vcxproj -> C:\Users\user\Documents\GitHub\d-14\build\prog\bemestar.exe
1>C:\Users\user\Documents\GitHub\d-14\src\..\build\lib\GestorDadosIO.dll
1>1 File(s) copied
1>Done building project "LESI_PI_TP_a28602.vcxproj".
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
========== Build completed at 16:42 and took 03.845 seconds ==========
```
Resultado: *Sucesso.* A biblioteca `.dll` e o programa `.exe` foram gerados com sucesso.

1.1.8.2 Teste de compilação da biblioteca/programa - Linux

Utilizando o makefile:
```
user@DESKTOP-2TL1NH1:~/d-14$ make
mkdir -p build
gcc -shared -o libGestorDadosIO.so src/GestorDadosIO/ExportarDados.o src/GestorDadosIO/ImportarDados.o
gcc -Wall -Wextra -Wno-unknown-pragmas -o BemEstar src/LESI_PI_TP_a28602/Debug.o
      src/LESI_PI_TP_a28602/main.o src/LESI_PI_TP_a28602/Menu.o src/LESI_PI_TP_a28602/OperacoesFicheiros.o
      src/LESI_PI_TP_a28602/OperacoesIniciais.o src/LESI_PI_TP_a28602/Utils.o libGestorDadosIO.so
      -Isrc/GestorDadosIO -L. -lGestorDadosIO -Wl,-rpath,'$ORIGIN'
mv BemEstar libGestorDadosIO.so build
```
Resultado: *Sucesso.* A biblioteca `.so` e o programa foram gerados com sucesso.

1.1.8.3 Teste do parâmetro <tt>ajuda</tt> - Detalhes

- **Comando de Teste:** `./bemestar -ajuda`
- **Esperado:** Apresentação do menu de ajuda do programa Bem Estar.
- **Resultado:** *Sucesso.*

1.1.8.4 Importação de Pacientes de Ficheiro Texto

- **Teste efetuado:** Ficheiro de texto:
  ```
  0001;Paula;123456789
  0002;Joao;987654321
  0003;Maria;111223344
  0004;Carlos;555566667
  0005;Ana;888777666
  0006;Pedro;999888777
  0007;Luisa;444555666
  ```

```
0008;Miguel;333444555
0009;Isabel;222333444
0010;Ricardo;111222333
```

Comando de Teste:
```
./bemestar.exe -p pacientes.txt
```

- **Esperado:** Importação bem-sucedida dos dados dos pacientes a partir de um ficheiro de texto no ambiente Windows.
- **Resultado:** *Sucesso.*

1.1.8.5 Importação de Dietas de Ficheiro Tab

- **Teste efetuado:** Ficheiro de texto separado por tab:
```
0001    01-01-2023  pequeno almoco  pao 100 cal
0001    01-01-2023  pequeno almoco  banana  120 cal
0001    02-01-2023  pequeno almoco  pao 100 cal
0001    02-01-2023  pequeno almoco  maca    180 cal
0003    14-02-2023  almoco  sopa    120 cal
0004    15-03-2023  almoco  salada  300 cal
0003    20-04-2023  pequeno almoco  cereais 150 cal
0002    23-05-2023  jantar  prato de carne  1200 cal
0005    10-06-2023  jantar  peixe grelhado  500 cal
0002    02-08-2023  almoco  frango  800 cal
0006    03-09-2023  pequeno almoco  iogurte 80 cal
0007    12-10-2023  jantar  pizza   900 cal
0008    25-11-2023  almoco  sanduiche   350 cal
0009    05-12-2023  jantar  hamburguer  700 cal
0010    15-01-2024  pequeno almoco  muffin  200 cal
```

Comando de Teste:
```
./bemestar -d dietas_tab.txt -tab
```

- **Esperado:** Importação bem-sucedida dos dados das dietas a partir de um ficheiro de texto separado por tabs no ambiente Linux.
- **Resultado:** *Sucesso.*

1.1.8.6 Importação de Planos de Ficheiro Binário

- **Teste efetuado:** Comando de Teste:
```
./bemestar.exe -l planos_bin.txt -bin
```
- **Esperado:** Importação bem-sucedida dos dados dos planos a partir de um ficheiro binário no ambiente Windows.
- **Resultado:** *Sucesso.*

1.1.9 Documentação LaTeX

Foi pedido a utilização de um ficheiro `.bib` com o LaTeX para adicionar as referências bibliográficas consultadas ao longo do desenvolvimento do trabalho.
Para isto, foi necessário adicionar o seguinte ao ficheiro LaTeX principal `refman.tex`:
```
% Bibliography
\newpage
\phantomsection
\addcontentsline{toc}{chapter}{\bibname}
\bibliographystyle{unsrt}
\nocite{*}
\bibliography{references}
```
E adicionar um ficheiro `references.bib`` que contém os dados da bibliografia:
```
@book{KernighanC89,
  author    = {Brian W. Kernighan and Dennis M. Ritchie},
  title     = {The C Programming Language},
  year      = {1988},
  publisher = {Prentice Hall},
  address   = {Englewood Cliffs, NJ},
  isbn      = {978-0131103627},
}

@online{CProgrammingTutorial,
  author    = {GeeksforGeeks},
  title     = {C Programming Language},
  year      = {2022},
```

```
  url       = {https://www.geeksforgeeks.org/c-programming-language/},
}

@online{LearnC,
  author    = {Learn-C.org},
  title     = {Learn C - Free Interactive C Tutorial},
  year      = {2023},
  url       = {https://www.learn-c.org/},
}

@online{TutorialsPoint,
  author    = {tutorialspoint.com},
  title     = {Learn C Programming},
  year      = {2023},
  url       = {https://www.tutorialspoint.com},
}

@online{ChatGPT,
  author    = {OpenAI},
  title     = {ChatGPT},
  year      = {2023},
  url       = {https://www.chat.openai.com},
}
```

Também é pedido "algum contexto para a definição de um grafo para representar alguma informação (exemplo: a hierarquia entre pastas do sistema de ficheiros)" dentro do ficheiro Tex. Por isso, com a dependência `dirtree`, adicionei o tree da hierarquia das pastas do projeto:

```
% Hierarquia de Ficheiros
  \dirtree{%
  .1 /.
  .2 assets/.
  .3 (imagens).
  .2 build/.
  .2 doc/.
  .3 v0.1/.
  .4 html/.
  .5 (website).
  .4 latex/.
  .5 (ficheiros Latex e PDF).
  .3 v1.0/.
  .4 Biblioteca/.
  .5 html/.
  .6 (website).
  .5 latex/.
  .6 (ficheiros Latex e PDF).
  .4 Programa/.
  .5 html/.
  .6 (website).
  .5 latex/.
  .6 (ficheiros Latex e PDF).
  .2 examples/.
  .3 (ficheiros de exemplo).
  .2 src/.
  .3 GestorDadosIO/.
  .4 (código da biblioteca).
  .3 LESI\_PI\_TP\_a28602/.
  .4 (código-fonte principal).
  .2 README.md.
  .2 Makefile.
  .2 .gitignore.
  .2 .gitattributes.
}
```

Foi necessário adicionar a dependência `dirtree` ao ficheiro:

```
\usepackage{dirtree}
```

A parte final do ficheiro `.tex` ficou assim:

```
%--- End generated contents ---
% Hierarquia de Ficheiros
  \backmatter
  \newpage
  \section*{Hierarquia de Ficheiros}
% Hierarquia de Ficheiros
  \dirtree{%
  .1 /.
  .2 assets/.
  .3 (imagens).
  .2 build/.
  .2 doc/.
  .3 v0.1/.
  .4 html/.
  .5 (website).
  .4 latex/.
  .5 (ficheiros Latex e PDF).
  .3 v1.0/.
  .4 Biblioteca/.
  .5 html/.
  .6 (website).
```

```
  .5 latex/.
  .6 (ficheiros Latex e PDF).
  .4 Programa/.
  .5 html/.
  .6 (website).
  .5 latex/.
  .6 (ficheiros Latex e PDF).
  .2 examples/.
  .3 (ficheiros de exemplo).
  .2 src/.
  .3 GestorDadosIO/.
  .4 (código da biblioteca).
  .3 LESI\_PI\_TP\_a28602/.
  .4 (código-fonte principal).
  .2 README.md.
  .2 Makefile.
  .2 .gitignore.
  .2 .gitattributes.
}
% Bibliography
  \phantomsection
  \addcontentsline{toc}{chapter}{\bibname}
  \bibliographystyle{unsrt}
  \nocite{*}
  \bibliography{references}
% Index
  \newpage
  \phantomsection
  \clearemptydoublepage
  \addcontentsline{toc}{chapter}{\indexname}
  \printindex
% Required for some languages (in combination with latexdocumentpre from the header)
\end{document}
```

Para compilar o Tex utilizei os seguintes comandos:

```
pdflatex refman
bibtex refman
pdflatex refman
pdflatex refman
```

1.1.10 Utilização do Repositório

Este repositório foi organizado considerando os exemplos abordados nas aulas de Laboratórios de Informática. Aqui estão alguns pontos importantes do repositório:

- **README.md:** Utilizamos o Markdown para fazer uma apresentação clara e detalhada do conteúdo do repositório, incluindo uma breve descrição do projeto, instruções de instalação e uso, e outros detalhes relevantes.
- ∗∗.gitignore:∗∗ Configuramos o ficheiro `.gitignore` para excluir tipos de ficheiros não desejados no repositório.
- **Atualizações Locais e Remotas:** Utilizamos os comandos Git apropriados para atualizar o código localmente e, de seguida, sincronizamos as alterações no repositório remoto no GitHub. Documentamos todas as alterações significativas para manter um histórico claro.
- **Divisão de Tarefas:** Todas as tarefas foram criadas pelo Enrique Rodrigues no Github como *Issues* para serem realizadas. Os membros interessados em assumir uma tarefa podiam-se designar como *Assignees*.
- **Branches:** Utilizamos branches de forma adequada para garantir a segurança das versões estáveis da solução. Cada funcionalidade ou correção de bug foi desenvolvida no seu próprio branch antes de ser colocada na branch principal.
- **Pull Requests:** Adotamos o uso de pull requests para documentar e revisar adequadamente as alterações propostas por cada membro da equipa. Isso proporciona uma visão clara das modificações introduzidas em relação à versão anterior.
- **Documentação:** O ficheiro README.md serve como documentação central do projeto, incluindo a divisão de tarefas, os principais comandos utilizados e uma conclusão geral do trabalho realizado.

1.1.11 Changelog e Versões

Para as últimas alterações e versões, consulte a seção `Releases`.

Capítulo 2

Índice das estruturas de dados

2.1 Estruturas de dados

Lista das estruturas de dados com uma breve descrição:

Capítulo 3

Índice dos ficheiros

3.1 Lista de ficheiros

Lista de todos os ficheiros documentados com uma breve descrição:

Capítulo 4

Documentação da estruturas de dados

4.1 Referência à estrutura Dieta

Campos de Dados

- int **idPaciente**
- time_t **data**
- int **refeicao**
- char **comida** [TAMANHO_NOME_REFEICAO]
- int **calorias**

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- EstruturaDados.h

4.2 Referência à estrutura Paciente

Campos de Dados

- int **id**
- char **nome** [TAMANHO_NOME_PACIENTE]
- long **tel**

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- EstruturaDados.h

4.3 Referência à estrutura Plano

Campos de Dados

- int **idPaciente**
- time_t **data**
- int **refeicao**
- int **calorias** [NUMERO_CALORIAS]

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- EstruturaDados.h

4.4 Referência à estrutura ResultadoProcessamentoFunc3

Campos de Dados

- int **idPaciente**

- char **nome** [TAMANHO_NOME_PACIENTE]
- char **refeicao** [TAMANHO_NOME_REFEICAO]
- int **totalCalorias**
- int **caloriasMin**
- int **caloriasMax**
- char **data** [TAMANHO_DATA_TEXTO]

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- EstruturaDados.h

4.5 Referência à estrutura ResultadoProcessamentoFunc4

Campos de Dados

- int **idPaciente**
- char **tipoRefeicao** [TAMANHO_REFEICAO]
- char **data** [TAMANHO_DATA_TEXTO]
- int **caloriasMin**
- int **caloriasMax**

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- EstruturaDados.h

4.6 Referência à estrutura ResultadoProcessamentoFunc5

Campos de Dados

- int **idPaciente**
- char **nomePaciente** [TAMANHO_NOME_PACIENTE]
- char **descricaoRefeicao** [TAMANHO_NOME_REFEICAO]
- int **totalCalorias**
- int **numRefeicoes**
- double **mediaCalorias**

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

- EstruturaDados.h

Capítulo 5

Documentação do ficheiro

5.1 Referência ao ficheiro Constantes.h

Ficheiro de cabeçalho contendo definições de constantes para o programa.

Macros

- #define **MODO_DEBUG** 1
- #define **TAMANHO_NOME_PACIENTE** 50
- #define **TAMANHO_NOME_REFEICAO** 50
- #define **TAMANHO_MAX_ARRAY** 15
- #define **TAMANHO_DATA_TEXTO** 11
- #define **TAMANHO_REFEICAO** 20
- #define **LIMITE_MAX_CALORIAS** 99999
- #define **LIMITE_MAX_INPUT_ID_PACIENTE** 9999
- #define **LIMITE_MIN_INPUT_TIPOS_REFEICAO** 1
- #define **LIMITE_MAX_INPUT_TIPOS_REFEICAO** 3
- #define **LIMITE_MIN_INPUT_TIPOS_FICHEIRO** 1
- #define **LIMITE_MAX_INPUT_TIPOS_FICHEIRO** 6
- #define **NUMERO_CALORIAS** 2
- #define **TAMANHO_MAX_BUFFER_INPUT** 1000
- #define **TAMANHO_MAX_BUFFER_FICHEIRO** 5000
- #define **FICHEIRO_BIN_PACIENTES** "pacientes.dat"
- #define **FICHEIRO_BIN_DIETAS** "dietas_pacientes.dat"
- #define **FICHEIRO_BIN_PLANOS** "planos_pacientes.dat"
- #define **ANSI_COR_PRETO** "\x1b[30m"
- #define **ANSI_COR_VERMELHO** "\x1b[31m"
- #define **ANSI_COR_VERMELHO_MAIS** "\033[1;37;41m"
- #define **ANSI_COR_VERDE** "\x1b[32m"
- #define **ANSI_COR_AMARELO** "\x1b[33m"
- #define **ANSI_COR_AZUL** "\x1b[34m"
- #define **ANSI_COR_MAGENTA** "\x1b[35m"
- #define **ANSI_COR_CYAN** "\x1b[36m"
- #define **ANSI_COR_BRANCO** "\x1b[37m"
- #define **CONFIRMAR_SIM** 'y'
- #define **CONFIRMAR_NAO** 'n'
- #define **ANSI_INFO** ANSI_COR_AZUL
- #define **ANSI_ERRO** ANSI_COR_VERMELHO
- #define **ANSI_SUCESSO** ANSI_COR_VERDE
- #define **ANSI_ALERTA** ANSI_COR_AMARELO
- #define **ANSI_DEBUG** ANSI_COR_MAGENTA
- #define **ANSI_RESET** "\x1b[0m"

- #define **ANSI_TITULO** "\033[4;1m"
- #define **TIPO_FICHEIRO_TXT** 1
- #define **TIPO_FICHEIRO_BIN** 2
- #define **TIPO_FICHEIRO_TAB** 3
- #define **TIPO_DADOS_PACIENTES** 1
- #define **TIPO_DADOS_DIETAS** 2
- #define **TIPO_DADOS_PLANOS** 3
- #define **FICHEIRO_PACIENTES_TXT** 1
- #define **FICHEIRO_PACIENTES_TAB** 2
- #define **FICHEIRO_DIETAS_TXT** 3
- #define **FICHEIRO_DIETAS_TAB** 4
- #define **FICHEIRO_PLANOS_TXT** 5
- #define **FICHEIRO_PLANOS_TAB** 6
- #define **ERRO_REFEICAO_ID** -1
- #define **PEQUENO_ALMOCO_ID** 1
- #define **ALMOCO_ID** 2
- #define **JANTAR_ID** 3
- #define **PEQUENO_ALMOCO** "pequeno almoco"
- #define **ALMOCO** "almoco"
- #define **JANTAR** "jantar"
- #define **SUCESSO** 0
- #define **FICHEIRO_EXISTE** 1
- #define **FICHEIRO_NAO_EXISTE** 0
- #define **ERRO_NAO_EXISTEM_FICHEIROS_BIN** 1
- #define **ERRO_CRIAR_FICHEIRO** 2
- #define **ERRO_IMPORTAR_DADOS_TEXTO** 3
- #define **ERRO_IMPORTAR_DADOS_BIN** 4
- #define **ERRO_APAGAR_FICHEIROS_BIN** 5
- #define **ERRO_FICHEIRO_NAO_ENCONTRADO** 7
- #define **ERRO_FICHEIRO_VAZIO** 8
- #define **ERRO_LEITURA_FICHEIRO** 9
- #define **ERRO_ALOCAR_MEMORIA** 6
- #define **ERRO_FORMATO_INVALIDO** 10
- #define **ERRO_NAO_DEFINIDO** 11
- #define **ERRO_ABRIR_FICHEIRO** 12
- #define **ERRO_FICHEIRO_NULLO** 24
- #define **ERRO_UTILIZADOR_NAO_DEU_PERMISSAO** 25
- #define **ERRO_VALOR_NULLO** 26
- #define **ERRO_TAMANHO_EXCEDIDO** 13
- #define **ERRO_CONVERSAO_TEXTO_DATETIME** 14
- #define **ERRO_ARRAY_DADOS_NULL** 15
- #define **ERRO_OPCAO_UTILIZADOR_DESCONHECIDA** 16
- #define **ERRO_APAGAR_BASE_DADOS** 17
- #define **ERRO_PROCESSAMENTO** 18
- #define **ERRO_IMPORTAR_DADOS** 19
- #define **ERRO_LEITURA_DATA** 20
- #define **ERRO_LEITURA_LIMITE_CALORIAS** 21
- #define **ERRO_LEITURA_ID_PACIENTE** 22
- #define **ERRO_LEITURA_TIPO_REFEICAO** 23
- #define **ERRO_LEITURA_TIPO_FICHEIRO** 27
- #define **FORMATO_PACIENTES_TXT** "%d;%49[^;];%ld"
- #define **FORMATO_PACIENTES_TAB** "%d\t%49[^\t]\t%ld"
- #define **FORMATO_DIETAS_TXT** "%d; %10[^;]; %19[^;]; %49[^;]; %d cal"
- #define **FORMATO_DIETAS_TAB** "%d\t %10[^\t]\t %19[^\t]\t %49[^\t]\t %d cal"
- #define **FORMATO_PLANOS_TXT** "%d; %10[^;]; %19[^;]; %d Cal, %d Cal"

- #define **FORMATO_PLANOS_TAB** "%d\t %10[^\t]\t %19[^\t]\t %d Cal, %d Cal"
- #define **OUTPUT_FORMATO_DEBUG_PACIENTES_MEMORIA** "PID: %d, Nome: %s, Tel: %ld\n"
- #define **OUTPUT_FORMATO_DEBUG_DIETAS_MEMORIA_WIN** "PID: %d, Data: %s (%lld), Refeicao: %s (tipo: %d), Comida: %s, Calorias: %d\n"
- #define **OUTPUT_FORMATO_DEBUG_DIETAS_MEMORIA_LINUX** "PID: %d, Data: %s (%ld), Refeicao: %s (tipo: %d), Comida: %s, Calorias: %d\n"
- #define **OUTPUT_FORMATO_DEBUG_PLANOS_MEMORIA_WIN** "PID: %d, Data: %s (%lld), Refeicao↩ : %s (tipo: %d), Calorias entre: %d e %d\n"
- #define **OUTPUT_FORMATO_DEBUG_PLANOS_MEMORIA_LINUX** "PID: %d, Data: %s (%ld), Refeicao: %s (tipo: %d), Calorias entre: %d e %d\n"

5.1.1 Descrição detalhada

Ficheiro de cabeçalho contendo definições de constantes para o programa.
Este ficheiro define várias constantes utilizadas ao longo do programa, incluindo nomes de ficheiros, códigos de cor ANSI e mensagens de erro.

Autor

Enrique George Rodrigues

Data

Criado: 18.11.2023

Modificado: 28.12.2023

Copyright

5.2 Constantes.h

Ir para a documentação deste ficheiro.

```
00001
00018 #ifndef CONSTANTES_H
00019 #define CONSTANTES_H
00020
00021 // Modo Debug
00022 #define MODO_DEBUG 1
00023
00024 // Tamanhos pré-defenidos
00025 #define TAMANHO_NOME_PACIENTE              50 // Tamanho máximo de um nome de paciente
00026 #define TAMANHO_NOME_REFEICAO              50 // Tamanho máximo de um nome de uma refeição
00027 #define TAMANHO_MAX_ARRAY                  15 // Tamanho máximo de arrays de pacientes/dietas/planos
00028 #define TAMANHO_DATA_TEXTO                 11 // Tamanho de uma data no formato ``DD-MM-AAAA" com
      '\0' no fim.
00029 #define TAMANHO_REFEICAO                   20 // Tamanho máximo de uma refeição
00030 #define LIMITE_MAX_CALORIAS                99999 // Limite máximo de calorias
00031 #define LIMITE_MAX_INPUT_ID_PACIENTE       9999 // Número máximo do input possível para o ID paciente
00032 #define LIMITE_MIN_INPUT_TIPOS_REFEICAO    1 // A primeira refeição começa com o valor de 1
00033 #define LIMITE_MAX_INPUT_TIPOS_REFEICAO    3 // Só temos 3 tipos de refeiçoes
00034 #define LIMITE_MIN_INPUT_TIPOS_FICHEIRO    1 // O ficheiro de tipo pacientes tem o valor de 1, logo
      não pode ser inferior a 1
00035 #define LIMITE_MAX_INPUT_TIPOS_FICHEIRO    6 // 3 tipos de ficheiros (pacientes, dietas, planos) e
      dois formatos para cada (txt ou tab)
00036 #define NUMERO_CALORIAS                    2 // Calorias mínimas e máximas, array de tamanho 2
00037 #define TAMANHO_MAX_BUFFER_INPUT           1000 // Tamanho máximo para o buffer de input de uma linha
      quando importamos do stdin
00038 #define TAMANHO_MAX_BUFFER_FICHEIRO        5000 // Tamanho máximo do buffer para o ficheiro todo
      quando importamos do stdin
00039
00040 // Nomes de ficheiros binários
00041 #define FICHEIRO_BIN_PACIENTES         "pacientes.dat"
00042 #define FICHEIRO_BIN_DIETAS            "dietas_pacientes.dat"
00043 #define FICHEIRO_BIN_PLANOS            "planos_pacientes.dat"
00044
00045 // Códigos de cor ANSI
00046 #define ANSI_COR_PRETO                 "\x1b[30m"
00047 #define ANSI_COR_VERMELHO              "\x1b[31m"
00048 #define ANSI_COR_VERMELHO_MAIS         "\033[1;37;41m"
00049 #define ANSI_COR_VERDE                 "\x1b[32m"
```

```
00050 #define ANSI_COR_AMARELO                "\x1b[33m"
00051 #define ANSI_COR_AZUL                   "\x1b[34m"
00052 #define ANSI_COR_MAGENTA                "\x1b[35m"
00053 #define ANSI_COR_CYAN                   "\x1b[36m"
00054 #define ANSI_COR_BRANCO                 "\x1b[37m"
00055
00056 // Respostas de confirmação
00057 #define CONFIRMAR_SIM                   'y'
00058 #define CONFIRMAR_NAO                   'n'
00059
00060 // Tipos de mensagens
00061 #define ANSI_INFO                       ANSI_COR_AZUL
00062 #define ANSI_ERRO                       ANSI_COR_VERMELHO
00063 #define ANSI_SUCESSO                    ANSI_COR_VERDE
00064 #define ANSI_ALERTA                     ANSI_COR_AMARELO
00065 #define ANSI_DEBUG                      ANSI_COR_MAGENTA
00066 #define ANSI_RESET                      "\x1b[0m"
00067 #define ANSI_TITULO                     "\033[4;1m"
00068
00069 // Tipos de ficheiros
00070 #define TIPO_FICHEIRO_TXT               1
00071 #define TIPO_FICHEIRO_BIN               2
00072 #define TIPO_FICHEIRO_TAB               3
00073
00074 // Tipos de Dados de ficheiros
00075 #define TIPO_DADOS_PACIENTES            1
00076 #define TIPO_DADOS_DIETAS               2
00077 #define TIPO_DADOS_PLANOS               3
00078
00079 // Leitura do stdin
00080 #define FICHEIRO_PACIENTES_TXT                  1
00081 #define FICHEIRO_PACIENTES_TAB                  2
00082 #define FICHEIRO_DIETAS_TXT                     3
00083 #define FICHEIRO_DIETAS_TAB                     4
00084 #define FICHEIRO_PLANOS_TXT                     5
00085 #define FICHEIRO_PLANOS_TAB                     6
00086
00087 // Tipos de refeicoes
00088 #define ERRO_REFEICAO_ID                -1
00089 #define PEQUENO_ALMOCO_ID               1
00090 #define ALMOCO_ID                       2
00091 #define JANTAR_ID                       3
00092
00093 // Nomes de refeicoes
00094 #define PEQUENO_ALMOCO                  "pequeno almoco"
00095 #define ALMOCO                          "almoco"
00096 #define JANTAR                          "jantar"
00097
00098 // Código de sucesso
00099 #define SUCESSO                         0
00100
00101 // Códigos de existência de ficheiros
00102 #define FICHEIRO_EXISTE                 1
00103 #define FICHEIRO_NAO_EXISTE             0
00104
00105 // Erros de Operações de Ficheiros
00106 #define ERRO_NAO_EXISTEM_FICHEIROS_BIN          1
00107 #define ERRO_CRIAR_FICHEIRO                     2
00108 #define ERRO_IMPORTAR_DADOS_TEXTO               3
00109 #define ERRO_IMPORTAR_DADOS_BIN                 4
00110 #define ERRO_APAGAR_FICHEIROS_BIN               5
00111 #define ERRO_FICHEIRO_NAO_ENCONTRADO            7
00112 #define ERRO_FICHEIRO_VAZIO                     8
00113 #define ERRO_LEITURA_FICHEIRO                   9
00114
00115 // Erros de Gestão de Memória
00116 #define ERRO_ALOCAR_MEMORIA                     6
00117
00118 // Erros de Formato de Ficheiro e Análise
00119 #define ERRO_FORMATO_INVALIDO                   10
00120
00121 // Erros Gerais
00122 #define ERRO_NAO_DEFINIDO                       11
00123 #define ERRO_ABRIR_FICHEIRO                     12
00124 #define ERRO_FICHEIRO_NULLO                     24
00125 #define ERRO_UTILIZADOR_NAO_DEU_PERMISSAO       25
00126 #define ERRO_VALOR_NULLO                        26
00127
00128 // Erros de Tamanho e Limites
00129 #define ERRO_TAMANHO_EXCEDIDO                   13
00130
00131 // Erros de Conversão de Dados
00132 #define ERRO_CONVERSAO_TEXTO_DATETIME           14
00133
00134 // Erros de Processamento de Dados
00135 #define ERRO_ARRAY_DADOS_NULL                   15
00136 #define ERRO_OPCAO_UTILIZADOR_DESCONHECIDA      16
```

```
00137 #define ERRO_APAGAR_BASE_DADOS                17
00138 #define ERRO_PROCESSAMENTO                    18
00139 #define ERRO_IMPORTAR_DADOS                   19
00140
00141 // Erros de Manipulação de Inputs
00142 #define ERRO_LEITURA_DATA                     20
00143 #define ERRO_LEITURA_LIMITE_CALORIAS          21
00144 #define ERRO_LEITURA_ID_PACIENTE              22
00145 #define ERRO_LEITURA_TIPO_REFEICAO            23
00146 #define ERRO_LEITURA_TIPO_FICHEIRO            27
00147
00148 // Formato dos ficheiros de texto
00149 #define FORMATO_PACIENTES_TXT                 "%d;%49[^;];%ld"
00150 #define FORMATO_PACIENTES_TAB                 "%d\t%49[^\t]\t%ld"
00151 #define FORMATO_DIETAS_TXT                    "%d; %10[^;]; %19[^;]; %49[^;]; %d cal"
00152 #define FORMATO_DIETAS_TAB                    "%d\t %10[^\t]\t %19[^\t]\t %49[^\t]\t %d cal"
00153 #define FORMATO_PLANOS_TXT                    "%d; %10[^;]; %19[^;]; %d Cal, %d Cal"
00154 #define FORMATO_PLANOS_TAB                    "%d\t %10[^\t]\t %19[^\t]\t %d Cal, %d Cal"
00155
00156 // Constantes de formato de output
00157 #define OUTPUT_FORMATO_DEBUG_PACIENTES_MEMORIA          "PID: %d, Nome: %s, Tel: %ld\n"
00158 #define OUTPUT_FORMATO_DEBUG_DIETAS_MEMORIA_WIN         "PID: %d, Data: %s (%lld), Refeicao: %s (tipo:
      %d), Comida: %s, Calorias: %d\n"
00159 #define OUTPUT_FORMATO_DEBUG_DIETAS_MEMORIA_LINUX       "PID: %d, Data: %s (%ld), Refeicao: %s (tipo:
      %d), Comida: %s, Calorias: %d\n"
00160 #define OUTPUT_FORMATO_DEBUG_PLANOS_MEMORIA_WIN         "PID: %d, Data: %s (%lld), Refeicao: %s (tipo:
      %d), Calorias entre: %d e %d\n"
00161 #define OUTPUT_FORMATO_DEBUG_PLANOS_MEMORIA_LINUX       "PID: %d, Data: %s (%ld), Refeicao: %s (tipo:
      %d), Calorias entre: %d e %d\n"
00162
00163 #endif // CONSTANTES_H
```

5.3 Referência ao ficheiro ControloErros.c

Funções que lidam com erros.

```
#include <stdio.h>
#include "Constantes.h"
#include "Utils.h"
```

Funções

- void LidarErrosImportacaoDados (int codigoErro)

  Lida com os erros da importação de dados.
- void LidarErroVerificacaoFicheiroBinario (int codigoErro)

  Lida com erros na verificação da existência dos ficheiros da base de dados.
- void LidarErroPacientesUltrapassamCalorias (int codigoErro)

  Lida com os erros da funcionalidade 2.
- void LidarErroPacientesForaPlano (int codigoErro)

  Lida com os erros da funcionalidade 3.
- void LidarErroListarPlanoNutricional (int codigoErro)

  Lida com os erros da funcionalidade 4.
- void LidarErroListarMediasCaloriasPorRefeicao (int codigoErro)

  Lida com os erros da funcionalidade 5.
- void LidarErroGerarTabelaRefeicoes (int codigoErro)

  Lida com os erros da funcionalidade 6.
- void LidarErroGuardarDados (int codigoErro)

  Lida com os erros ao guardar dados.

5.3.1 Descrição detalhada

Funções que lidam com erros.

Autor

Enrique George Rodrigues

Data

Criado: 27.12.2023

Modificado: 28.12.2023

Copyright

5.3.2 Documentação das funções

5.3.2.1 LidarErroGerarTabelaRefeicoes()

```
void LidarErroGerarTabelaRefeicoes (
            int codigoErro )
```

Lida com os erros da funcionalidade 6.

Parâmetros

codigoErro	- O código de erro.

5.3.2.2 LidarErroGuardarDados()

```
void LidarErroGuardarDados (
            int codigoErro )
```

Lida com os erros ao guardar dados.

Parâmetros

codigoErro	- O código de erro.

5.3.2.3 LidarErroListarMediasCaloriasPorRefeicao()

```
void LidarErroListarMediasCaloriasPorRefeicao (
            int codigoErro )
```

Lida com os erros da funcionalidade 5.

Parâmetros

codigoErro	- O código de erro.

5.3.2.4 LidarErroListarPlanoNutricional()

```
void LidarErroListarPlanoNutricional (
            int codigoErro )
```

Lida com os erros da funcionalidade 4.

Parâmetros

codigoErro	- O código de erro.

5.3.2.5 LidarErroPacientesForaPlano()

```
void LidarErroPacientesForaPlano (
            int codigoErro )
```

Lida com os erros da funcionalidade 3.

Parâmetros

codigoErro	- O código de erro.

5.3.2.6 LidarErroPacientesUltrapassamCalorias()

```
void LidarErroPacientesUltrapassamCalorias (
            int codigoErro )
```

Lida com os erros da funcionalidade 2.

Parâmetros

codigoErro	- O código do erro.

5.3.2.7 LidarErrosImportacaoDados()

```
void LidarErrosImportacaoDados (
            int codigoErro )
```

Lida com os erros da importação de dados.

Parâmetros

codigoErro	- O código do erro.

5.3.2.8 LidarErroVerificacaoFicheiroBinario()

```
void LidarErroVerificacaoFicheiroBinario (
            int codigoErro )
```

Lida com erros na verificação da existência dos ficheiros da base de dados.

Parâmetros

codigoErro	- O código do erro.

5.4 Referência ao ficheiro ControloErros.h

Funções que lidam com erros.

Funções

- void LidarErrosImportacaoDados (int codigoErro)

  Lida com os erros da importação de dados.
- void LidarErroVerificacaoFicheiroBinario (int codigoErro)

  Lida com erros na verificação da existência dos ficheiros da base de dados.
- void LidarErroPacientesUltrapassamCalorias (int codigoErro)

  Lida com os erros da funcionalidade 2.
- void LidarErroPacientesForaPlano (int codigoErro)

  Lida com os erros da funcionalidade 3.
- void LidarErroListarPlanoNutricional (int codigoErro)

Lida com os erros da funcionalidade 4.

- void LidarErroListarMediasCaloriasPorRefeicao (int codigoErro)

  Lida com os erros da funcionalidade 5.
- void LidarErroGerarTabelaRefeicoes (int codigoErro)

  Lida com os erros da funcionalidade 6.
- void LidarErroGuardarDados (int codigoErro)

  Lida com os erros ao guardar dados.

5.4.1 Descrição detalhada

Funções que lidam com erros.
~

Autor

Enrique George Rodrigues

Data

Criado: 27.12.2023

Modificado: 28.12.2023

Copyright

5.4.2 Documentação das funções

5.4.2.1 LidarErroGerarTabelaRefeicoes()

```
void LidarErroGerarTabelaRefeicoes (
            int codigoErro )
```

Lida com os erros da funcionalidade 6.

Parâmetros

codigoErro	- O código de erro.

5.4.2.2 LidarErroGuardarDados()

```
void LidarErroGuardarDados (
            int codigoErro )
```

Lida com os erros ao guardar dados.

Parâmetros

codigoErro	- O código de erro.

5.4.2.3 LidarErroListarMediasCaloriasPorRefeicao()

```
void LidarErroListarMediasCaloriasPorRefeicao (
            int codigoErro )
```

Lida com os erros da funcionalidade 5.

Parâmetros

codigoErro	- O código de erro.

5.4.2.4 LidarErroListarPlanoNutricional()

```
void LidarErroListarPlanoNutricional (
            int codigoErro )
```

Lida com os erros da funcionalidade 4.

Parâmetros

codigoErro	- O código de erro.

5.4.2.5 LidarErroPacientesForaPlano()

```
void LidarErroPacientesForaPlano (
            int codigoErro )
```

Lida com os erros da funcionalidade 3.

Parâmetros

codigoErro	- O código de erro.

5.4.2.6 LidarErroPacientesUltrapassamCalorias()

```
void LidarErroPacientesUltrapassamCalorias (
            int codigoErro )
```

Lida com os erros da funcionalidade 2.

Parâmetros

codigoErro	- O código do erro.

5.4.2.7 LidarErrosImportacaoDados()

```
void LidarErrosImportacaoDados (
            int codigoErro )
```

Lida com os erros da importação de dados.

Parâmetros

codigoErro	- O código do erro.

5.4.2.8 LidarErroVerificacaoFicheiroBinario()

```
void LidarErroVerificacaoFicheiroBinario (
            int codigoErro )
```

Lida com erros na verificação da existência dos ficheiros da base de dados.

Parâmetros

codigoErro	- O código do erro.

5.5 ControloErros.h

Ir para a documentação deste ficheiro.

```
00001
00017 #ifndef CONTROLO_ERROS_H
00018 #define CONTROLO_ERROS_H
00019
00024 void LidarErrosImportacaoDados(int codigoErro);
00025
00030 void LidarErroVerificacaoFicheiroBinario(int codigoErro);
00031
00036 void LidarErroPacientesUltrapassamCalorias(int codigoErro);
00037
00042 void LidarErroPacientesForaPlano(int codigoErro);
00043
00048 void LidarErroListarPlanoNutricional(int codigoErro);
00049
00054 void LidarErroListarMediasCaloriasPorRefeicao(int codigoErro);
00055
00060 void LidarErroGerarTabelaRefeicoes(int codigoErro);
00061
00066 void LidarErroGuardarDados(int codigoErro);
00067
00068 #endif
```

5.6 Referência ao ficheiro Debug.c

Funções que apresentam informações durante o modo debug.

```
#include <stdio.h>
#include "Constantes.h"
#include "EstruturaDados.h"
#include "Utils.h"
```

Funções

- void DebugMostrarFicheirosUtilizador (char *ficheiroPacientes, char *ficheiroDietas, char *ficheiroPlanos, int pacientesTipo, int dietasTipo, int planosTipo)

  Apresenta os nomes e tipos de ficheiros que o utilizador deseja importar.
- void DebugMostrarPacientes (Paciente pacientes[TAMANHO_MAX_ARRAY], int numPacientes)

  Apresenta os dados dos pacientes na memória para debugging.
- void DebugMostrarDietas (Dieta dietas[TAMANHO_MAX_ARRAY], int numDietas)

  Apresenta os dados das dietas na memória para debugging.
- void DebugMostrarPlanos (Plano planos[TAMANHO_MAX_ARRAY], int numPlanos)

  Apresenta os dados dos planos na memória para debugging.
- void DebugMostrarDadosMemoria (Paciente pacientes[TAMANHO_MAX_ARRAY], int numPacientes, Dieta dietas[TAMANHO_MAX_ARRAY], int numDietas, Plano planos[TAMANHO_MAX_ARRAY], int numPlanos)

  Apresenta os dados na memória para debugging.

5.6.1 Descrição detalhada

Funções que apresentam informações durante o modo debug.

Autor

Enrique George Rodrigues

Data

Criado: 16.11.2023

Modificado: 28.12.2023

Copyright

5.6.2 Documentação das funções

5.6.2.1 DebugMostrarDadosMemoria()

```
void DebugMostrarDadosMemoria (
            Paciente pacientes[TAMANHO_MAX_ARRAY],
            int numPacientes,
            Dieta dietas[TAMANHO_MAX_ARRAY],
            int numDietas,
            Plano planos[TAMANHO_MAX_ARRAY],
            int numPlanos )
```

Apresenta os dados na memória para debugging.

Parâmetros

pacientes	- Array de pacientes.
numPacientes	- Número de pacientes.
dietas	- Array de dietas.
numDietas	- Número de dietas.
planos	- Array de planos.
numPlanos	- Número de planos.

5.6.2.2 DebugMostrarDietas()

```
void DebugMostrarDietas (
            Dieta dietas[TAMANHO_MAX_ARRAY],
            int numDietas )
```

Apresenta os dados das dietas na memória para debugging.

Parâmetros

dietas	- Array de dietas.
numDietas	- Número de dietas.

5.6.2.3 DebugMostrarFicheirosUtilizador()

```
void DebugMostrarFicheirosUtilizador (
            char * ficheiroPacientes,
            char * ficheiroDietas,
            char * ficheiroPlanos,
            int pacientesTipo,
            int dietasTipo,
            int planosTipo )
```

Apresenta os nomes e tipos de ficheiros que o utilizador deseja importar.

Parâmetros

ficheiroPacientes	- O nome do ficheiro de pacientes.
ficheiroDietas	- O nome do ficheiro de dietas.
ficheiroPlanos	- O nome do ficheiro de planos.
tipoFicheiroPacientes	- O tipo do ficheiro de pacientes (txt, bin, tab).
tipoFicheiroDietas	- O tipo do ficheiro de dietas (txt, bin, tab).
tipoFicheiroPlanos	- O tipo do ficheiro de planos (txt, bin, tab).

5.6.2.4 DebugMostrarPacientes()

```
void DebugMostrarPacientes (
            Paciente pacientes[TAMANHO_MAX_ARRAY],
            int numPacientes )
```

Apresenta os dados dos pacientes na memória para debugging.

Parâmetros

pacientes	Array de pacientes.
numPacientes	Número de pacientes.

5.6.2.5 DebugMostrarPlanos()

```
void DebugMostrarPlanos (
            Plano planos[TAMANHO_MAX_ARRAY],
            int numPlanos )
```

Apresenta os dados dos planos na memória para debugging.

Parâmetros

planos	- Array de planos.
numPlanos	- Número de planos.

5.7 Referência ao ficheiro Debug.h

Funções que apresentam informações durante o modo debug.

```
#include <stdio.h>
#include "Constantes.h"
#include "EstruturaDados.h"
#include "Utils.h"
```

Funções

- void DebugMostrarFicheirosUtilizador (char ∗ficheiroPacientes, char ∗ficheiroDietas, char ∗ficheiroPlanos, int pacientesTipo, int dietasTipo, int planosTipo)

  Apresenta os nomes e tipos de ficheiros que o utilizador deseja importar.
- void DebugMostrarPacientes (Paciente pacientes[TAMANHO_MAX_ARRAY], int numPacientes)

  Apresenta os dados dos pacientes na memória para debugging.
- void DebugMostrarDietas (Dieta dietas[TAMANHO_MAX_ARRAY], int numDietas)

  Apresenta os dados das dietas na memória para debugging.
- void DebugMostrarPlanos (Plano planos[TAMANHO_MAX_ARRAY], int numPlanos)

  Apresenta os dados dos planos na memória para debugging.
- void DebugMostrarDadosMemoria (Paciente pacientes[TAMANHO_MAX_ARRAY], int numPacientes, Dieta dietas[TAMANHO_MAX_ARRAY], int numDietas, Plano planos[TAMANHO_MAX_ARRAY], int numPlanos)

  Apresenta os dados na memória para debugging.

5.7.1 Descrição detalhada

Funções que apresentam informações durante o modo debug.

∼

Autor

Enrique George Rodrigues

Data

Criado: 17.12.2023

Modificado: 28.12.2023

Copyright

5.7.2 Documentação das funções

5.7.2.1 DebugMostrarDadosMemoria()

```
void DebugMostrarDadosMemoria (
            Paciente pacientes[TAMANHO_MAX_ARRAY],
            int numPacientes,
            Dieta dietas[TAMANHO_MAX_ARRAY],
            int numDietas,
            Plano planos[TAMANHO_MAX_ARRAY],
            int numPlanos )
```

Apresenta os dados na memória para debugging.

Parâmetros

pacientes	- Array de pacientes.
numPacientes	- Número de pacientes.
dietas	- Array de dietas.
numDietas	- Número de dietas.
planos	- Array de planos.
numPlanos	- Número de planos.

5.7.2.2 DebugMostrarDietas()

```
void DebugMostrarDietas (
            Dieta dietas[TAMANHO_MAX_ARRAY],
            int numDietas )
```

Apresenta os dados das dietas na memória para debugging.

Parâmetros

dietas	- Array de dietas.
numDietas	- Número de dietas.

5.7.2.3 DebugMostrarFicheirosUtilizador()

```
void DebugMostrarFicheirosUtilizador (
            char * ficheiroPacientes,
            char * ficheiroDietas,
            char * ficheiroPlanos,
            int pacientesTipo,
            int dietasTipo,
            int planosTipo )
```

Apresenta os nomes e tipos de ficheiros que o utilizador deseja importar.

Parâmetros

ficheiroPacientes	- O nome do ficheiro de pacientes.
ficheiroDietas	- O nome do ficheiro de dietas.
ficheiroPlanos	- O nome do ficheiro de planos.
pacientesTipo	- O tipo do ficheiro de pacientes (txt, bin, tab).
dietasTipo	- O tipo do ficheiro de dietas (txt, bin, tab).
planosTipo	- O tipo do ficheiro de planos (txt, bin, tab).
ficheiroPacientes	- O nome do ficheiro de pacientes.
ficheiroDietas	- O nome do ficheiro de dietas.
ficheiroPlanos	- O nome do ficheiro de planos.
tipoFicheiroPacientes	- O tipo do ficheiro de pacientes (txt, bin, tab).
tipoFicheiroDietas	- O tipo do ficheiro de dietas (txt, bin, tab).
tipoFicheiroPlanos	- O tipo do ficheiro de planos (txt, bin, tab).

5.7.2.4 DebugMostrarPacientes()

```
void DebugMostrarPacientes (
            Paciente pacientes[TAMANHO_MAX_ARRAY],
            int numPacientes )
```

Apresenta os dados dos pacientes na memória para debugging.

Parâmetros

pacientes	Array de pacientes.
numPacientes	Número de pacientes.

5.7.2.5 DebugMostrarPlanos()

```
void DebugMostrarPlanos (
            Plano planos[TAMANHO_MAX_ARRAY],
            int numPlanos )
```

Apresenta os dados dos planos na memória para debugging.

Parâmetros

planos	- Array de planos.
numPlanos	- Número de planos.

5.8 Debug.h

Ir para a documentação deste ficheiro.

```
00001
00017 #ifndef DEBUG_H
00018 #define DEBUG_H
00019
00020 #include <stdio.h>
00021
00022 #include "Constantes.h"
00023 #include "EstruturaDados.h"
00024 #include "Utils.h"
00025
00035 void DebugMostrarFicheirosUtilizador(char* ficheiroPacientes,
```

```
00036     char* ficheiroDietas, char* ficheiroPlanos, int pacientesTipo, int dietasTipo, int planosTipo);
00037
00043 void DebugMostrarPacientes(Paciente pacientes[TAMANHO_MAX_ARRAY], int numPacientes);
00044
00050 void DebugMostrarDietas(Dieta dietas[TAMANHO_MAX_ARRAY], int numDietas);
00051
00057 void DebugMostrarPlanos(Plano planos[TAMANHO_MAX_ARRAY], int numPlanos);
00058
00068 void DebugMostrarDadosMemoria(Paciente pacientes[TAMANHO_MAX_ARRAY], int numPacientes,
00069     Dieta dietas[TAMANHO_MAX_ARRAY], int numDietas,
00070     Plano planos[TAMANHO_MAX_ARRAY], int numPlanos);
00071
00072 #endif
```

5.9 Referência ao ficheiro EstruturaDados.h

Definição das estruturas de dados para pacientes, dietas e planos.

```
#include <time.h>
#include "Constantes.h"
```

Estruturas de Dados

- struct Paciente
- struct Dieta
- struct Plano
- struct ResultadoProcessamentoFunc3
- struct ResultadoProcessamentoFunc4
- struct ResultadoProcessamentoFunc5

5.9.1 Descrição detalhada

Definição das estruturas de dados para pacientes, dietas e planos.
Este ficheiro contém as definições das estruturas de dados utilizadas no programa para representar informações sobre pacientes, dietas e planos. As estruturas incluem detalhes como identificação, nome, telefone, data, refeição, comida e calorias.

Autor

Enrique George Rodrigues

Data

Criado: 29.11.2023

Modificado: 28.12.2023

Copyright

5.10 EstruturaDados.h

Ir para a documentação deste ficheiro.

```
00001
00019 #ifndef ESTRUTURA_DADOS_H
00020 #define ESTRUTURA_DADOS_H
00021
00022 #include <time.h>
00023
00024 #include "Constantes.h"
00025
00026 // Estrutura dos pacientes
00027 typedef struct {
00028     int id; // Identificação única do paciente
00029     char nome[TAMANHO_NOME_PACIENTE]; // Nome do paciente
00030     long tel; // Número de telefone do paciente
```

```
00031 } Paciente;
00032
00033 // Estrutura das dietas
00034 typedef struct {
00035     int idPaciente; // Identificação única do paciente
00036     time_t data; // Data em formato timestamp
00037     int refeicao; // Tipo de refeição (PEQUENO_ALMOCO, ALMOCO, JANTAR)
00038     char comida[TAMANHO_NOME_REFEICAO]; // Nome da comida
00039     int calorias; // Quantidade de calorias da comida
00040 } Dieta;
00041
00042 // Estrutura dos planos
00043 typedef struct {
00044     int idPaciente; // Identificação única do paciente
00045     time_t data; // Data em formato timestamp
00046     int refeicao; // Tipo de refeição (PEQUENO_ALMOCO, ALMOCO, JANTAR)
00047     int calorias[NUMERO_CALORIAS]; // Intervalo de calorias. Índice 0 = cal mínimas, índice 1 = cal
      máximas
00048 } Plano;
00049
00050 // Utilizado na funcionalidade 3
00051 typedef struct {
00052     int idPaciente; // Identificação única do paciente
00053     char nome[TAMANHO_NOME_PACIENTE]; // Nome do paciente
00054     char refeicao[TAMANHO_NOME_REFEICAO]; // Nome da refeição
00055     int totalCalorias; // Valor total das calorias
00056     int caloriasMin; // Valor mínimo das calorias
00057     int caloriasMax; // Valor máximo das calorias
00058     char data[TAMANHO_DATA_TEXTO]; // Data em texto
00059 } ResultadoProcessamentoFunc3;
00060
00061 // Utilizado na funcionalidade 4
00062 typedef struct {
00063     int idPaciente; // Identificação única do paciente
00064     char tipoRefeicao[TAMANHO_REFEICAO]; // Tipo de refeição
00065     char data[TAMANHO_DATA_TEXTO]; // Data em texto
00066     int caloriasMin; // Calorias mínimas
00067     int caloriasMax; // Calorias máximas
00068 } ResultadoProcessamentoFunc4;
00069
00070 // Utilizado na funcionalidade 5
00071 typedef struct {
00072     int idPaciente; // Identificação única do paciente
00073     char nomePaciente[TAMANHO_NOME_PACIENTE]; // Nome do paciente
00074     char descricaoRefeicao[TAMANHO_NOME_REFEICAO]; // Refeição em texto
00075     int totalCalorias; // Total calorias
00076     int numRefeicoes; // Número de refeições
00077     double mediaCalorias; // Média das calorias em relação ao número de refeições
00078 } ResultadoProcessamentoFunc5;
00079
00080 #endif
```

5.11 Referência ao ficheiro InputStdin.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Constantes.h"
#include "Utils.h"
#include "Debug.h"
#include "../GestorDadosIO/IODadosAPI.h"
#include <unistd.h>
```

Funções

- int **ImportarDadosStdin** (Paciente pacientes[TAMANHO_MAX_ARRAY], int *numPacientes, Dieta dietas[TAMANHO_MAX_ARRAY], int *numDietas, Plano planos[TAMANHO_MAX_ARRAY], int *num↩ Planos)

5.11.1 Descrição detalhada

Autor

Enrique George Rodrigues

Data

29.12.2023

Copyright

5.12 Referência ao ficheiro InputStdin.h

Funções

- int **ImportarDadosStdin** (Paciente pacientes[TAMANHO_MAX_ARRAY], int *numPacientes, Dieta dietas[TAMANHO_MAX_ARRAY], int *numDietas, Plano planos[TAMANHO_MAX_ARRAY], int *numPlanos)

5.12.1 Descrição detalhada

~

Autor

Enrique George Rodrigues

Data

29.12.2023

Copyright

5.13 InputStdin.h

Ir para a documentação deste ficheiro.

```
00001
00011 #ifndef INPUT_STDIN_H
00012 #define INPUT_STDIN_H
00013
00014 int ImportarDadosStdin(Paciente pacientes[TAMANHO_MAX_ARRAY], int* numPacientes,
00015     Dieta dietas[TAMANHO_MAX_ARRAY], int* numDietas, Plano planos[TAMANHO_MAX_ARRAY], int* numPlanos);
00016
00017 #endif
```

5.14 Referência ao ficheiro main.c

Ponto de entrada principal da aplicação que apoia o bem-estar e cuidados nutricionais. Desenvolvido por Enrique Rodrigues na unidade curricular de Programação Imperativa.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
#include "Constantes.h"
#include "EstruturaDados.h"
#include "Utils.h"
#include "Menu.h"
#include "InputStdin.h"
#include "Parametros.h"
```

```
#include "ControloErros.h"
#include "OperacoesFicheiros.h"
#include "../GestorDadosIO/IODadosAPI.h"
#include "Debug.h"
```

Funções

- int main (int argc, char ∗argv[ ])

  Função principal que controla a execução do programa.

5.14.1 Descrição detalhada

Ponto de entrada principal da aplicação que apoia o bem-estar e cuidados nutricionais. Desenvolvido por Enrique Rodrigues na unidade curricular de Programação Imperativa.
O programa visa apoiar o bem-estar e cuidados nutricionais, abordando o problema do comportamento alimentar saudável. Funcionalidades incluem carregamento de dados de pacientes, dietas e planos nutricionais, apresentação de estatísticas de calorias, listagem de pacientes com comportamento fora do plano nutricional e análises diversas. Desenvolvido com base em métodos rigorosos de análise de problemas e as melhores práticas de programação imperativa em C.

Autor

Enrique George Rodrigues

Data

Criado: 16.11.2023

Modificado: 28.12.2023

Copyright

5.14.2 Documentação das funções

5.14.2.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Função principal que controla a execução do programa.
Esta função inicia o programa e controla a execução de diferentes blocos de código, incluindo o controlo dos parametros do utlizador, a verificação (e criação) dos ficheiros binários, a importação dos dados para memória do sistema, a apresentação de um menu para interação com o utilizador e a execução das opções escolhidas no menu.

Aviso

Esta função pode interagir com o utilizador para corrigir erros nos dados dos pacientes.

O utilizador pode apagar todos os dados dos pacientes se chamar o programa com o parámetro "--apagarbd".

O menu corre num ciclo infinito até ser escolhido 'q' (terminar e guardar dados) ou 'f' (terminar sem guardar).

Parâmetros

in	*argc*	O número de argumentos da linha de comandos.
in	*argv*	Um array de strings representando os argumentos da linha de comandos.

Retorna

0 - O programa correu sem erros (SUCESSO).

1 - Não existem os ficheiros binários (ERRO_NAO_EXISTEM_FICHEIROS_BIN).

2 - Erro ao criar o ficheiro (ERRO_CRIAR_FICHEIRO).

3 - Erro ao importar dados do texto (ERRO_IMPORTAR_DADOS_TEXTO).

4 - Erro ao importar dados binários (ERRO_IMPORTAR_DADOS_BIN).

5 - Erro ao apagar ficheiros binários (ERRO_APAGAR_FICHEIROS_BIN).

6 - Erro ao alocar memória (ERRO_ALOCAR_MEMORIA).

7 - Ficheiro não encontrado (ERRO_FICHEIRO_NAO_ENCONTRADO).

8 - Ficheiro vazio (ERRO_FICHEIRO_VAZIO).

9 - Erro durante a leitura do ficheiro (ERRO_LEITURA_FICHEIRO).

10 - Formato inválido (ERRO_FORMATO_INVALIDO).

11 - Erro não definido (ERRO_NAO_DEFINIDO).

12 - Erro ao abrir o ficheiro (ERRO_ABRIR_FICHEIRO).

13 - Erro tamanho array excedido (ERRO_TAMANHO_EXCEDIDO).

14 - Erro conversão data texto para objeto datetime (ERRO_CONVERSAO_TEXTO_DATETIME).

15 - Erro array de dados nullo (ERRO_ARRAY_DADOS_NULL).

16 - Erro opção introduzida pelo utilizador desconhecida (ERRO_OPCAO_UTILIZADOR_DESCONHECIDA).

17 - Erro apagar ficheiros da base de dados (.dat) (ERRO_APAGAR_BASE_DADOS).

18 - Erro processamento dos ficheiros (ERRO_PROCESSAMENTO).

19 - Erro importar dados dos ficheiros (ERRO_IMPORTAR_DADOS).

5.15 Referência ao ficheiro Menu.c

Implementação das funções relacionadas ao menu de opções do programa.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Constantes.h"
#include "Utils.h"
```

Funções

- void ApresentarMenu ()

  Apresenta o menu de opções para interação com o programa.
- bool LerData (char *data, const char *proposito)

  Lê uma string do utilizador para a data.
- int PacientesUltrapassaramCalorias (Dieta *dietas, int numDietas, int numPacientes)

  Função para executar a funcionalidade 2 - Pacientes que ultrapassam um dado limite de calorias entre determinadas datas.
- int PacientesForaPlano (Paciente *pacientes, Dieta *dietas, Plano *planos)

  Função que executa a funcionalidade 3 - apresentar pacientes com dietas fora do seu plano.
- int ListarPlanoNutricional (Plano planos[ ], int numPlanos)

  Lista o plano nutricional de um paciente para determinada refeição ao longo de um determinado período.
- int ListarMediasCaloriasPorRefeicao (Paciente pacientes[ ], Dieta dietas[ ])

  Lista médias de calorias por refeição para um intervalo de datas específico.
- int GerarTabelaRefeicoes (Paciente pacientes[ ], int numPacientes, Plano planos[ ], int numPlanos, Dieta dietas[ ], int numDietas)

  Gera a tabela de refeições.

5.15.1 Descrição detalhada

Implementação das funções relacionadas ao menu de opções do programa.

Autor

Enrique George Rodrigues

Data

Criado: 18.11.2023

Modificado: 28.12.2023

Copyright

5.15.2 Documentação das funções

5.15.2.1 ApresentarMenu()

```
void ApresentarMenu ( )
```

Apresenta o menu de opções para interação com o programa.
Esta função imprime no ecrã as opções disponíveis para interação com o programa. O utilizador é solicitado a escolher uma das opções apresentadas.

Nota

As opções incluem carregar dados, verificar pacientes com calorias ultrapassadas, verificar pacientes com calorias fora do intervalo, obter o plano nutricional de um paciente para uma refeição específica, calcular médias de calorias consumidas por refeição por cada paciente, e apresentar uma tabela das refeições planeadas e realizadas para todos os pacientes. As opções 'q' e 'f' fecham o programa.

Aviso

A função não lida com a leitura da escolha do utilizador e não verifica a validade da escolha inserida. Apenas apresenta o menu no stdout.

5.15.2.2 GerarTabelaRefeicoes()

```
int GerarTabelaRefeicoes (
            Paciente pacientes[],
            int numPacientes,
            Plano planos[],
            int numPlanos,
            Dieta dietas[],
            int numDietas )
```

Gera a tabela de refeições.

Parâmetros

pacientes	- Array de pacientes.
numPacientes	- Número de pacientes.
planos	- Array de planos.
numPlanos	- Número de planos.
dietas	- Array de dietas.
numDietas	- Número de dietas.

Valores retornados

-	0 se correr com sucesso.
-	26 se não foi possível encontrar um paciente.

5.15.2.3 LerData()

```
bool LerData (
            char * data,
            const char * proposito )
```

Lê uma string do utilizador para a data.

Parâmetros

data	- A variável onde se quer guardar a data.
proposito	- O objetivo da leitura (p. ex., "inicial" ou "final").

Valores retornados

-	Verdadeiro se correu sem erros, Falso se correu com erros.

5.15.2.4 ListarMediasCaloriasPorRefeicao()

```
int ListarMediasCaloriasPorRefeicao (
            Paciente pacientes[],
            Dieta dietas[] )
```

Lista médias de calorias por refeição para um intervalo de datas específico.

Parâmetros

pacientes	- O array de pacientes.
dietas	- O array de todas as dietas.
resultados	- O array de resultados.
numResultados	- O número de resultados.

Valores retornados

-	0 se correr com sucesso.
-	20 se houver um erro com a leitura de datas.

5.15.2.5 ListarPlanoNutricional()

```
int ListarPlanoNutricional (
            Plano planos[],
            int numPlanos )
```

Lista o plano nutricional de um paciente para determinada refeição ao longo de um determinado período.
Função que executa a funcionalidade 4 - Listar o plano nutricional de um paciente para determinada refeição ao longo de um determinado período.

Parâmetros

planos	- O array de planos nutricionais.

Parâmetros

numPlanos	- O número de planos nutricionais no array.

Valores retornados

-	0 se correr com sucesso.
-	20 se houver um erro com a leitura de datas (ERRO_LEITURA_DATA).
-	22 se não for possível ler o ID do paciente (ERRO_LEITURA_ID_PACIENTE).
-	23 se não for possível ler o tipo de refeição (ERRO_LEITURA_TIPO_REFEICAO).

5.15.2.6 PacientesForaPlano()

```
int PacientesForaPlano (
            Paciente * pacientes,
            Dieta * dietas,
            Plano * planos )
```

Função que executa a funcionalidade 3 - apresentar pacientes com dietas fora do seu plano.
Método que executa a funcionalidade 3 - apresentar pacientes com dietas fora do seu plano.

Parâmetros

in	*pacientes*	- O array de pacientes.
in	*dietas*	- O array de dietas.
in	*planos*	- O array de planos.

Valores retornados

0	se correr com sucesso.
20	se houver um erro com a leitura de datas (ERRO_LEITURA_DATA).

5.15.2.7 PacientesUltrapassaramCalorias()

```
int PacientesUltrapassaramCalorias (
            Dieta * dietas,
            int numDietas,
            int numPacientes )
```

Função para executar a funcionalidade 2 - Pacientes que ultrapassam um dado limite de calorias entre determinadas datas.

Parâmetros

dietas	- O array de dietas.
numDietas	- O número de dietas.
numPacientes	- O número de pacientes.

Valores retornados

-	0 se o programa correu sem erros (SUCESSO).
-	6 se houve um erro na alocação de memória (ERRO_ALOCAR_MEMORIA).
-	20 se houve erro na leitura da data (ERRO_LEITURA_DATA).

Valores retornados

-	21 se houve erro na leitura do limite de calorias (ERRO_LEITURA_LIMITE_CALORIAS).

5.16 Referência ao ficheiro Menu.h

Ficheiro de cabeçalho para funcionalidades do menu.

Funções

- void ApresentarMenu ()

  Apresenta o menu de opções para interação com o programa.
- bool LerData (char *data, const char *proposito)

  Lê uma string do utilizador para a data.
- int PacientesUltrapassaramCalorias (Dieta *dietas, int numDietas, int numPacientes)

  Função para executar a funcionalidade 2 - Pacientes que ultrapassam um dado limite de calorias entre determinadas datas.
- int PacientesForaPlano (Paciente *pacientes, Dieta *dietas, Plano *planos)

  Método que executa a funcionalidade 3 - apresentar pacientes com dietas fora do seu plano.
- int ListarPlanoNutricional (Plano planos[ ], int numPlanos)

  Função que executa a funcionalidade 4 - Listar o plano nutricional de um paciente para determinada refeição ao longo de um determinado período.
- int ListarMediasCaloriasPorRefeicao (Paciente pacientes[ ], Dieta dietas[ ])

  Lista médias de calorias por refeição para um intervalo de datas específico.
- int GerarTabelaRefeicoes (Paciente pacientes[ ], int numPacientes, Plano planos[ ], int numPlanos, Dieta dietas[ ], int numDietas)

  Gera a tabela de refeições.

5.16.1 Descrição detalhada

Ficheiro de cabeçalho para funcionalidades do menu.
Este ficheiro contém declarações para funcionalidades relacionadas ao menu do programa.

Autor

Enrique George Rodrigues

Data

Criado: 18.11.2023

Modificado: 28.12.2023

Copyright

5.16.2 Documentação das funções

5.16.2.1 ApresentarMenu()

```
void ApresentarMenu ( )
```

Apresenta o menu de opções para interação com o programa.
Esta função imprime no ecrã as opções disponíveis para interação com o programa. O utilizador é solicitado a escolher uma das opções apresentadas.

Nota

As opções incluem carregar dados, verificar pacientes com calorias ultrapassadas, verificar pacientes com calorias fora do intervalo, obter o plano nutricional de um paciente para uma refeição específica, calcular médias de calorias consumidas por refeição por cada paciente, e apresentar uma tabela das refeições planeadas e realizadas para todos os pacientes. As opções 'q' e 'f' fecham o programa.

Aviso

A função não lida com a leitura da escolha do utilizador e não verifica a validade da escolha inserida. Apenas apresenta o menu no stdout.

5.16.2.2 GerarTabelaRefeicoes()

```
int GerarTabelaRefeicoes (
            Paciente pacientes[],
            int numPacientes,
            Plano planos[],
            int numPlanos,
            Dieta dietas[],
            int numDietas )
```

Gera a tabela de refeições.

Parâmetros

pacientes	- Array de pacientes.
numPacientes	- Número de pacientes.
planos	- Array de planos.
numPlanos	- Número de planos.
dietas	- Array de dietas.
numDietas	- Número de dietas.

Valores retornados

-	0 se correr com sucesso.
-	26 se não foi possível encontrar um paciente.

5.16.2.3 LerData()

```
bool LerData (
            char * data,
            const char * proposito )
```

Lê uma string do utilizador para a data.

Parâmetros

data	- A variável onde se quer guardar a data.
proposito	- O objetivo da leitura (p. ex., "inicial" ou "final").

Valores retornados

-	Verdadeiro se correu sem erros, Falso se correu com erros.

5.16.2.4 ListarMediasCaloriasPorRefeicao()

```
int ListarMediasCaloriasPorRefeicao (
            Paciente pacientes[],
            Dieta dietas[] )
```

Lista médias de calorias por refeição para um intervalo de datas específico.

Parâmetros

pacientes	- O array de pacientes.
dietas	- O array de todas as dietas.

Valores retornados

-	0 se correr com sucesso.
-	20 se houver um erro com a leitura de datas.

Parâmetros

pacientes	- O array de pacientes.
dietas	- O array de todas as dietas.
resultados	- O array de resultados.
numResultados	- O número de resultados.

Valores retornados

-	0 se correr com sucesso.
-	20 se houver um erro com a leitura de datas.

5.16.2.5 ListarPlanoNutricional()

```
int ListarPlanoNutricional (
            Plano planos[],
            int numPlanos )
```

Função que executa a funcionalidade 4 - Listar o plano nutricional de um paciente para determinada refeição ao longo de um determinado período.

Parâmetros

planos	- O array de planos.
numPlanos	- O número de planos.

Valores retornados

-	0 se correr com sucesso.
-	20 se houver um erro com a leitura de datas (ERRO_LEITURA_DATA).
-	22 se não for possível ler o ID do paciente (ERRO_LEITURA_ID_PACIENTE).
-	23 se não for possível ler o tipo de refeição (ERRO_LEITURA_TIPO_REFEICAO).

Função que executa a funcionalidade 4 - Listar o plano nutricional de um paciente para determinada refeição ao longo de um determinado período.

Parâmetros

planos	- O array de planos nutricionais.
numPlanos	- O número de planos nutricionais no array.

Valores retornados

-	0 se correr com sucesso.
-	20 se houver um erro com a leitura de datas (ERRO_LEITURA_DATA).
-	22 se não for possível ler o ID do paciente (ERRO_LEITURA_ID_PACIENTE).
-	23 se não for possível ler o tipo de refeição (ERRO_LEITURA_TIPO_REFEICAO).

5.16.2.6 PacientesForaPlano()

```
int PacientesForaPlano (
            Paciente * pacientes,
            Dieta * dietas,
            Plano * planos )
```

Método que executa a funcionalidade 3 - apresentar pacientes com dietas fora do seu plano.

Parâmetros

pacientes	- O array de pacientes.
dietas	- O array de dietas.
planos	- O array de planos.

Valores retornados

-	0 se correr com sucesso.
-	20 se houver um erro com a leitura de datas.

Método que executa a funcionalidade 3 - apresentar pacientes com dietas fora do seu plano.

Parâmetros

in	*pacientes*	- O array de pacientes.
in	*dietas*	- O array de dietas.
in	*planos*	- O array de planos.

Valores retornados

0	se correr com sucesso.
20	se houver um erro com a leitura de datas (ERRO_LEITURA_DATA).

5.16.2.7 PacientesUltrapassaramCalorias()

```
int PacientesUltrapassaramCalorias (
            Dieta * dietas,
            int numDietas,
            int numPacientes )
```

Função para executar a funcionalidade 2 - Pacientes que ultrapassam um dado limite de calorias entre determinadas datas.

Parâmetros

dietas	- O array de dietas.
numDietas	- O número de dietas.
numPacientes	- O número de pacientes.

Valores retornados

-	0 se o programa correu sem erros (SUCESSO).
-	6 se houve um erro na alocação de memória (ERRO_ALOCAR_MEMORIA).
-	20 se houve erro na leitura da data (ERRO_LEITURA_DATA).
-	21 se houve erro na leitura do limite de calorias (ERRO_LEITURA_LIMITE_CALORIAS).

5.17 Menu.h

Ir para a documentação deste ficheiro.

```
00001
00018 #ifndef MENU_H
00019 #define MENU_H
00020
00036 void ApresentarMenu();
00037
00044 bool LerData(char* data, const char* proposito);
00045
00056 int PacientesUltrapassaramCalorias(Dieta* dietas, int numDietas, int numPacientes);
00057
00066 int PacientesForaPlano(Paciente* pacientes, Dieta* dietas, Plano* planos);
00067
00077 int ListarPlanoNutricional(Plano planos[], int numPlanos);
00078
00086 int ListarMediasCaloriasPorRefeicao(Paciente pacientes[], Dieta dietas[]);
00087
00099 int GerarTabelaRefeicoes(Paciente pacientes[], int numPacientes, Plano planos[], int numPlanos, Dieta
      dietas[], int numDietas);
00100
00101 #endif
```

5.18 Referência ao ficheiro OperacoesFicheiros.c

Implementação das funções relacionadas às operações em ficheiros.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include "../GestorDadosIO/IODadosAPI.h"
#include "Constantes.h"
#include "Utils.h"
```

Funções

- bool ExisteFicheiroBinario (const char *nomeFicheiro)

  Verifica se existe um dado ficheiro binário através do nome do ficheiro.
- int VerificarEAtualizarFicheiroBinario (const char *nomeFicheiro)

  Função que verifica se exite um ficheiro binário. Se não existir, pergunta ao utilizador se quer criá-lo.
- void ApagarBaseDados ()

  Lida com o pedido de apagar a base de dados. Depois de terminar, seja com sucesso ou não, o programa termina.

- void ImportarDadosBaseDados (const char ∗caminhoFicheiro, void ∗dados, size_t tamDados, int ∗num↩ Dados, const char ∗tipoDados)

  Importa os dados de um ficheiro específico da base de dados (ficheiro .dat).

5.18.1 Descrição detalhada

Implementação das funções relacionadas às operações em ficheiros.
Este ficheiro contém implementações das funções para verificar e controlar ficheiros.

Autor

Enrique George Rodrigues

Data

Criado: 28.11.2023

Modificado: 28.12.2023

Copyright

5.18.2 Documentação das funções

5.18.2.1 ApagarBaseDados()

```
void ApagarBaseDados ( )
```

Lida com o pedido de apagar a base de dados. Depois de terminar, seja com sucesso ou não, o programa termina.
Lida com o pedido de apagar a base de dados.

5.18.2.2 ExisteFicheiroBinario()

```
bool ExisteFicheiroBinario (
            const char * nomeFicheiro )
```

Verifica se existe um dado ficheiro binário através do nome do ficheiro.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário

Valores retornados

-	true (ficheiro existe)
-	false (ficheiro não existe)

5.18.2.3 ImportarDadosBaseDados()

```
void ImportarDadosBaseDados (
            const char * caminhoFicheiro,
            void * dados,
            size_t tamDados,
            int * numDados,
            const char * tipoDados )
```

Importa os dados de um ficheiro específico da base de dados (ficheiro .dat).

Parâmetros

Parâmetros

caminhoFicheiro	- Caminho do arquivo binário a ser importado.
dados	- Array de estruturas para guardar os dados importados.
tamDados	- Tamanho de cada elemento do array de dados.
numDados	- O tamanho do array.
tipoDados	- Nome do tipo de dados sendo importado (para a mensagem de log).

5.18.2.4 VerificarEAtualizarFicheiroBinario()

```
int VerificarEAtualizarFicheiroBinario (
            const char * nomeFicheiro )
```

Função que verifica se exite um ficheiro binário. Se não existir, pergunta ao utilizador se quer criá-lo.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.

Nota

Esta função está a ser utilizada nas verificações iniciais do programa para garantir que todos os ficheiros binários necessários existem antes de executar o programa.

Valores retornados

0	- Ficheiro criado com sucesso.
1	- Ficheiro existe (FICHEIRO_EXISTE).
2	- Erro criar ficheiro (ERRO_CRIAR_FICHEIRO).
25	- Utilizador não deu permissão para criar o ficheiro. (ERRO_UTILIZADOR_NAO_DEU_PERMISSAO).

5.19 Referência ao ficheiro OperacoesFicheiros.h

Funções para operações relacionadas a ficheiros.

```
#include <stdbool.h>
```

Funções

- bool ExisteFicheiroBinario (const char ∗nomeFicheiro)

  Verifica se existe um dado ficheiro binário através do nome do ficheiro.
- int VerificarEAtualizarFicheiroBinario (const char ∗nomeFicheiro)

  Função que verifica se exite um ficheiro binário. Se não existir, pergunta ao utilizador se quer criá-lo.
- void ApagarBaseDados ()

  Lida com o pedido de apagar a base de dados.
- void ImportarDadosBaseDados (const char ∗caminhoFicheiro, void ∗dados, size_t tamDados, int ∗num↩ Dados, const char ∗tipoDados)

  Importa os dados de um ficheiro específico da base de dados (ficheiro .dat).

5.19.1 Descrição detalhada

Funções para operações relacionadas a ficheiros.
Este ficheiro contém declarações de funções que realizam operações diversas relacionadas a ficheiros.

Autor

Enrique George Rodrigues

Data

Criado: 28.11.2023

Modificado: 28.12.2023

Copyright

5.19.2 Documentação das funções

5.19.2.1 ApagarBaseDados()

```
void ApagarBaseDados ( )
```

Lida com o pedido de apagar a base de dados.
Lida com o pedido de apagar a base de dados.

5.19.2.2 ExisteFicheiroBinario()

```
bool ExisteFicheiroBinario (
            const char * nomeFicheiro )
```

Verifica se existe um dado ficheiro binário através do nome do ficheiro.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário

Valores retornados

-	true (ficheiro existe)
-	false (ficheiro não existe)

5.19.2.3 ImportarDadosBaseDados()

```
void ImportarDadosBaseDados (
            const char * caminhoFicheiro,
            void * dados,
            size_t tamDados,
            int * numDados,
            const char * tipoDados )
```

Importa os dados de um ficheiro específico da base de dados (ficheiro .dat).

Parâmetros

caminhoFicheiro	- Caminho do arquivo binário a ser importado.
dados	- Array de estruturas para guardar os dados importados.
tamDados	- Tamanho de cada elemento do array de dados.
numDados	- O tamanho do array.

Parâmetros

tipoDados	- Nome do tipo de dados sendo importado (para a mensagem de log).

5.19.2.4 VerificarEAtualizarFicheiroBinario()

```
int VerificarEAtualizarFicheiroBinario (
            const char * nomeFicheiro )
```

Função que verifica se exite um ficheiro binário. Se não existir, pergunta ao utilizador se quer criá-lo.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.

Nota

Esta função está a ser utilizada nas verificações iniciais do programa para garantir que todos os ficheiros binários necessários existem antes de executar o programa.

Valores retornados

0	- Ficheiro criado com sucesso.
1	- Ficheiro existe (FICHEIRO_EXISTE).
2	- Erro criar ficheiro (ERRO_CRIAR_FICHEIRO).
25	- Utilizador não deu permissão para criar o ficheiro. (ERRO_UTILIZADOR_NAO_DEU_PERMISSAO).

5.20 OperacoesFicheiros.h

Ir para a documentação deste ficheiro.

```
00001
00018 #ifndef OPERACOES_FICHEIROS_H
00019 #define OPERACOES_FICHEIROS_H
00020
00021 #include <stdbool.h>
00022
00023 // TODO: move to library?
00024
00031 bool ExisteFicheiroBinario(const char* nomeFicheiro);
00032
00047 int VerificarEAtualizarFicheiroBinario(const char* nomeFicheiro);
00048
00052 void ApagarBaseDados();
00053
00063 void ImportarDadosBaseDados(const char* caminhoFicheiro, void* dados, size_t tamDados, int* numDados,
      const char* tipoDados);
00064
00065 #endif
```

5.21 Referência ao ficheiro Parametros.c

Funções que controlam e lidam com os parâmetros opcionais do programa.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Constantes.h"
#include "Utils.h"
#include "OperacoesFicheiros.h"
```

Funções

- void ManipularParametrosOpcionais (int argc, char ∗argv[ ], char ∗∗ficheiroPacientes, char ∗∗ficheiroDietas, char ∗∗ficheiroPlanos, int ∗tipoFicheiroPacientes, int ∗tipoFicheiroDietas, int ∗tipoFicheiroPlanos)

  Manipula os parâmetros opcionais.

5.21.1 Descrição detalhada

Funções que controlam e lidam com os parâmetros opcionais do programa.

Autor

Enrique George Rodrigues

Data

Criado: 27.12.2023

Modificado: 28.12.2023

Copyright

5.21.2 Documentação das funções

5.21.2.1 ManipularParametrosOpcionais()

```
void ManipularParametrosOpcionais (
            int argc,
            char * argv[],
            char ** ficheiroPacientes,
            char ** ficheiroDietas,
            char ** ficheiroPlanos,
            int * tipoFicheiroPacientes,
            int * tipoFicheiroDietas,
            int * tipoFicheiroPlanos )
```

Manipula os parâmetros opcionais.

Parâmetros

argc	- Número de argumentos da linha de comandos.
argv	- Vetor de argumentos da linha de comandos.
ficheiroPacientes	- Apontador para o nome do ficheiro de pacientes.
ficheiroDietas	- Apontador para o nome do ficheiro de dietas.
ficheiroPlanos	- Apontador para o nome do ficheiro de planos.
tipoFicheiroPacientes	- Apontador para o tipo de ficheiro de pacientes.
tipoFicheiroDietas	- Apontador para o tipo de ficheiro de dietas.
tipoFicheiroPlanos	- Apontador para o tipo de ficheiro de planos.

5.22 Referência ao ficheiro Parametros.h

Funções que controlam e lidam com os parâmetros opcionais do programa.

Funções

- void ManipularParametrosOpcionais (int argc, char ∗argv[ ], char ∗∗ficheiroPacientes, char ∗∗ficheiroDietas, char ∗∗ficheiroPlanos, int ∗tipoFicheiroPacientes, int ∗tipoFicheiroDietas, int ∗tipoFicheiroPlanos)

  Manipula os parâmetros opcionais.

5.22.1 Descrição detalhada

Funções que controlam e lidam com os parâmetros opcionais do programa.
∼

Autor

Enrique George Rodrigues

Data

Criado: 27.12.2023

Modificado: 28.12.2023

Copyright

5.22.2 Documentação das funções

5.22.2.1 ManipularParametrosOpcionais()

```
void ManipularParametrosOpcionais (
            int argc,
            char * argv[],
            char ** ficheiroPacientes,
            char ** ficheiroDietas,
            char ** ficheiroPlanos,
            int * tipoFicheiroPacientes,
            int * tipoFicheiroDietas,
            int * tipoFicheiroPlanos )
```

Manipula os parâmetros opcionais.

Parâmetros

argc	- Número de argumentos da linha de comandos.
argv	- Vetor de argumentos da linha de comandos.
ficheiroPacientes	- Apontador para o nome do ficheiro de pacientes.
ficheiroDietas	- Apontador para o nome do ficheiro de dietas.
ficheiroPlanos	- Apontador para o nome do ficheiro de planos.
tipoFicheiroPacientes	- Apontador para o tipo de ficheiro de pacientes.
tipoFicheiroDietas	- Apontador para o tipo de ficheiro de dietas.
tipoFicheiroPlanos	- Apontador para o tipo de ficheiro de planos.

5.23 Parametros.h

Ir para a documentação deste ficheiro.

```
00001
00017 #ifndef PARAMETROS_H
00018 #define PARAMETROS_H
00019
```

```
00031 void ManipularParametrosOpcionais(int argc, char* argv[], char** ficheiroPacientes, char**
      ficheiroDietas, char** ficheiroPlanos,
00032     int* tipoFicheiroPacientes, int* tipoFicheiroDietas, int* tipoFicheiroPlanos);
00033
00034 #endif
```

5.24 Referência ao ficheiro Utils.c

Funções utilitárias do programa.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <stdarg.h>
#include <time.h>
#include <string.h>
#include <unistd.h>
#include "Constantes.h"
#include "EstruturaDados.h"
#include "ControloErros.h"
#include "../GestorDadosIO/IODadosAPI.h"
```

Funções

- void CustomSleep (int segundos)

  Uma função que para o programa durante n segundos.
- int LerTecla ()

  Uma função que permite ler um carater.
- void LimparEcra ()

  Limpa o ecrã do terminal.
- void EscreverComCor (const char *cor, const char *formato,...)

  Uma função que permite escrever no terminal numa cor definida por códigos ANSI.
- void ApresentarAjuda ()

  Apresenta o manual de ajuda ao utilizador.
- void **ApresentarErros** ()

  Apresenta os possíveis erros na execução do programa.
- void TerminarProgramaComErro (int codigoErro)

  Termina o programa apresentando uma mensagem de erro e terminando com um código de erro.
- void **PerguntarUtilizadorIniciarPrograma** ()

  Pergunta ao utilizador se deseja iniciar o programa ou sair.
- time_t ConverterStringParaDatetime (const char *dateString)

  Converte uma data no formato texto `DD-MM-AAAA` para um objeto datetime.
- const char * ConverterDatetimeParaString (time_t datetime)

  Converte um objeto datetime para uma data no formato texto `DD-MM-AAAA`.
- void LibertarMemoria (Paciente **pacientes, int *numPacientes, Dieta **dietas, int *numDietas, Plano **planos, int *numPlanos)

  Liberta toda a memoria alocada aos pacientes, dietas e planos.
- int CompararPacientePorId (const void *a, const void *b)

  Compara dois elementos do tipo Paciente por ID.
- void OrganizarPacientesPorId (Paciente pacientes[ ], int tamanho)

  Organiza um array de Paciente por ID usando o algoritmo de ordenação qsort.
- int CompararDietaPorId (const void *a, const void *b)

  Compara dois elementos do tipo Dieta por ID do paciente associado.
- int CompararDietaPorData (const void *a, const void *b)

Compara dois elementos do tipo Dieta por data em ordem decrescente.

- void OrganizarDietasPorId (Dieta dietas[ ], int tamanho)

  Organiza um array de Dieta por ID do paciente associado usando o algoritmo de ordenação qsort.
- int CompararPlanoPorId (const void ∗a, const void ∗b)

  Compara dois elementos do tipo Plano por ID do paciente associado.
- void OrganizarPlanosPorId (Plano planos[ ], int tamanho)

  Organiza um array de Plano por ID do paciente associado usando o algoritmo de ordenação qsort.
- bool ValidarFormatoData (const char ∗dataStr)

  Verifica se uma string está no formato de data DD-MM-AAAA.
- int ProcessarFicheiro (const char ∗nomeFicheiro, int tipoDadosFicheiro, int tipoFicheiro, void ∗dados, size_t tamanhoRegisto, int ∗numRegistos)

  Processa um ficheiro, importando dados de acordo com o tipo de ficheiro e tipo de dados especificados.
- int ImportarEProcessarFicheiro (const char ∗ficheiro, int tipoDados, int tipoFicheiro, void ∗dados, size_t tamanhoRegisto, int ∗numRegistos)

  Importa e processa um ficheiro, apresentando informações e lidando com o resultado.
- int ConverterRefeicaoParaId (char refeicao[TAMANHO_REFEICAO])

  Converte uma string de refeição para o ID (PEQUENO_ALMOCO_ID, ALMOCO_ID, JANTAR_ID).
- const char ∗ ConverterRefeicaoParaTexto (int refeicao)

  Converte uma string de refeição para o ID (PEQUENO_ALMOCO_ID, ALMOCO_ID, JANTAR_ID).
- int GuardarDados (Paciente pacientes[ ], int numPacientes, Dieta dietas[ ], int numDietas, Plano planos[ ], int numPlanos)

  Guarda os dados para os ficheiros binários.

5.24.1 Descrição detalhada

Funções utilitárias do programa.

Autor

Enrique George Rodrigues

Data

Criado: 18.11.2023

Modificado: 28.12.2023

Copyright

5.24.2 Documentação das funções

5.24.2.1 ApresentarAjuda()

```
void ApresentarAjuda ( )
```

Apresenta o manual de ajuda ao utilizador.
Esta função demonstra como utilizar o programa.

5.24.2.2 CompararDietaPorData()

```
int CompararDietaPorData (
            const void * a,
            const void * b )
```

Compara dois elementos do tipo Dieta por data em ordem decrescente.

Parâmetros

a	- Um apontador para o primeiro elemento Dieta.
b	- Um apontador para o segundo elemento Dieta.

Valores retornados

Um	número negativo se a data de 'a' é mais recente que a data de 'b', um número positivo se a data de 'a' é mais antiga que a data de 'b', e 0 se as datas são iguais.

5.24.2.3 CompararDietaPorId()

```
int CompararDietaPorId (
            const void * a,
            const void * b )
```

Compara dois elementos do tipo Dieta por ID do paciente associado.

Parâmetros

a	- Apontador para o primeiro elemento Dieta.
b	- Apontador para o segundo elemento Dieta.

Retorna

Um número negativo se o ID do paciente associado de 'a' é menor que o ID do paciente associado de 'b', um número positivo se o ID do paciente associado de 'a' é maior que o ID do paciente associado de 'b' e 0 se os IDs do paciente associado são iguais.

5.24.2.4 CompararPacientePorId()

```
int CompararPacientePorId (
            const void * a,
            const void * b )
```

Compara dois elementos do tipo Paciente por ID.

Parâmetros

a	- Apontador para o primeiro elemento Paciente.
b	- Apontador para o segundo elemento Paciente.

Retorna

Um número negativo se o ID de 'a' é menor que o ID de 'b', um número positivo se o ID de 'a' é maior que o ID de 'b' e 0 se os IDs são iguais.

5.24.2.5 CompararPlanoPorId()

```
int CompararPlanoPorId (
            const void * a,
            const void * b )
```

Compara dois elementos do tipo Plano por ID do paciente associado.

Parâmetros

a	- Apontador para o primeiro elemento Plano.
b	- Apontador para o segundo elemento Plano.

Retorna

Um número negativo se o ID do paciente associado de 'a' é menor que o ID do paciente associado de 'b', um número positivo se o ID do paciente associado de 'a' é maior que o ID do paciente associado de 'b' e 0 se os IDs do paciente associado são iguais.

5.24.2.6 ConverterDatetimeParaString()

```
const char * ConverterDatetimeParaString (
            time_t datetime )
```

Converte um objeto datetime para uma data no formato texto `DD-MM-AAAA`.

Parâmetros

datetime	- O objeto datetime.

Valores retornados

-	A string formatada "DD-MM-AAAA".

5.24.2.7 ConverterRefeicaoParaId()

```
int ConverterRefeicaoParaId (
            char refeicao[TAMANHO_REFEICAO] )
```

Converte uma string de refeição para o ID (PEQUENO_ALMOCO_ID, ALMOCO_ID, JANTAR_ID).

Parâmetros

refeicao	- A string da refeição.

Valores retornados

-	O ID da refeição ou ERRO_REFEICAO_ID (-1) em caso de erro.

5.24.2.8 ConverterRefeicaoParaTexto()

```
const char * ConverterRefeicaoParaTexto (
            int refeicao )
```

Converte uma string de refeição para o ID (PEQUENO_ALMOCO_ID, ALMOCO_ID, JANTAR_ID).

Parâmetros

refeicao	- O ID da refeição.

Valores retornados

-	A string da refeição.

5.24.2.9 ConverterStringParaDatetime()

```
time_t ConverterStringParaDatetime (
            const char * dateString )
```

Converte uma data no formato texto `DD-MM-AAAA` para um objeto datetime.

Parâmetros

dateString	- A data em formato texto.

Valores retornados

-	O objeto datetime.
-	14 - Erro na conversão (ERRO_CONVERSAO_TEXTO_DATETIME).

5.24.2.10 CustomSleep()

```
void CustomSleep (
            int segundos )
```

Uma função que para o programa durante n segundos.

Parâmetros

segundos	- O número de segundos que o programa deve parar.

Nota

- A função usa o "Sleep()" para Windows e "sleep()" para sistemas Unix.

5.24.2.11 EscreverComCor()

```
void EscreverComCor (
            const char * cor,
            const char * formato,
             ...  )
```

Uma função que permite escrever no terminal numa cor definida por códigos ANSI.

Parâmetros

cor	- O código ANSI ou modo pré-definido (ex. ANSI_INFO ou ANSI_ERROR).
formato	- O texto a ser apresentado no terminal com suporte para parâmetros (ex. d).
...	- Os parâmetros opcionais para apresentar valores de variáveis (ex. d).

Nota

- Esta função é uma função variádica que suporta `n` parâmetros adicionais para escrever valores de variáveis.

Exemplo de uso: `EscreverComCor(ANSI_INFO, "Isto e uma mensagem de info! d, &variavelTipoInt);`

5.24.2.12 GuardarDados()

```
int GuardarDados (
            Paciente pacientes[],
            int numPacientes,
```

```
            Dieta dietas[],
            int numDietas,
            Plano planos[],
            int numPlanos )
```

Guarda os dados para os ficheiros binários.

Parâmetros

pacientes	Array de pacientes a serem guardados.
numPacientes	Número de elementos no array de pacientes.
dietas	Array de dietas a serem guardadas.
numDietas	Número de elementos no array de dietas.
planos	Array de planos a serem guardados.
numPlanos	Número de elementos no array de planos.

5.24.2.13 ImportarEProcessarFicheiro()

```
int ImportarEProcessarFicheiro (
            const char * ficheiro,
            int tipoDados,
            int tipoFicheiro,
            void * dados,
            size_t tamanhoRegisto,
            int * numRegistos )
```

Importa e processa um ficheiro, apresentando informações e lidando com o resultado.

Parâmetros

ficheiro	- O nome do ficheiro a ser importado.
tipoDados	- O tipo de dados a ser importado (TIPO_DADOS_PACIENTES, TIPO_DADOS_DIETAS, TIPO_DADOS_PLANOS).
tipoFicheiro	- O tipo de ficheiro a ser importado (TIPO_FICHEIRO_TXT, TIPO_FICHEIRO_BIN, TIPO_FICHEIRO_TAB).
dados	- Um apontador para a estrutura de dados onde os dados serão guardados.
tamanhoRegisto	- O tamanho de cada registo dos dados.
numRegistos	- Um apontador para a variável que vai guardar o número de registos importados.

Nota

Esta função é utilizada para simplificar o processo de importação, apresentação de informações e manipulação do resultado.

Valores retornados

-	0 se o programa correu sem erros (SUCESSO).
-	24 se o ficheiro introduzido é NULLO (ERRO_FICHEIRO_NULLO).

5.24.2.14 LerTecla()

```
int LerTecla ( )
```

Uma função que permite ler um carater.

Nota

- A função usa o "_getch()" para Windows e um processo especial para sistemas Unix.

5.24.2.15 LibertarMemoria()

```
void LibertarMemoria (
            Paciente ** pacientes,
            int * numPacientes,
            Dieta ** dietas,
            int * numDietas,
            Plano ** planos,
            int * numPlanos )
```

Liberta toda a memoria alocada aos pacientes, dietas e planos.

Parâmetros

pacientes	- Array pacientes.
numPacientes	- Apontador para o número de pacientes.
dietas	- Array dietas.
numDietas	- Apontador para o número de dietas.
planos	- Array planos.
numPlanos	- Apontador para o número de planos.

Nota

- Esta função não está a ser utilizada neste momento devido à migração para arrays de tamanho constante. Mas numa implentação futura, pode-se alocar a memória dinamicamente, colocando os dados no heap em vez do stack.

5.24.2.16 LimparEcra()

```
void LimparEcra ( )
```

Limpa o ecrã do terminal.
Esta função utiliza a função 'system()' para chamar o comando adequado para limpar o ecrã do terminal. Suporta sistemas Windows e Unix.

Nota

A função suporta sistemas Windows (utiliza "cls") e sistemas Unix (utiliza "clear").

5.24.2.17 OrganizarDietasPorId()

```
void OrganizarDietasPorId (
            Dieta dietas[],
            int tamanho )
```

Organiza um array de Dieta por ID do paciente associado usando o algoritmo de ordenação qsort.

Parâmetros

dietas	- Array de Dieta.
tamanho	- Número de elementos no array.

5.24.2.18 OrganizarPacientesPorId()

```
void OrganizarPacientesPorId (
```

```
            Paciente pacientes[],
            int tamanho )
```

Organiza um array de Paciente por ID usando o algoritmo de ordenação qsort.

Parâmetros

pacientes	- Array de Paciente.
tamanho	- Número de elementos no array.

5.24.2.19 OrganizarPlanosPorId()

```
void OrganizarPlanosPorId (
            Plano planos[],
            int tamanho )
```

Organiza um array de Plano por ID do paciente associado usando o algoritmo de ordenação qsort.

Parâmetros

planos	- Array de Plano.
tamanho	- Número de elementos no array.

5.24.2.20 ProcessarFicheiro()

```
int ProcessarFicheiro (
            const char * nomeFicheiro,
            int tipoDadosFicheiro,
            int tipoFicheiro,
            void * dados,
            size_t tamanhoRegisto,
            int * numRegistos )
```

Processa um ficheiro, importando dados de acordo com o tipo de ficheiro e tipo de dados especificados.

Parâmetros

nomeFicheiro	- O nome do ficheiro a ser processado.
tipoDadosFicheiro	- O tipo de dados a ser importado (TIPO_DADOS_PACIENTES, TIPO_DADOS_DIETAS, TIPO_DADOS_PLANOS).
tipoFicheiro	- O tipo de ficheiro a ser processado (TIPO_FICHEIRO_TXT, TIPO_FICHEIRO_BIN, TIPO_FICHEIRO_TAB).
dados	- Um apontador para a estrutura de dados onde os dados serão guardados.
tamanhoRegisto	- O tamanho de cada registo dos dados.
numRegistos	- Um apontador para a variável que vai guardar o número de registos importados.

Nota

Esta função é responsável por chamar funções específicas de importação de acordo com os tipos especificados.

Valores retornados

resultado	- O código de resultado da operação.

5.24.2.21 TerminarProgramaComErro()

```
void TerminarProgramaComErro (
            int codigoErro )
```

Termina o programa apresentando uma mensagem de erro e terminando com um código de erro.
Esta função imprime uma mensagem de erro no standard error (stderr) seguida pelo código de erro especificado. A seguir, termina o programa utilizando 'exit()' com o código de erro "codigoErro".

Parâmetros

codigoErro	- O código de erro a ser apresentado e utilizado para encerrar o programa.

5.24.2.22 ValidarFormatoData()

```
bool ValidarFormatoData (
            const char * dataStr )
```

Verifica se uma string está no formato de data DD-MM-AAAA.

Parâmetros

dataStr	- String de data para verificação.

Retorna

true se estiver no formato correto, false caso contrário.

5.25 Referência ao ficheiro Utils.h

Ficheiro de cabeçalho para utilitários do programa.

```
#include <stdbool.h>
#include "EstruturaDados.h"
```

Funções

- void CustomSleep (int segundos)

  Uma função que para o programa durante n segundos.
- int LerTecla ()

  Uma função que permite ler um carater.
- void LimparEcra ()

  Limpa o ecrã do terminal.
- void TerminarProgramaComErro (int erro)

  Termina o programa apresentando uma mensagem de erro e terminando com um código de erro.
- void EscreverComCor (const char ∗cor, const char ∗formato,...)

  Uma função que permite escrever no terminal numa cor definida por códigos ANSI.
- void ApresentarAjuda ()

  Apresenta o manual de ajuda ao utilizador.
- void **ApresentarErros** ()

  Apresenta os possíveis erros na execução do programa.
- void **PerguntarUtilizadorIniciarPrograma** ()

  Pergunta ao utilizador se deseja iniciar o programa ou sair.
- time_t ConverterStringParaDatetime (const char ∗dateString)

  Converte uma data no formato texto `DD-MM-AAAA` para um objeto datetime.
- const char ∗ ConverterDatetimeParaString (time_t datetime)

Converte um objeto datetime para uma data no formato texto DD-MM-AAAA.

- void LibertarMemoria (Paciente **pacientes, int *numPacientes, Dieta **dietas, int *numDietas, Plano **planos, int *numPlanos)

  Liberta toda a memoria alocada aos pacientes, dietas e planos.
- int CompararPacientePorId (const void *a, const void *b)

  Compara dois elementos do tipo Paciente por ID.
- void OrganizarPacientesPorId (Paciente pacientes[ ], int tamanho)

  Organiza um array de Paciente por ID usando o algoritmo de ordenação qsort.
- int CompararDietaPorId (const void *a, const void *b)

  Compara dois elementos do tipo Dieta por ID do paciente associado.
- int CompararDietaPorData (const void *a, const void *b)

  Compara dois elementos do tipo Dieta por data em ordem decrescente.
- void OrganizarDietasPorId (Dieta dietas[ ], int tamanho)

  Organiza um array de Dieta por ID do paciente associado usando o algoritmo de ordenação qsort.
- int CompararPlanoPorId (const void *a, const void *b)

  Compara dois elementos do tipo Plano por ID do paciente associado.
- void OrganizarPlanosPorId (Plano planos[ ], int tamanho)

  Organiza um array de Plano por ID do paciente associado usando o algoritmo de ordenação qsort.
- bool ValidarFormatoData (const char *dataStr)

  Verifica se uma string está no formato de data DD-MM-AAAA.
- int ProcessarFicheiro (const char *nomeFicheiro, int tipoDadosFicheiro, int tipoFicheiro, void *dados, size_t tamanhoRegisto, int *numRegistos)

  Processa um ficheiro, importando dados de acordo com o tipo de ficheiro e tipo de dados especificados.
- int ImportarEProcessarFicheiro (const char *ficheiro, int tipoDados, int tipoFicheiro, void *dados, size_t tamanhoRegisto, int *numRegistos)

  Importa e processa um ficheiro, apresentando informações e lidando com o resultado.
- int ConverterRefeicaoParaId (char refeicao[TAMANHO_REFEICAO])

  Converte uma string de refeição para o ID (PEQUENO_ALMOCO_ID, ALMOCO_ID, JANTAR_ID).
- const char * ConverterRefeicaoParaTexto (int refeicao)

  Converte uma string de refeição para o ID (PEQUENO_ALMOCO_ID, ALMOCO_ID, JANTAR_ID).
- int GuardarDados (Paciente pacientes[ ], int numPacientes, Dieta dietas[ ], int numDietas, Plano planos[ ], int numPlanos)

  Guarda os dados para os ficheiros binários.

5.25.1 Descrição detalhada

Ficheiro de cabeçalho para utilitários do programa.
Este ficheiro contém declarações para funções utilitárias do programa.

Autor

Enrique George Rodrigues

Data

Criado: 18.11.2023

Modificado: 28.12.2023

Copyright

5.25.2 Documentação das funções

5.25.2.1 ApresentarAjuda()

```
void ApresentarAjuda ( )
```

Apresenta o manual de ajuda ao utilizador.
Esta função demonstra como utilizar o programa.

5.25.2.2 CompararDietaPorData()

```
int CompararDietaPorData (
            const void * a,
            const void * b )
```

Compara dois elementos do tipo Dieta por data em ordem decrescente.

Parâmetros

a	- Um apontador para o primeiro elemento Dieta.
b	- Um apontador para o segundo elemento Dieta.

Valores retornados

Um	número negativo se a data de 'a' é mais recente que a data de 'b', um número positivo se a data de 'a' é mais antiga que a data de 'b', e 0 se as datas são iguais.

5.25.2.3 CompararDietaPorId()

```
int CompararDietaPorId (
            const void * a,
            const void * b )
```

Compara dois elementos do tipo Dieta por ID do paciente associado.

Parâmetros

a	- Apontador para o primeiro elemento Dieta.
b	- Apontador para o segundo elemento Dieta.

Retorna

Um número negativo se o ID do paciente associado de 'a' é menor que o ID do paciente associado de 'b', um número positivo se o ID do paciente associado de 'a' é maior que o ID do paciente associado de 'b' e 0 se os IDs do paciente associado são iguais.

5.25.2.4 CompararPacientePorId()

```
int CompararPacientePorId (
            const void * a,
            const void * b )
```

Compara dois elementos do tipo Paciente por ID.

Parâmetros

a	- Apontador para o primeiro elemento Paciente.
b	- Apontador para o segundo elemento Paciente.

Retorna

Um número negativo se o ID de 'a' é menor que o ID de 'b', um número positivo se o ID de 'a' é maior que o ID de 'b' e 0 se os IDs são iguais.

5.25.2.5 CompararPlanoPorId()

```
int CompararPlanoPorId (
            const void * a,
            const void * b )
```

Compara dois elementos do tipo Plano por ID do paciente associado.

Parâmetros

a	- Apontador para o primeiro elemento Plano.
b	- Apontador para o segundo elemento Plano.

Retorna

Um número negativo se o ID do paciente associado de 'a' é menor que o ID do paciente associado de 'b', um número positivo se o ID do paciente associado de 'a' é maior que o ID do paciente associado de 'b' e 0 se os IDs do paciente associado são iguais.

5.25.2.6 ConverterDatetimeParaString()

```
const char * ConverterDatetimeParaString (
            time_t datetime )
```

Converte um objeto datetime para uma data no formato texto `DD-MM-AAAA`.

Parâmetros

datetime	- O objeto datetime.

Valores retornados

-	A string formatada "DD-MM-AAAA".

5.25.2.7 ConverterRefeicaoParaId()

```
int ConverterRefeicaoParaId (
            char refeicao[TAMANHO_REFEICAO] )
```

Converte uma string de refeição para o ID (PEQUENO_ALMOCO_ID, ALMOCO_ID, JANTAR_ID).

Parâmetros

refeicao	- A string da refeição.

Valores retornados

-	O ID da refeição.

Parâmetros

refeicao	- A string da refeição.

Valores retornados

-	O ID da refeição ou ERRO_REFEICAO_ID (-1) em caso de erro.

5.25.2.8 ConverterRefeicaoParaTexto()

```
const char * ConverterRefeicaoParaTexto (
            int refeicao )
```

Converte uma string de refeição para o ID (PEQUENO_ALMOCO_ID, ALMOCO_ID, JANTAR_ID).

Parâmetros

refeicao	- O ID da refeição.

Valores retornados

-	A string da refeição.

5.25.2.9 ConverterStringParaDatetime()

```
time_t ConverterStringParaDatetime (
            const char * dateString )
```

Converte uma data no formato texto `DD-MM-AAAA` para um objeto datetime.

Parâmetros

dateString	- A data em formato texto.

Valores retornados

-	O objeto datetime.
-	14 - Erro na conversão (ERRO_CONVERSAO_TEXTO_DATETIME).

5.25.2.10 CustomSleep()

```
void CustomSleep (
            int segundos )
```

Uma função que para o programa durante n segundos.

Parâmetros

segundos	- O número de segundos que o programa deve parar.

Nota

- A função usa o "Sleep()" para Windows e "sleep()" para sistemas Unix.

5.25.2.11 EscreverComCor()

```
void EscreverComCor (
            const char * cor,
            const char * formato,
             ... )
```

Uma função que permite escrever no terminal numa cor definida por códigos ANSI.

Parâmetros

cor	- O código ANSI ou modo pré-definido (ex. ANSI_INFO ou ANSI_ERROR).
formato	- O texto a ser apresentado no terminal com suporte para parâmetros (ex. d).
...	- Os parâmetros opcionais para apresentar valores de variáveis (ex. d).

Nota

- Esta função é uma função variádica que suporta `n` parâmetros adicionais para escrever valores de variáveis.

Exemplo de uso: `EscreverComCor(ANSI_INFO, "Isto e uma mensagem de info! d, &variavelTipoInt);`

5.25.2.12 GuardarDados()

```
int GuardarDados (
            Paciente pacientes[],
            int numPacientes,
            Dieta dietas[],
            int numDietas,
            Plano planos[],
            int numPlanos )
```

Guarda os dados para os ficheiros binários.

Parâmetros

pacientes	Array de pacientes a serem guardados.
numPacientes	Número de elementos no array de pacientes.
dietas	Array de dietas a serem guardadas.
numDietas	Número de elementos no array de dietas.
planos	Array de planos a serem guardados.
numPlanos	Número de elementos no array de planos.

5.25.2.13 ImportarEProcessarFicheiro()

```
int ImportarEProcessarFicheiro (
            const char * ficheiro,
            int tipoDados,
            int tipoFicheiro,
            void * dados,
            size_t tamanhoRegisto,
            int * numRegistos )
```

Importa e processa um ficheiro, apresentando informações e lidando com o resultado.

Parâmetros

ficheiro	- O nome do ficheiro a ser importado.
tipoDados	- O tipo de dados a ser importado (TIPO_DADOS_PACIENTES, TIPO_DADOS_DIETAS, TIPO_DADOS_PLANOS).
tipoFicheiro	- O tipo de ficheiro a ser importado (TIPO_FICHEIRO_TXT, TIPO_FICHEIRO_BIN, TIPO_FICHEIRO_TAB).
dados	- Um apontador para a estrutura de dados onde os dados serão guardados.
tamanhoRegisto	- O tamanho de cada registo dos dados.
numRegistos	- Um apontador para a variável que vai guardar o número de registos importados.

Nota

Esta função é utilizada para simplificar o processo de importação, apresentação de informações e manipulação do resultado.

Valores retornados

-	0 se o programa correu sem erros (SUCESSO).
-	24 se o ficheiro introduzido é NULLO (ERRO_FICHEIRO_NULLO).

5.25.2.14 LerTecla()

```
int LerTecla ( )
```

Uma função que permite ler um carater.

Nota

- A função usa o "_getch()" para Windows e um processo especial para sistemas Unix.

5.25.2.15 LibertarMemoria()

```
void LibertarMemoria (
            Paciente ** pacientes,
            int * numPacientes,
            Dieta ** dietas,
            int * numDietas,
            Plano ** planos,
            int * numPlanos )
```

Liberta toda a memoria alocada aos pacientes, dietas e planos.

Parâmetros

pacientes	- Array pacientes.
numPacientes	- Pointer para o número de pacientes.
dietas	- Array dietas.
numDietas	- Pointer para o número de dietas.
planos	- Array planos.
numPlanos	- Pointer para o número de planos.
pacientes	- Array pacientes.
numPacientes	- Apontador para o número de pacientes.
dietas	- Array dietas.
numDietas	- Apontador para o número de dietas.

Parâmetros

planos	- Array planos.
numPlanos	- Apontador para o número de planos.

Nota

- Esta função não está a ser utilizada neste momento devido à migração para arrays de tamanho constante. Mas numa implentação futura, pode-se alocar a memória dinamicamente, colocando os dados no heap em vez do stack.

5.25.2.16 LimparEcra()

```
void LimparEcra ( )
```

Limpa o ecrã do terminal.
Esta função utiliza a função 'system()' para chamar o comando adequado para limpar o ecrã do terminal. Suporta sistemas Windows e Unix.

Nota

A função suporta sistemas Windows (utiliza "cls") e sistemas Unix (utiliza "clear").

5.25.2.17 OrganizarDietasPorId()

```
void OrganizarDietasPorId (
            Dieta dietas[],
            int tamanho )
```

Organiza um array de Dieta por ID do paciente associado usando o algoritmo de ordenação qsort.

Parâmetros

dietas	- Array de Dieta.
tamanho	- Número de elementos no array.

5.25.2.18 OrganizarPacientesPorId()

```
void OrganizarPacientesPorId (
            Paciente pacientes[],
            int tamanho )
```

Organiza um array de Paciente por ID usando o algoritmo de ordenação qsort.

Parâmetros

pacientes	- Array de Paciente.
tamanho	- Número de elementos no array.

5.25.2.19 OrganizarPlanosPorId()

```
void OrganizarPlanosPorId (
            Plano planos[],
            int tamanho )
```

Organiza um array de Plano por ID do paciente associado usando o algoritmo de ordenação qsort.

Parâmetros

planos	- Array de Plano.
tamanho	- Número de elementos no array.

5.25.2.20 ProcessarFicheiro()

```
int ProcessarFicheiro (
            const char * nomeFicheiro,
            int tipoDadosFicheiro,
            int tipoFicheiro,
            void * dados,
            size_t tamanhoRegisto,
            int * numRegistos )
```

Processa um ficheiro, importando dados de acordo com o tipo de ficheiro e tipo de dados especificados.

Parâmetros

nomeFicheiro	- O nome do ficheiro a ser processado.
tipoDadosFicheiro	- O tipo de dados a ser importado (TIPO_DADOS_PACIENTES, TIPO_DADOS_DIETAS, TIPO_DADOS_PLANOS).
tipoFicheiro	- O tipo de ficheiro a ser processado (TIPO_FICHEIRO_TXT, TIPO_FICHEIRO_BIN, TIPO_FICHEIRO_TAB).
dados	- Um apontador para a estrutura de dados onde os dados serão guardados.
tamanhoRegisto	- O tamanho de cada registo dos dados.
numRegistos	- Um apontador para a variável que vai guardar o número de registos importados.

Nota

Esta função é responsável por chamar funções específicas de importação de acordo com os tipos especificados.

Valores retornados

resultado	- O código de resultado da operação.

5.25.2.21 TerminarProgramaComErro()

```
void TerminarProgramaComErro (
            int codigoErro )
```

Termina o programa apresentando uma mensagem de erro e terminando com um código de erro.
Esta função imprime uma mensagem de erro no standard error (stderr) seguida pelo código de erro especificado. A seguir, termina o programa utilizando 'exit()' com o código de erro "codigoErro".

Parâmetros

codigoErro	- O código de erro a ser apresentado e utilizado para encerrar o programa.

5.25.2.22 ValidarFormatoData()

```
bool ValidarFormatoData (
            const char * dataStr )
```

Verifica se uma string está no formato de data DD-MM-AAAA.

Parâmetros

dataStr	- String de data para verificação.

Retorna

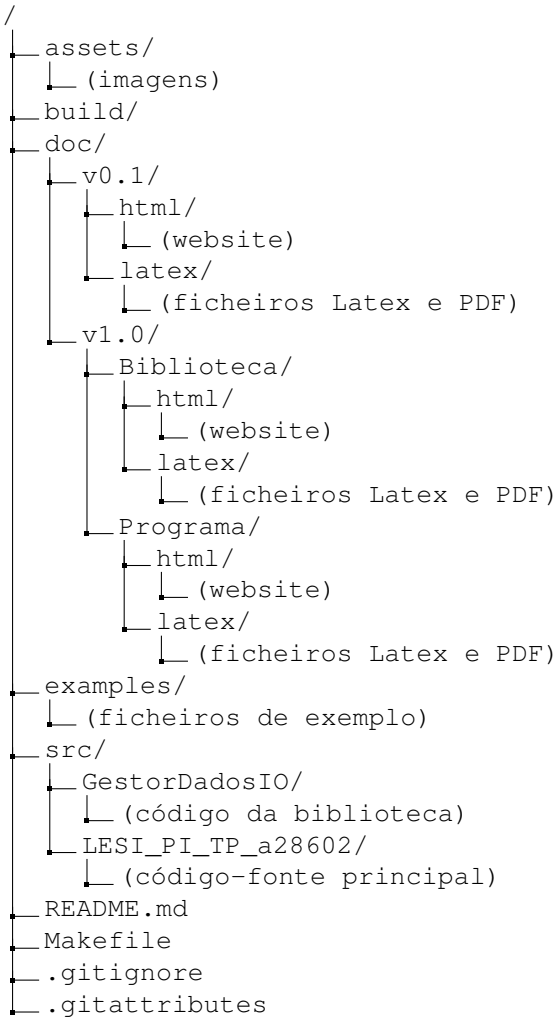
true se estiver no formato correto, false caso contrário.

5.26 Utils.h

Ir para a documentação deste ficheiro.

```
00001
00017 #ifndef UTILS_H
00018 #define UTILS_H
00019
00020 #include <stdbool.h>
00021 #include "EstruturaDados.h"
00022
00030 void CustomSleep(int segundos);
00031
00037 int LerTecla();
00038
00048 void LimparEcra();
00049
00059 void TerminarProgramaComErro(int erro);
00060
00074 void EscreverComCor(const char* cor, const char* formato, ...);
00075
00081 void ApresentarAjuda();
00082
00086 void ApresentarErros();
00087
00091 void PerguntarUtilizadorIniciarPrograma();
00092
00099 time_t ConverterStringParaDatetime(const char* dateString);
00100
00106 const char* ConverterDatetimeParaString(time_t datetime);
00107
00117 void LibertarMemoria(Paciente** pacientes, int* numPacientes, Dieta** dietas, int* numDietas, Plano**
      planos, int* numPlanos);
00118
00127 int CompararPacientePorId(const void* a, const void* b);
00128
00134 void OrganizarPacientesPorId(Paciente pacientes[], int tamanho);
00135
00144 int CompararDietaPorId(const void* a, const void* b);
00145
00155 int CompararDietaPorData(const void* a, const void* b);
00156
00162 void OrganizarDietasPorId(Dieta dietas[], int tamanho);
00163
00172 int CompararPlanoPorId(const void* a, const void* b);
00173
00179 void OrganizarPlanosPorId(Plano planos[], int tamanho);
00180
00186 bool ValidarFormatoData(const char* dataStr);
00187
00202 int ProcessarFicheiro(const char* nomeFicheiro, int tipoDadosFicheiro, int tipoFicheiro,
00203     void* dados, size_t tamanhoRegisto, int* numRegistos);
00204
00220 int ImportarEProcessarFicheiro(const char* ficheiro, int tipoDados, int tipoFicheiro,
00221     void* dados, size_t tamanhoRegisto, int* numRegistos);
00222
00228 int ConverterRefeicaoParaId(char refeicao[TAMANHO_REFEICAO]);
00229
00230
00236 const char* ConverterRefeicaoParaTexto(int refeicao);
00237
00247 int GuardarDados(Paciente pacientes[], int numPacientes, Dieta dietas[], int numDietas, Plano
      planos[], int numPlanos);
00248
00249 #endif
```

Hierarquia de Ficheiros



Conclusão

O projeto "Bem Estar"foi desenvolvido para abordar o desafio do comportamento alimentar saudável, oferecendo funcionalidades como o carregamento de dados de pacientes, dietas e planos nutricionais, apresentação de estatísticas de calorias, listagem de pacientes com comportamento fora do plano nutricional e análises diversas. Durante o desenvolvimento, a gestão eficiente do fluxo de trabalho, incluindo a criação de issues, a criação de "Milestones"/versões e o controlo de merges, foi feita por Enrique Rodrigues.
A solução foi criada com base em métodos rigorosos de análise de problemas e nas melhores práticas de programação imperativa em C. A documentação, os testes detalhados e os exemplos de testes fornecem uma visão clara da funcionalidade e robustez do programa.
O projeto "Bem Estar"atinge os seus objetivos, sendo uma solução eficaz e bem documentada para apoiar o bem-estar e cuidados nutricionais das pessoas.

Bibliografia

[1] Brian W. Kernighan and Dennis M. Ritchie. *The C Programming Language*. Prentice Hall, Englewood Cliffs, NJ, 1988.

[2] GeeksforGeeks. C programming language, 2022.

[3] Learn-C.org. Learn c - free interactive c tutorial, 2023.

[4] tutorialspoint.com. Learn c programming, 2023.

[5] OpenAI. Chatgpt, 2023.