Trabalho Prático G14

0.1

Gerado por Doxygen 1.9.8

Capítulo 1

Índice das estruturas de dados

1.1 Estruturas de dados

Lista das estruturas de dados com uma breve descrição:

Capítulo 2

Índice dos ficheiros

2.1 Lista de ficheiros

Lista de todos os ficheiros documentados com uma breve descrição:

Capítulo 3

Documentação da estruturas de dados

3.1 Referência à estrutura Dieta

Campos de Dados

- int **idPaciente**
- time_t **data**
- char **refeicao** [20]
- char **comida** [50]
- int **calorias**

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

- EstruturaDados.h

3.2 Referência à estrutura Paciente

Campos de Dados

- int **id**
- char **nome** [50]
- long **tel**

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

- EstruturaDados.h

3.3 Referência à estrutura Plano

Campos de Dados

- int **idPaciente**
- time_t **data**
- char **refeicao** [20]
- int **calorias** [2]

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

- EstruturaDados.h

Capítulo 4

Documentação do ficheiro

4.1 Referência ao ficheiro Constantes.h

Ficheiro de cabeçalho contendo definições de constantes para o programa.

Macros

- #define **MODO_DEBUG** 1
- #define **TAMANHO_MAX** 100
- #define **TAMANHO_DATA_TEXTO** 11
- #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_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 **ANSI_COR_RESET** "\x1b[0m"
- #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 **SUCESSO** 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_ALOCAR_MEMORIA** 6
- #define **ERRO_FICHEIRO_NAO_ENCONTRADO** 7
- #define **ERRO_FICHEIRO_VAZIO** 8
- #define **ERRO_LEITURA_FICHEIRO** 9
- #define **ERRO_FORMATO_INVALIDO** 10
- #define **ERRO_NAO_DEFINIDO** 11
- #define **ERRO_ABRIR_FICHEIRO** 12
- #define **ERRO_TAMANHO_EXCEDIDO** 13
- #define **ERRO_CONVERSAO_TEXTO_DATETIME** 14

4.1.1 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

Created: 18.11.2023

Last modified: 06.12.2023

Copyright

© Enrique George Rodrigues, 2023. All right reserved.

4.2 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_MAX             100 // Tamanho máximo de arrays de pacientes/dietas/planos
00026 #define TAMANHO_DATA_TEXTO      11 // Tamanho de uma data no formato ``DD-MM-AAAA" com '\0' no fim.
00027
00028 // Nomes de ficheiros binários
00029 #define FICHEIRO_BIN_PACIENTES  "pacientes.dat"
00030 #define FICHEIRO_BIN_DIETAS     "dietas_pacientes.dat"
00031 #define FICHEIRO_BIN_PLANOS     "planos_pacientes.dat"
00032
00033 // Códigos de cor ANSI
00034 #define ANSI_COR_PRETO      "\x1b[30m"
00035 #define ANSI_COR_VERMELHO   "\x1b[31m"
00036 #define ANSI_COR_VERDE      "\x1b[32m"
00037 #define ANSI_COR_AMARELO    "\x1b[33m"
00038 #define ANSI_COR_AZUL       "\x1b[34m"
00039 #define ANSI_COR_MAGENTA    "\x1b[35m"
00040 #define ANSI_COR_CYAN       "\x1b[36m"
00041 #define ANSI_COR_BRANCO     "\x1b[37m"
00042 #define ANSI_COR_RESET      "\x1b[0m"
00043
00044 // Tipos de mensagens
00045 #define ANSI_INFO           ANSI_COR_AZUL
00046 #define ANSI_ERRO           ANSI_COR_VERMELHO
00047 #define ANSI_SUCESSO        ANSI_COR_VERDE
00048 #define ANSI_ALERTA         ANSI_COR_AMARELO
00049 #define ANSI_DEBUG          ANSI_COR_MAGENTA
00050
00051 // Código de sucesso
00052 #define SUCESSO 0
00053
00054 // Códigos de erro
00055 #define ERRO_NAO_EXISTEM_FICHEIROS_BIN          1
00056 #define ERRO_CRIAR_FICHEIRO                     2
00057 #define ERRO_IMPORTAR_DADOS_TEXTO               3
00058 #define ERRO_IMPORTAR_DADOS_BIN                 4
00059 #define ERRO_APAGAR_FICHEIROS_BIN               5
00060 #define ERRO_ALOCAR_MEMORIA                     6
00061 #define ERRO_FICHEIRO_NAO_ENCONTRADO            7
00062 #define ERRO_FICHEIRO_VAZIO                     8
00063 #define ERRO_LEITURA_FICHEIRO                   9
00064 #define ERRO_FORMATO_INVALIDO                   10
00065 #define ERRO_NAO_DEFINIDO                       11
00066 #define ERRO_ABRIR_FICHEIRO                     12
00067 #define ERRO_TAMANHO_EXCEDIDO                   13
00068 #define ERRO_CONVERSAO_TEXTO_DATETIME           14
00069
00070 #endif // CONSTANTES_H
```

4.3 Referência ao ficheiro ControlarFicheiros.h

Ficheiro de cabeçalho para funções de controlo de ficheiros.

Funções

- int FicheiroExiste (const char ∗nomeFicheiro)
  Verifica se um ficheiro existe.
- int CriarFicheiro (const char ∗nomeFicheiro)
  Cria um novo ficheiro.
- int VerificarEAtualizarFicheirosPacientes ()
  Verifica a existência dos ficheiros de dados dos pacientes e os cria se não existirem.
- int VerificarEstruturaFicheiro (const char ∗nomeFicheiro, const char ∗cabecalhoEsperado)
  Verifica se a estrutura de um ficheiro corresponde ao cabeçalho esperado.
- int VerificarIntegridadeFicheirosPacientes ()
  Verifica a integridade dos dados nos ficheiros de pacientes.
- int DefinirCabecalhosFicheiro (const char ∗nomeFicheiro, const char ∗cabecalho)
  Define os cabeçalhos para um ficheiro de dados dos pacientes.
- int DefinirCabecalhosFicheirosPacientes ()
  Define os cabeçalhos para os ficheiros de dados dos pacientes.
- int EscreverDadosPacientesParaCsv (const char ∗nomeFicheiro)
  Escreve os dados de pacientes de um ficheiro de texto para o ficheiro CSV de pacientes.

4.3.1 Descrição detalhada

Ficheiro de cabeçalho para funções de controlo de ficheiros.

Este ficheiro contém declarações para funções de controlo de ficheiros.

- FicheiroExiste: Verifica a existência de um ficheiro.
- CriarFicheiro: Cria um novo ficheiro.
- ExistemFicheirosPacientes: Verifica a existência dos ficheiros dos pacientes.
- VerificarEAtualizarFicheirosPacientes: Verifica e atualiza os ficheiros dos pacientes.
- VerificarEstruturaFicheiro: Verifica a estrutura de um ficheiro.
- VerificarIntegridadeFicheirosPacientes: Verifica a integridade dos ficheiros dos pacientes.
- DefinirCabecalhosFicheiro: Define os cabeçalhos para um ficheiro.
- DefinirCabecalhosFicheirosPacientes: Define os cabeçalhos para os ficheiros dos pacientes.
- CarregarDadosPacientes: Carrega dados de pacientes de um ficheiro de texto para o ficheiro CSV de pacientes.

Autor

Enrique George Rodrigues

Data

17.11.2023

Copyright

© Enrique George Rodrigues, 2023. All right reserved.

4.3.2 Documentação das funções

4.3.2.1 CriarFicheiro()

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

Cria um novo ficheiro.

Esta função tenta abrir o ficheiro especificado por 'nomeFicheiro' em modo escrita. Se a abertura for bem-sucedida, assume-se que o ficheiro foi criado com sucesso e é fechado. Caso contrário, é considerado que ocorreu um erro ao tentar criar o ficheiro.

Parâmetros

nomeFicheiro	- O caminho/nome do ficheiro a ser criado.

Valores retornados

1	se o ficheiro foi criado com sucesso.
0	se ocorreu um erro ao tentar criar o ficheiro.

4.3.2.2 DefinirCabecalhosFicheiro()

```
int DefinirCabecalhosFicheiro (
            const char * nomeFicheiro,
            const char * cabecalho )
```

Define os cabeçalhos para um ficheiro de dados dos pacientes.

Esta função cria e define os cabeçalhos para um ficheiro CSV dos pacientes. O cabeçalho é definido de acordo com as constantes especificadas em "Constantes.h".

Parâmetros

nomeFicheiro	- O nome do ficheiro para o qual definir o cabeçalho.
cabecalho	- O cabeçalho a ser definido no ficheiro.

Valores retornados

1	se o cabeçalho foi definido com sucesso.
0	se ocorreu um erro ao abrir o ficheiro.

4.3.2.3 DefinirCabecalhosFicheirosPacientes()

```
int DefinirCabecalhosFicheirosPacientes ( )
```

Define os cabeçalhos para os ficheiros de dados dos pacientes.

Esta função cria e define os cabeçalhos para os ficheiros CSV dos pacientes: "pacientes.csv", "dietas_pacientes.↩ csv" e "planos_pacientes.csv". Os cabeçalhos são definidos de acordo com as constantes especificadas em "↩ Constantes.h".

Valores retornados

1	se os cabeçalhos foram definidos com sucesso.
0	se ocorreu um erro ao abrir algum dos ficheiros.

4.3.2.4 EscreverDadosPacientesParaCsv()

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

Escreve os dados de pacientes de um ficheiro de texto para o ficheiro CSV de pacientes.

Esta função lê dados do ficheiro de texto especificado por 'nomeFicheiro' e adiciona esses dados ao ficheiro CSV de pacientes ("pacientes.csv"). Os dados são lidos do ficheiro de texto no formato esperado e são escritos no ficheiro CSV utilizando um formato específico.

Parâmetros

nomeFicheiro	- O caminho/nome do ficheiro de texto a ser lido.

Valores retornados

1	se os dados foram carregados com sucesso para o ficheiro CSV de pacientes.
0	se ocorreu um erro ao abrir algum dos ficheiros ou ao processar os dados.

4.3.2.5 FicheiroExiste()

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

Verifica se um ficheiro existe.

Esta função tenta abrir o ficheiro especificado por 'nomeFicheiro' em modo leitura. Se a abertura for bem-sucedida, assume-se que o ficheiro existe e é fechado. Caso contrário, é considerado que o ficheiro não existe.

Parâmetros

nomeFicheiro	- O caminho/nome do ficheiro a ser verificado.

Valores retornados

1	se o ficheiro existe.
0	se o ficheiro não existe ou ocorreu um erro ao tentar abri-lo.

4.3.2.6 VerificarEAtualizarFicheirosPacientes()

```
int VerificarEAtualizarFicheirosPacientes ( )
```

Verifica a existência dos ficheiros de dados dos pacientes e os cria se não existirem.

Esta função verifica se os ficheiros de dados dos pacientes: "pacientes.csv", "dietas_pacientes.csv" e "planos_↩ pacientes.csv", existem. Caso não existam, tenta criar esses ficheiros utilizando a função CriarFicheiro(). Após a verificação ou criação dos ficheiros, a função chama DefinirCabecalhosFicheiro() para definir os cabeçalhos dos ficheiros.

Valores retornados

1	se os ficheiros existem ou foram criados com sucesso e os cabeçalhos foram definidos.
0	se ocorreu um erro ao verificar ou criar algum dos ficheiros, ou ao definir os cabeçalhos.

4.3.2.7 VerificarEstruturaFicheiro()

```
int VerificarEstruturaFicheiro (
            const char * nomeFicheiro,
            const char * cabecalhoEsperado )
```

Verifica se a estrutura de um ficheiro corresponde ao cabeçalho esperado.

Esta função tenta abrir o ficheiro especificado por 'nomeFicheiro' em modo leitura e lê a primeira linha do ficheiro. A seguir, compara o cabeçalho lido com o cabeçalho esperado especificado por 'cabecalhoEsperado'.

Parâmetros

nomeFicheiro	- O caminho/nome do ficheiro a ser verificado.
cabecalhoEsperado	- O cabeçalho esperado que deve corresponder à estrutura do ficheiro.

Valores retornados

1	se a estrutura do ficheiro corresponde ao cabeçalho esperado.
0	se ocorreu um erro ao abrir ou ler o ficheiro, ou se o cabeçalho não corresponde.

4.3.2.8 VerificarIntegridadeFicheirosPacientes()

```
int VerificarIntegridadeFicheirosPacientes ( )
```

Verifica a integridade dos dados nos ficheiros de pacientes.

Esta função verifica se os ficheiros de dados dos pacientes: "pacientes.csv", "dietas_pacientes.csv" e "planos↩ _pacientes.csv" têm a estrutura correta especificada pelos respetivos cabeçalhos definidas no ficheiro "↩ Constantes.h". A verificação é feita chamando a função VerificarEstruturaFicheiro() para cada ficheiro.

Valores retornados

1	se a integridade dos ficheiros foi verificada com sucesso.
0	se ocorreu um erro ao verificar a integridade de algum dos ficheiros.

4.4 ControlarFicheiros.h

Ir para a documentação deste ficheiro.

```
00001
00023 #ifndef CONTROLAR_FICHEIROS_H
00024 #define CONTROLAR_FICHEIROS_H
00025
00038 int FicheiroExiste(const char* nomeFicheiro);
00039
00052 int CriarFicheiro(const char* nomeFicheiro);
00053
00066 int VerificarEAtualizarFicheirosPacientes();
00067
00081 int VerificarEstruturaFicheiro(const char* nomeFicheiro, const char* cabecalhoEsperado);
00082
00094 int VerificarIntegridadeFicheirosPacientes();
00095
00109 int DefinirCabecalhosFicheiro(const char* nomeFicheiro, const char* cabecalho);
00110
00122 int DefinirCabecalhosFicheirosPacientes();
00123
00137 int EscreverDadosPacientesParaCsv(const char* nomeFicheiro);
00138
00139 #endif // !CONTROLAR_FICHEIROS_H
```

4.5 Referência ao ficheiro EstruturaDados.h

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

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

Estruturas de Dados

- struct Paciente
- struct Dieta
- struct Plano

4.5.1 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

Created: 29.11.2023

Last modified: 06.12.2023

Copyright

© Enrique George Rodrigues, 2023. All right reserved.

4.6 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 // Estrutura dos pacientes
00025 typedef struct {
00026     int id; // <- Identificação única do paciente.
00027     char nome[50]; // <- Nome do paciente.
00028     long tel; // <- Número de telefone do paciente.
00029 } Paciente;
00030
00031 // Estrutura das dietas
00032 typedef struct {
00033     int idPaciente; // <- Identificação do paciente associado à dieta.
00034     time_t data; // <- Data em formato timestamp.
00035     char refeicao[20]; // <- Tipo de refeição (pequeno almoço, almoço, jantar).
00036     char comida[50]; // <- Nome da comida.
00037     int calorias; // <- Quantidade de calorias da comida.
00038 } Dieta;
00039
00040 // Estrutura dos planos
00041 typedef struct {
00042     int idPaciente; // <- Identificação do paciente associado ao plano.
00043     time_t data; // <- Data em formato timestamp.
00044     char refeicao[20]; // <- Tipo de refeição (pequeno almoço, almoço, jantar).
00045     int calorias[2]; // <- Intervalo de calorias. Índice 0 = cal mínimas, índice 1 = cal máximas.
00046 } Plano;
00047
00048 #endif
```

4.7 Referência ao ficheiro ExportarDados.c

Funções para exportar dados para ficheiros binários.

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

Funções

- int ExportarPacientesFichBinario (const char ∗nomeFicheiro, Paciente ∗pacientes, int numPacientes)

  Exporta os dados dos Pacientes para o ficheiro binário.
- int ExportarDietasFichBinario (const char ∗nomeFicheiro, Dieta ∗dietas, int numDietas)

  Exporta os dados das dietas para o ficheiro binário.
- int ExportarPlanosFichBinario (const char ∗nomeFicheiro, Plano ∗planos, int numPlanos)

  Exporta os dados dos planos para o ficheiro binário.

4.7.1 Descrição detalhada

Funções para exportar dados para ficheiros binários.

Este ficheiro contém implementações de funções para exportar dados de pacientes, dietas e planos para ficheiros binários.

Autor

Enrique George Rodrigues

Data

Created: 28.11.2023

Last modified: 02.12.2023

Copyright

© Enrique George Rodrigues, 2023. Todos os direitos reservados.

4.7.2 Documentação das funções

4.7.2.1 ExportarDietasFichBinario()

```
int ExportarDietasFichBinario (
            const char * nomeFicheiro,
            Dieta * dietas,
            int numDietas )
```

Exporta os dados das dietas para o ficheiro binário.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.
dietas	- As dietas na memória
numDietas	- O número de dietas na memória.

Valores retornados

0	- Sucesso na execução (SUCESSO).
12	- Erro ao abrir ficheiro (ERRO_ABRIR_FICHEIRO).

4.7.2.2 ExportarPacientesFichBinario()

```
int ExportarPacientesFichBinario (
            const char * nomeFicheiro,
            Paciente * pacientes,
            int numPacientes )
```

Exporta os dados dos Pacientes para o ficheiro binário.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.
pacientes	- Os pacientes da memória.
numPacientes	- O número de pacientes na memória.

Valores retornados

0	- Sucesso na execução (SUCESSO).
12	- Erro ao abrir ficheiro (ERRO_ABRIR_FICHEIRO).

4.7.2.3 ExportarPlanosFichBinario()

```
int ExportarPlanosFichBinario (
            const char * nomeFicheiro,
            Plano * planos,
            int numPlanos )
```

Exporta os dados dos planos para o ficheiro binário.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.
dietas	- Os planos na memória
numDietas	- O número de planos na memória.

Valores retornados

0	- Sucesso na execução (SUCESSO).
12	- Erro ao abrir ficheiro (ERRO_ABRIR_FICHEIRO).

4.8 Referência ao ficheiro ExportarDados.h

Declarações de funções para exportar dados para ficheiros binários.

```
#include "EstruturaDados.h"
```

Funções

- int ExportarPacientesFichBinario (const char ∗nomeFicheiro, Paciente ∗pacientes, int numPacientes)

  Exporta os dados dos Pacientes para o ficheiro binário.
- int ExportarDietasFichBinario (const char ∗nomeFicheiro, Dieta ∗dietas, int numDietas)

  Exporta os dados das dietas para o ficheiro binário.
- int ExportarPlanosFichBinario (const char ∗nomeFicheiro, Plano ∗planos, int numPlanos)

  Exporta os dados dos planos para o ficheiro binário.

4.8.1 Descrição detalhada

Declarações de funções para exportar dados para ficheiros binários.

Este ficheiro contém as declarações das funções para exportar dados de pacientes, dietas e planos para ficheiros binários.

Autor

Enrique George Rodrigues

Data

Created: 28.11.2023

Last modified: 02.12.2023

Copyright

© Enrique George Rodrigues, 2023. Todos os direitos reservados.

4.8.2 Documentação das funções

4.8.2.1 ExportarDietasFichBinario()

```
int ExportarDietasFichBinario (
            const char * nomeFicheiro,
            Dieta * dietas,
            int numDietas )
```

Exporta os dados das dietas para o ficheiro binário.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.
dietas	- As dietas na memória
numDietas	- O número de dietas na memória.

Valores retornados

0	- Sucesso na execução (SUCESSO).
12	- Erro ao abrir ficheiro (ERRO_ABRIR_FICHEIRO).

4.8.2.2 ExportarPacientesFichBinario()

```
int ExportarPacientesFichBinario (
            const char * nomeFicheiro,
            Paciente * pacientes,
            int numPacientes )
```

Exporta os dados dos Pacientes para o ficheiro binário.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.
pacientes	- Os pacientes da memória.
numPacientes	- O número de pacientes na memória.

Valores retornados

0	- Sucesso na execução (SUCESSO).
12	- Erro ao abrir ficheiro (ERRO_ABRIR_FICHEIRO).

4.8.2.3 ExportarPlanosFichBinario()

```
int ExportarPlanosFichBinario (
            const char * nomeFicheiro,
            Plano * planos,
            int numPlanos )
```

Exporta os dados dos planos para o ficheiro binário.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.
dietas	- Os planos na memória
numDietas	- O número de planos na memória.

Valores retornados

0	- Sucesso na execução (SUCESSO).
12	- Erro ao abrir ficheiro (ERRO_ABRIR_FICHEIRO).

4.9 ExportarDados.h

Ir para a documentação deste ficheiro.

```
00001
00018 #ifndef EXPORTAR_DADOS_H
00019 #define EXPORTAR_DADOS_H
00020
00021 #include "EstruturaDados.h"
00022
00033 int ExportarPacientesFichBinario(const char* nomeFicheiro, Paciente* pacientes, int numPacientes);
00034
00045 int ExportarDietasFichBinario(const char* nomeFicheiro, Dieta* dietas, int numDietas);
00046
00057 int ExportarPlanosFichBinario(const char* nomeFicheiro, Plano* planos, int numPlanos);
00058
00059 #endif
```

4.10 Referência ao ficheiro ImportarDados.c

Funções para importar dados a partir de ficheiros (binários e de texto).

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

Funções

- int ImportarPacientesFichBinario (const char *nomeFicheiro, Paciente pacientes[TAMANHO_MAX], int *numPacientes)

  Importa os dados dos Pacientes a partir do ficheiro binário.
- int ImportarDietasFichBinario (const char *nomeFicheiro, Dieta dietas[TAMANHO_MAX], int *numDietas)

  Importa os dados das Dietas a partir do ficheiro binário.
- int ImportarPlanosFichBinario (const char *nomeFicheiro, Plano planos[TAMANHO_MAX], int *numPlanos)

  Importa os dados dos Planos a partir do ficheiro binário.
- int ImportarPacientesFichTexto (const char *nomeFicheiro, Paciente pacientes[TAMANHO_MAX], int *num↩ Pacientes)

  Importa dados de pacientes de um ficheiro de texto e adiciona-os aos dados guardados, ignorando os recordes duplicados.
- int ImportarDietasFichTexto (const char *nomeFicheiro, Dieta dietas[TAMANHO_MAX], int *numDietas)

  Importa dados de dietas de um ficheiro de texto e adiciona-os aos dados guardados, ignorando os recordes duplicados.
- int ImportarPlanosFichTexto (const char *nomeFicheiro, Plano planos[TAMANHO_MAX], int *numPlanos)

  Importa dados de planos de um ficheiro de texto e adiciona-os aos dados guardados, ignorando os recordes duplicados.

4.10.1 Descrição detalhada

Funções para importar dados a partir de ficheiros (binários e de texto).

Este ficheiro contém implementações de funções para importar dados de pacientes, dietas e planos a partir de ficheiros binários e de texto. As funções são projetadas para adicionar novos dados aos dados já existentes, ignorando registros duplicados quando adequado.

Autor

Enrique George Rodrigues

Data

Created: 28.11.2023

Last modified: 06.12.2023

Copyright

© Enrique George Rodrigues, 2023. Todos os direitos reservados.

4.10.2 Documentação das funções

4.10.2.1 ImportarDietasFichBinario()

```
int ImportarDietasFichBinario (
            const char * nomeFicheiro,
            Dieta dietas[TAMANHO_MAX],
            int * numDietas )
```

Importa os dados das Dietas a partir do ficheiro binário.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.
dietas	- O array de dietas.
numDietas	- Um apontador para a variável que guarda o número de dietas.

Valores retornados

0	- Sucesso na execução (SUCESSO).
12	- Erro ao abrir o ficheiro (ERRO_ABRIR_FICHEIRO).
13	- Erro, tamanho máximo excedido. (ERRO_TAMANHO_EXCEDIDO).

4.10.2.2 ImportarDietasFichTexto()

```
int ImportarDietasFichTexto (
            const char * nomeFicheiro,
            Dieta dietas[TAMANHO_MAX],
            int * numDietas )
```

Importa dados de dietas de um ficheiro de texto e adiciona-os aos dados guardados, ignorando os recordes duplicados.

Parâmetros

nomeFicheiro	- O nome do ficheiro de texto.
dietas	- O array de dietas.
numDietas	- Um apontador para a variável que guarda o número de dietas.

Valores retornados

0	- Sucesso na execução (SUCESSO).
6	- Erro ao alocar memória (ERRO_ALOCAR_MEMORIA).
8	- Ficheiro vazio (ERRO_FICHEIRO_VAZIO).
9	- Erro na leitura do ficheiro (ERRO_LEITURA_FICHEIRO).
12	- Erro ao abrir o ficheiro (ERRO_ABRIR_FICHEIRO).
13	- Erro, tamanho máximo excedido. (ERRO_TAMANHO_EXCEDIDO).

4.10.2.3 ImportarPacientesFichBinario()

```
int ImportarPacientesFichBinario (
            const char * nomeFicheiro,
            Paciente pacientes[TAMANHO_MAX],
            int * numPacientes )
```

Importa os dados dos Pacientes a partir do ficheiro binário.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.
pacientes	- O array de pacientes.
numPacientes	- Um apontador para a variável que guarda o número de pacientes.

Valores retornados

0	- Sucesso na execução (SUCESSO).
12	- Erro ao abrir o ficheiro (ERRO_ABRIR_FICHEIRO).
13	- Erro, tamanho máximo excedido. (ERRO_TAMANHO_EXCEDIDO).

4.10.2.4 ImportarPacientesFichTexto()

```
int ImportarPacientesFichTexto (
            const char * nomeFicheiro,
            Paciente pacientes[TAMANHO_MAX],
            int * numPacientes )
```

Importa dados de pacientes de um ficheiro de texto e adiciona-os aos dados guardados, ignorando os recordes duplicados.

Parâmetros

nomeFicheiro	- O nome do ficheiro de texto.
pacientes	- O array de pacientes.
numPacientes	- Um apontador para a variável que guarda o número de pacientes.

Valores retornados

0	- Sucesso na execução (SUCESSO).
6	- Erro ao alocar memória (ERRO_ALOCAR_MEMORIA).
8	- Ficheiro vazio (ERRO_FICHEIRO_VAZIO).
9	- Erro na leitura do ficheiro (ERRO_LEITURA_FICHEIRO).
12	- Erro ao abrir o ficheiro (ERRO_ABRIR_FICHEIRO).
13	- Erro, tamanho máximo excedido. (ERRO_TAMANHO_EXCEDIDO).

4.10.2.5 ImportarPlanosFichBinario()

```
int ImportarPlanosFichBinario (
            const char * nomeFicheiro,
            Plano planos[TAMANHO_MAX],
            int * numPlanos )
```

Importa os dados dos Planos a partir do ficheiro binário.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.
planos	- O array de planos.
numPlanos	- Um apontador para a variável que guarda o número de planos.

Valores retornados

0	- Sucesso na execução (SUCESSO).
12	- Erro ao abrir o ficheiro (ERRO_ABRIR_FICHEIRO).
13	- Erro, tamanho máximo excedido. (ERRO_TAMANHO_EXCEDIDO).

4.10.2.6 ImportarPlanosFichTexto()

```
int ImportarPlanosFichTexto (
            const char * nomeFicheiro,
            Plano planos[TAMANHO_MAX],
            int * numPlanos )
```

Importa dados de planos de um ficheiro de texto e adiciona-os aos dados guardados, ignorando os recordes duplicados.

Parâmetros

nomeFicheiro	- O nome do ficheiro de texto.
planos	- O array de planos.
numPlanos	- Um apontador para a variável que guarda o número de planos.

Valores retornados

0	- Sucesso na execução (SUCESSO).
6	- Erro ao alocar memória (ERRO_ALOCAR_MEMORIA).
8	- Ficheiro vazio (ERRO_FICHEIRO_VAZIO).
9	- Erro na leitura do ficheiro (ERRO_LEITURA_FICHEIRO).
12	- Erro ao abrir o ficheiro (ERRO_ABRIR_FICHEIRO).
13	- Erro, tamanho máximo excedido. (ERRO_TAMANHO_EXCEDIDO).

4.11 Referência ao ficheiro ImportarDados.h

Funções para importação de dados a partir de ficheiros binários e de texto.

```
#include "EstruturaDados.h"
```

Funções

- int ImportarPacientesFichBinario (const char ∗nomeFicheiro, Paciente pacientes[TAMANHO_MAX], int ∗numPacientes)

  Importa os dados dos Pacientes a partir do ficheiro binário.
- int ImportarDietasFichBinario (const char ∗nomeFicheiro, Dieta dietas[TAMANHO_MAX], int ∗numDietas)

  Importa os dados das Dietas a partir do ficheiro binário.
- int ImportarPlanosFichBinario (const char ∗nomeFicheiro, Plano planos[TAMANHO_MAX], int ∗numPlanos)

  Importa os dados dos Planos a partir do ficheiro binário.
- int ImportarPacientesFichTexto (const char ∗nomeFicheiro, Paciente pacientes[TAMANHO_MAX], int ∗num↩ Pacientes)

  Importa dados de pacientes de um ficheiro de texto e adiciona-os aos dados guardados, ignorando os recordes duplicados.
- int ImportarDietasFichTexto (const char ∗nomeFicheiro, Dieta dietas[TAMANHO_MAX], int ∗numDietas)

  Importa dados de dietas de um ficheiro de texto e adiciona-os aos dados guardados, ignorando os recordes duplicados.
- int ImportarPlanosFichTexto (const char ∗nomeFicheiro, Plano planos[TAMANHO_MAX], int ∗numPlanos)

  Importa dados de planos de um ficheiro de texto e adiciona-os aos dados guardados, ignorando os recordes duplicados.

4.11.1 Descrição detalhada

Funções para importação de dados a partir de ficheiros binários e de texto.

Este ficheiro contém declarações de funções responsáveis por importar dados relacionados a Pacientes, Dietas e Planos a partir de ficheiros binários e de texto. As funções aqui declaradas permitem carregar esses dados para as estruturas de dados definidas no ficheiro "EstruturaDados.h".

Autor

Enrique George Rodrigues

Data

Created: 28.11.2023

Last modified: 05.12.2023

Copyright

© Enrique George Rodrigues, 2023. Todos os direitos reservados.

4.11.2 Documentação das funções

4.11.2.1 ImportarDietasFichBinario()

```
int ImportarDietasFichBinario (
            const char * nomeFicheiro,
            Dieta dietas[TAMANHO_MAX],
            int * numDietas )
```

Importa os dados das Dietas a partir do ficheiro binário.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.
dietas	- O array de dietas.
numDietas	- Um apontador para a variável que guarda o número de dietas.

Valores retornados

0	- Sucesso na execução (SUCESSO).
12	- Erro ao abrir o ficheiro (ERRO_ABRIR_FICHEIRO).
13	- Erro, tamanho máximo excedido. (ERRO_TAMANHO_EXCEDIDO).

4.11.2.2 ImportarDietasFichTexto()

```
int ImportarDietasFichTexto (
            const char * nomeFicheiro,
            Dieta dietas[TAMANHO_MAX],
            int * numDietas )
```

Importa dados de dietas de um ficheiro de texto e adiciona-os aos dados guardados, ignorando os recordes duplicados.

Parâmetros

nomeFicheiro	- O nome do ficheiro de texto.
dietas	- O array de dietas.
numDietas	- Um apontador para a variável que guarda o número de dietas.

Valores retornados

0	- Sucesso na execução (SUCESSO).
6	- Erro ao alocar memória (ERRO_ALOCAR_MEMORIA).
8	- Ficheiro vazio (ERRO_FICHEIRO_VAZIO).
9	- Erro na leitura do ficheiro (ERRO_LEITURA_FICHEIRO).
12	- Erro ao abrir o ficheiro (ERRO_ABRIR_FICHEIRO).
13	- Erro, tamanho máximo excedido. (ERRO_TAMANHO_EXCEDIDO).

4.11.2.3 ImportarPacientesFichBinario()

```
int ImportarPacientesFichBinario (
            const char * nomeFicheiro,
            Paciente pacientes[TAMANHO_MAX],
            int * numPacientes )
```

Importa os dados dos Pacientes a partir do ficheiro binário.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.
pacientes	- O array de pacientes.
numPacientes	- Um apontador para a variável que guarda o número de pacientes.

Valores retornados

0	- Sucesso na execução (SUCESSO).
12	- Erro ao abrir o ficheiro (ERRO_ABRIR_FICHEIRO).
13	- Erro, tamanho máximo excedido. (ERRO_TAMANHO_EXCEDIDO).

4.11.2.4 ImportarPacientesFichTexto()

```
int ImportarPacientesFichTexto (
            const char * nomeFicheiro,
            Paciente pacientes[TAMANHO_MAX],
            int * numPacientes )
```

Importa dados de pacientes de um ficheiro de texto e adiciona-os aos dados guardados, ignorando os recordes duplicados.

Parâmetros

nomeFicheiro	- O nome do ficheiro de texto.
pacientes	- O array de pacientes.
numPacientes	- Um apontador para a variável que guarda o número de pacientes.

Valores retornados

0	- Sucesso na execução (SUCESSO).
6	- Erro ao alocar memória (ERRO_ALOCAR_MEMORIA).
8	- Ficheiro vazio (ERRO_FICHEIRO_VAZIO).
9	- Erro na leitura do ficheiro (ERRO_LEITURA_FICHEIRO).
12	- Erro ao abrir o ficheiro (ERRO_ABRIR_FICHEIRO).
13	- Erro, tamanho máximo excedido. (ERRO_TAMANHO_EXCEDIDO).

4.11.2.5 ImportarPlanosFichBinario()

```
int ImportarPlanosFichBinario (
            const char * nomeFicheiro,
            Plano planos[TAMANHO_MAX],
            int * numPlanos )
```

Importa os dados dos Planos a partir do ficheiro binário.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.
planos	- O array de planos.
numPlanos	- Um apontador para a variável que guarda o número de planos.

Valores retornados

0	- Sucesso na execução (SUCESSO).
12	- Erro ao abrir o ficheiro (ERRO_ABRIR_FICHEIRO).
13	- Erro, tamanho máximo excedido. (ERRO_TAMANHO_EXCEDIDO).

4.11.2.6 ImportarPlanosFichTexto()

```
int ImportarPlanosFichTexto (
            const char * nomeFicheiro,
            Plano planos[TAMANHO_MAX],
            int * numPlanos )
```

Importa dados de planos de um ficheiro de texto e adiciona-os aos dados guardados, ignorando os recordes duplicados.

Parâmetros

nomeFicheiro	- O nome do ficheiro de texto.
planos	- O array de planos.
numPlanos	- Um apontador para a variável que guarda o número de planos.

Valores retornados

0	- Sucesso na execução (SUCESSO).
6	- Erro ao alocar memória (ERRO_ALOCAR_MEMORIA).
8	- Ficheiro vazio (ERRO_FICHEIRO_VAZIO).
9	- Erro na leitura do ficheiro (ERRO_LEITURA_FICHEIRO).
12	- Erro ao abrir o ficheiro (ERRO_ABRIR_FICHEIRO).
13	- Erro, tamanho máximo excedido. (ERRO_TAMANHO_EXCEDIDO).

4.12 ImportarDados.h

Ir para a documentação deste ficheiro.

```
00001
00021 #ifndef IMPORTAR_DADOS_H
00022 #define IMPORTAR_DADOS_H
00023
00024 #include "EstruturaDados.h"
00025
00037 int ImportarPacientesFichBinario(const char* nomeFicheiro, Paciente pacientes[TAMANHO_MAX], int*
      numPacientes);
00038
```

```
00050 int ImportarDietasFichBinario(const char* nomeFicheiro, Dieta dietas[TAMANHO_MAX], int* numDietas);
00051
00063 int ImportarPlanosFichBinario(const char* nomeFicheiro, Plano planos[TAMANHO_MAX], int* numPlanos);
00064
00080 int ImportarPacientesFichTexto(const char* nomeFicheiro, Paciente pacientes[TAMANHO_MAX], int*
      numPacientes);
00081
00097 int ImportarDietasFichTexto(const char* nomeFicheiro, Dieta dietas[TAMANHO_MAX], int* numDietas);
00098
00114 int ImportarPlanosFichTexto(const char* nomeFicheiro, Plano planos[TAMANHO_MAX], int* numPlanos);
00115
00116 #endif
```

4.13 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 "VerificacoesIniciais.h"
#include "OperacoesFicheiros.h"
#include "ImportarDados.h"
#include "ExportarDados.h"
```

Funções

- int main (int argc, char ∗argv[ ])

  Função principal que controla a execução do programa.

4.13.1 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

Created: 16.11.2023

Last modified: 06.12.2023

Copyright

© Enrique George Rodrigues, 2023. All right reserved.

4.13.2 Documentação das funções

4.13.2.1 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).

4.14 Referência ao ficheiro Menu.c

Implementação das funções relacionadas ao menu de opções do programa.

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

Funções

- void ApresentarMenu ()

  Apresenta o menu de opções para interação com o programa.

4.14.1 Descrição detalhada

Implementação das funções relacionadas ao menu de opções do programa.

Autor

Enrique George Rodrigues

Data

Created: 18.11.2023

Last modified: 03.12.2023

Copyright

© Enrique George Rodrigues, 2023. Todos os direitos reservados.

4.14.2 Documentação das funções

4.14.2.1 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.

4.15 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.

4.15.1 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

Created: 18.11.2023

Last modified: 03.12.2023

Copyright

© Enrique George Rodrigues, 2023. All right reserved.

4.15.2 Documentação das funções

4.15.2.1 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.

4.16 Menu.h

Ir para a documentação deste ficheiro.

```
00001
00018 #ifndef MENU_H
00019 #define MENU_H
00020
00036 void ApresentarMenu();
00037
00038 #endif
```

4.17 Referência ao ficheiro OperacoesFicheiros.c

Implementação das funções relacionadas às operações em ficheiros.

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

Funções

- bool ExisteFicheiroBinario (const char *nomeFicheiro)

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

  Cria um ficheiro binário com um dado nome.

4.17.1 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

Created: 28.11.2023

Last modified: 03.12.2023

Copyright

© Enrique George Rodrigues, 2023. Todos os direitos reservados.

4.17.2 Documentação das funções

4.17.2.1 CriarFicheiroBinario()

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

Cria um ficheiro binário com um dado nome.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.

Valores retornados

-	0 se conseguir criar o ficheiro - (SUCESSO).
-	2 se não conseguir criar o ficheiro - (ERRO_CRIAR_FICHEIRO).

4.17.2.2 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)

4.18 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 CriarFicheiroBinario (const char *nomeFicheiro)

  Cria um ficheiro binário com um dado nome.

4.18.1 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

Created: 28.11.2023

Last modified: 03.12.2023

Copyright

© Enrique George Rodrigues, 2023. Todos os direitos reservados.

4.18.2 Documentação das funções

4.18.2.1 CriarFicheiroBinario()

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

Cria um ficheiro binário com um dado nome.

Parâmetros

nomeFicheiro	- O nome do ficheiro binário.

Valores retornados

-	0 se conseguir criar o ficheiro - (SUCESSO).
-	2 se não conseguir criar o ficheiro - (ERRO_CRIAR_FICHEIRO).

4.18.2.2 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)

4.19 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
00029 bool ExisteFicheiroBinario(const char* nomeFicheiro);
00030
00037 int CriarFicheiroBinario(const char* nomeFicheiro);
00038
00039 #endif
```

4.20 Referência ao ficheiro Utils.c

Funções utilitárias do programa.

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include <time.h>
#include <string.h>
#include <unistd.h>
#include <curses.h>
#include "Constantes.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 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 TerminarProgramaComErro (int codigoErro)

  Termina o programa apresentando uma mensagem de erro e terminando com um código de erro.
- int ApagarFicheirosBinarios ()

  Apaga os ficheiros binarios que servem de base de dados da aplicacao.
- 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.
- void LibertarMemoria (Paciente **pacientes, int *numPacientes, Dieta **dietas, int *numDietas, Plano **planos, int *numPlanos)

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

4.20.1 Descrição detalhada

Funções utilitárias do programa.

Autor

Enrique George Rodrigues

Data

Criado: 18.11.2023

Última modificação: 06.12.2023

Copyright

© Enrique George Rodrigues, 2023. Todos os direitos reservados.

4.20.2 Documentação das funções

4.20.2.1 ApagarFicheirosBinarios()

```
int ApagarFicheirosBinarios ( )
```

Apaga os ficheiros binarios que servem de base de dados da aplicacao.

Util quando se quer fazer um "reset" em caso de problemas. Esta ação é irreversível e deve ser utilizado com cuidado. Pergunte sempre ao utilizador para confirmar.

Aviso

Esta função apaga TODOS os dados dos utilizadores. DEVE SER UTILIZADO COM CUIDADO!

Valores retornados

-	5 se houver erro ao apagar os ficheiros binários - ERRO_APAGAR_FICHEIROS_BIN. 0 se correr com sucesso - SUCESSO.

4.20.2.2 ApresentarAjuda()

```
void ApresentarAjuda ( )
```

Apresenta o manual de ajuda ao utilizador.

Esta função escreve o texto especificado no terminal com o objetivo de ajudar o utilizador.

4.20.2.3 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).

4.20.2.4 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.

4.20.2.5 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);`

4.20.2.6 LerTecla()

```
int LerTecla ( )
```

Uma função que permite ler um carater.

Nota

- A função usa o "_getch()" para Windows e "getch()" para sistemas Unix.

4.20.2.7 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.

4.20.2.8 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").

4.20.2.9 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.

4.21 Referência ao ficheiro Utils.h

Ficheiro de cabeçalho para utilitários do programa.

```
#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.
- int ApagarFicheirosBinarios ()
  Apaga os ficheiros binarios que servem de base de dados da aplicacao.
- 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.
- void LibertarMemoria (Paciente **pacientes, int *numPacientes, Dieta **dietas, int *numDietas, Plano **planos, int *numPlanos)
  Liberta toda a memoria alocada aos pacientes, dietas e planos.

4.21.1 Descrição detalhada

Ficheiro de cabeçalho para utilitários do programa.

Este ficheiro contém declarações para funções utilitárias do programa.

Autor

Enrique George Rodrigues

Data

Created: 18.11.2023

Last modified: 06.12.2023

Copyright

© Enrique George Rodrigues, 2023. Todos os direitos reservados.

4.21.2 Documentação das funções

4.21.2.1 ApagarFicheirosBinarios()

```
int ApagarFicheirosBinarios ( )
```

Apaga os ficheiros binarios que servem de base de dados da aplicacao.

Util quando se quer fazer um "reset" em caso de problemas. Esta ação é irreversível e deve ser utilizado com cuidado. Pergunte sempre ao utilizador para confirmar.

Valores retornados

-	0 se houver erro. 1 se os ficheiros foram apagados com sucesso.

Util quando se quer fazer um "reset" em caso de problemas. Esta ação é irreversível e deve ser utilizado com cuidado. Pergunte sempre ao utilizador para confirmar.

Aviso

Esta função apaga TODOS os dados dos utilizadores. DEVE SER UTILIZADO COM CUIDADO!

Valores retornados

-	5 se houver erro ao apagar os ficheiros binários - ERRO_APAGAR_FICHEIROS_BIN. 0 se correr com sucesso - SUCESSO.

4.21.2.2 ApresentarAjuda()

```
void ApresentarAjuda ( )
```

Apresenta o manual de ajuda ao utilizador.

Esta função escreve o texto especificado no terminal com o objetivo de ajudar o utilizador.

4.21.2.3 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).

4.21.2.4 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.

4.21.2.5 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);`

4.21.2.6 LerTecla()

```
int LerTecla ( )
```

Uma função que permite ler um carater.

Nota

- A função usa o "_getch()" para Windows e "getch()" para sistemas Unix.

4.21.2.7 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.

4.21.2.8 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").

4.21.2.9 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.

4.22 Utils.h

Ir para a documentação deste ficheiro.

```
00001
00017 #ifndef UTILS_H
00018 #define UTILS_H
00019
00020 #include "EstruturaDados.h"
00021
00029 void CustomSleep(int segundos);
00030
00036 int LerTecla();
00037
00047 void LimparEcra();
00048
00058 void TerminarProgramaComErro(int erro);
00059
00073 void EscreverComCor(const char* cor, const char* formato, ...);
00074
00081 void ApresentarAjuda();
00082
00092 int ApagarFicheirosBinarios();
00093
00097 void PerguntarUtilizadorIniciarPrograma();
00098
00105 time_t converterStringParaDatetime(const char* dateString);
00106
00116 void LibertarMemoria(Paciente** pacientes, int* numPacientes, Dieta** dietas, int* numDietas, Plano**
      planos, int* numPlanos);
00117
00118 #endif
```

4.23 Referência ao ficheiro VerificacoesIniciais.c

Funções para realizar verificações iniciais antes da execução do programa.

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

Funções

- int VerificarFicheiroBinario (const char *nomeFicheiro)

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

4.23.1 Descrição detalhada

Funções para realizar verificações iniciais antes da execução do programa.

Este ficheiro contém todas as funções para as verificações iniciais.

Autor

Enrique George Rodrigues

Data

Criado: 28.11.2023

Última modificação: 06.12.2023

Copyright

© Enrique George Rodrigues, 2023. Todos os direitos reservados.

4.23.2 Documentação das funções

4.23.2.1 VerificarFicheiroBinario()

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

Função que verifica se exite um ficheiros 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 - Sucesso na execução (SUCESSO).
-	2 - Erro a criar o ficheiro (ERRO_CRIAR_FICHEIRO).

4.24 Referência ao ficheiro VerificacoesIniciais.h

Ficheiro de cabeçalho para verificações iniciais do programa.

Funções

- int VerificarFicheiroBinario (const char ∗nomeFicheiro)

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

4.24.1 Descrição detalhada

Ficheiro de cabeçalho para verificações iniciais do programa.

Este ficheiro contém declarações para funções que realizam verificações iniciais, como a existência de ficheiros binários necessários, antes de iniciar o programa.

Autor

Enrique George Rodrigues

Data

Created: 28.11.2023

Last modified: 03.12.2023

Copyright

© Enrique George Rodrigues, 2023. Todos os direitos reservados.

4.24.2 Documentação das funções

4.24.2.1 VerificarFicheiroBinario()

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

Função que verifica se exite um ficheiros 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 - Sucesso na execução (SUCESSO).
-	2 - Erro a criar o ficheiro (ERRO_CRIAR_FICHEIRO).

4.25 VerificacoesIniciais.h

Ir para a documentação deste ficheiro.

```
00001
00018 #ifndef VERIFICACOES_INICIAIS_H
00019 #define VERIFICACOES_INICIAIS_H
00020
00033 int VerificarFicheiroBinario(const char* nomeFicheiro);
00034
00035 #endif
```

Índice