Trabalho Prático d-18

Generated by Doxygen 1.9.1

1 Main Page 1

1.1 Trabalho Prático . . . 1

1.1.1 grupo 18 . . . 1

1.1.2 Organização . . . 1

1.1.3 Compilação e Execução . . . 1

1.1.3.1 Compilação com make . . . 1

1.1.3.2 Opções de execução . . . 2

1.1.3.3 Comandos de execução . . . 2

1.1.4 Divisão de Tarefas . . . 2

2 Data Structure Index 3

2.1 Data Structures . . . 3

3 File Index 5

3.1 File List . . . 5

4 Data Structure Documentation 7

4.1 Dieta Struct Reference . . . 7

4.1.1 Field Documentation . . . 7

4.1.1.1 alimento . . . 7

4.1.1.2 calorias . . . 7

4.1.1.3 datafim . . . 7

4.1.1.4 datainicio . . . 8

4.1.1.5 numpaciente . . . 8

4.1.1.6 tp . . . 8

4.2 MediaPaciente Struct Reference . . . 8

4.2.1 Field Documentation . . . 8

4.2.1.1 mediaalmoco . . . 8

4.2.1.2 mediajantar . . . 8

4.2.1.3 mediapequenoalmoco . . . 9

4.2.1.4 numpaciente . . . 9

4.3 NaoCumpPaciente Struct Reference . . . 9

4.3.1 Field Documentation . . . 9

4.3.1.1 nome . . . 9

4.3.1.2 numpaciente . . . 9

4.3.1.3 telefone . . . 9

4.4 Paciente Struct Reference . . . 10

4.4.1 Field Documentation . . . 10

4.4.1.1 nome . . . 10

4.4.1.2 numpaciente . . . 10

4.4.1.3 telefone . . . 10

4.5 PlanoNutri Struct Reference . . . 10

4.5.1 Field Documentation . . . 11

4.5.1.1 caloriasMax . . . 11
4.5.1.2 caloriasMin . . . 11
4.5.1.3 datafim . . . 11
4.5.1.4 datainicio . . . 11
4.5.1.5 numpaciente . . . 11
4.5.1.6 tp . . . 11

5 File Documentation **13**

5.1 /home/carolina/Desktop/Trabalho Prático/d-18/src/Dados.h File Reference . . . 13
5.1.1 Detailed Description . . . 15
5.2 /home/carolina/Desktop/Trabalho Prático/d-18/src/Funcoes.h File Reference . . . 15
5.2.1 Detailed Description . . . 17
5.2.2 Function Documentation . . . 17
5.2.2.1 CalcularMediasRefeicoes() . . . 17
5.2.2.2 ExportaDadosDieta() . . . 18
5.2.2.3 ExportaDadosPacientes() . . . 19
5.2.2.4 ExportaDadosPlanoNutri() . . . 19
5.2.2.5 GerarTabelaRefeicoes() . . . 20
5.2.2.6 IdentificaForaIntervalo() . . . 21
5.2.2.7 LeDadosDieta() . . . 21
5.2.2.8 LeDadosPacientes() . . . 22
5.2.2.9 LeDadosPlanoNutri() . . . 23
5.2.2.10 ListaPlanoNutricional() . . . 24
5.2.2.11 MostraAjuda() . . . 25
5.2.2.12 NumPacientesUltrapassamCal() . . . 25
5.2.2.13 OrdenaPacientesForaIntervalo() . . . 26
5.3 /home/carolina/Desktop/Trabalho Prático/d-18/src/GereDietas.c File Reference . . . 26
5.3.1 Detailed Description . . . 28
5.3.2 Function Documentation . . . 28
5.3.2.1 CalcularMediasRefeicoes() . . . 28
5.3.2.2 ExportaDadosDieta() . . . 29
5.3.2.3 ExportaDadosPlanoNutri() . . . 30
5.3.2.4 IdentificaForaIntervalo() . . . 30
5.3.2.5 LeDadosDieta() . . . 31
5.3.2.6 LeDadosPlanoNutri() . . . 32
5.3.2.7 ListaPlanoNutricional() . . . 33
5.3.2.8 MostraAjuda() . . . 34
5.3.2.9 NumPacientesUltrapassamCal() . . . 34
5.3.2.10 OrdenaPacientesForaIntervalo() . . . 34
5.4 /home/carolina/Desktop/Trabalho Prático/d-18/src/GerePacientes.c File Reference . . . 35
5.4.1 Detailed Description . . . 36
5.4.2 Function Documentation . . . 36

5.4.2.1 ExportaDadosPacientes() 36
5.4.2.2 GerarTabelaRefeicoes() 37
5.4.2.3 LeDadosPacientes() 38
5.5 /home/carolina/Desktop/Trabalho Prático/d-18/src/main.c File Reference 38
5.5.1 Detailed Description 39
5.5.2 Function Documentation 40
5.5.2.1 main() 40

Chapter 1

Main Page

1.1 Trabalho Prático

Licenciatura em Engenharia de Sistemas Informáticos 2023-24

Laboratórios de Informática

1.1.1 grupo 18

Número	Nome
27960	Pedro Ribeiro
27961	Ricardo Fernandes
27983	Carolina Branco

1.1.2 Organização

`Documentação Doxygen` Documentação gerada pelo Doxygen

`Relatório` Relatório pdf desenvolvido

`Código da Solução` Código da solução desenvolvida

1.1.3 Compilação e Execução

Aqui estão os passos para compilar e executar o programa:

1.1.3.1 Compilação com make

Para compilar o programa, podemos usar o seguinte comando make:

- 'make'

1.1.3.2 Opções de execução

O Makefile oferece diferentes opções que podem ser utilizadas para executar o programa:

- 'run2f': A opção run2f é utilizada para executar o programa sem nenhuma opção extra com apenas dois ficheiros passados por argumento com os dados separados por ponto e vírgulas (;)
- 'run3f': A opção run2f é utilizada para executar o programa sem nenhuma opção extra com três ficheiros passados por argumento com os dados separados por ponto e vírgulas (;)
- 'ajuda': A opção ajuda é utilizada para fornecer ajuda ao utilizador sobre os comandos a utilizar
- 'tab2f': A opção tab2f é utilizada para executar o programa com apenas dois ficheiros passados por argumento com os dados separados por tabs
- 'tab3f': A opção tab3f é utilizada para executar o programa com três ficheiros passados por argumento com os dados separados por tabs

1.1.3.3 Comandos de execução

Para executar o programa utilizamos os seguintes comandos:

- 'make run2f': Compila e executa o programa com a opção 'run2f'
- 'make run3f': Compila e executa o programa com a opção 'run3f'
- 'make ajuda': Compila e executa o programa com a opção 'ajuda'
- 'make tab2f': Compila e executa o programa com a opção 'tab2f'
- 'make tab3f': Compila e executa o programa com a opção 'tab3f'

1.1.4 Divisão de Tarefas

Durante a execução do projeto, embora tenhamos colaborado de forma conjunta em várias etapas e nos encontrássemos envolvidos em todas as partes do trabalho, identificamos uma distribuição específica de responsabilidades. Cada membro do grupo concentrou-se em áreas específicas, resultando numa colaboração eficiente. As principais responsabilidades de cada membro foram:

- Pedro Ribeiro (27960): Manipulação do código para a possibilidade de inserir dados por argumentos e por stdin;
- Ricardo Fernandes (27961): Desenvolvimento da Makefile e do ficheiro README.md;
- Carolina Branco (27983): Configuração e criação da documentação Doxygen e criação do documento refman.pdf.

Além de todas estas funções, cada um de nós participou igualmente na realização do relatório, cada um na sua responsabilidade correspondente.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Dieta 7
MediaPaciente 8
NaoCumpPaciente 9
Paciente 10
PlanoNutri 10

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

/home/carolina/Desktop/Trabalho Prático/d-18/src/Dados.h
Fornece estruturas de dados que podem ser usadas para representar informações sobre pacientes, dietas, planos nutricionais e métricas relacionadas à ingestão de calorias 13
/home/carolina/Desktop/Trabalho Prático/d-18/src/Funcoes.h
Declaração das funções 15
/home/carolina/Desktop/Trabalho Prático/d-18/src/GereDietas.c
Leitura, escrita e manipulação de informações relacionadas com dietas e planos nutricionais . 26
/home/carolina/Desktop/Trabalho Prático/d-18/src/GerePacientes.c
Leitura, escrita e manipulação de informações relacionadas a pacientes 35
/home/carolina/Desktop/Trabalho Prático/d-18/src/main.c
Local de implementação das funções 38

Chapter 4

Data Structure Documentation

4.1 Dieta Struct Reference

Data Fields

- int numpaciente
- char datainicio [N]
- char datafim [N]
- Refeicao tp
- char alimento [M]
- int calorias

4.1.1 Field Documentation

4.1.1.1 alimento

```
char alimento[M]
```

Alimento ingerido pelo paciente

4.1.1.2 calorias

```
int calorias
```

Calorias ingeridas pelo paciente

4.1.1.3 datafim

```
char datafim[N]
```

Data fim da dieta

4.1.1.4 datainicio

```
char datainicio[N]
```

Data de início da dieta

4.1.1.5 numpaciente

```
int numpaciente
```

Número de paciente

4.1.1.6 tp

```
Refeicao tp
```

Tipo da refeição (Pequeno-almoço;Almoço;Jantar)

The documentation for this struct was generated from the following file:

- /home/carolina/Desktop/Trabalho Prático/d-18/src/Dados.h

4.2 MediaPaciente Struct Reference

Data Fields

- int numpaciente
- double mediapequenoalmoco
- double mediaalmoco
- double mediajantar

4.2.1 Field Documentation

4.2.1.1 mediaalmoco

```
double mediaalmoco
```

Média de calorias consumidas durante o almoço durante um período específico de tempo

4.2.1.2 mediajantar

```
double mediajantar
```

Média de calorias consumidas durante o jantar durante um período específico de tempo

4.2.1.3 mediapequenoalmoco

```
double mediapequenoalmoco
```

Média de calorias consumidas durante o pequeno-almoço durante um período específico de tempo

4.2.1.4 numpaciente

```
int numpaciente
```

Número de paciente

The documentation for this struct was generated from the following file:

- /home/carolina/Desktop/Trabalho Prático/d-18/src/Dados.h

4.3 NaoCumpPaciente Struct Reference

Data Fields

- int numpaciente
- char nome [N]
- int telefone

4.3.1 Field Documentation

4.3.1.1 nome

```
char nome[N]
```

Nome do paciente

4.3.1.2 numpaciente

```
int numpaciente
```

Número do paicente

4.3.1.3 telefone

```
int telefone
```

Número de telefone do paciente

The documentation for this struct was generated from the following file:

- /home/carolina/Desktop/Trabalho Prático/d-18/src/Dados.h

4.4 Paciente Struct Reference

Data Fields

- int numpaciente
- char nome [N]
- int telefone

4.4.1 Field Documentation

4.4.1.1 nome

```
char nome[N]
```

Nome do paciente

4.4.1.2 numpaciente

```
int numpaciente
```

Número de paciente

4.4.1.3 telefone

```
int telefone
```

Número do telefone do paciente

The documentation for this struct was generated from the following file:

- /home/carolina/Desktop/Trabalho Prático/d-18/src/Dados.h

4.5 PlanoNutri Struct Reference

Data Fields

- int numpaciente
- char datainicio [N]
- char datafim [N]
- Refeicao tp
- int caloriasMin
- int caloriasMax

4.5.1 Field Documentation

4.5.1.1 caloriasMax

```
int caloriasMax
```

Número máximo de calorias que o paciente deve consumir

4.5.1.2 caloriasMin

```
int caloriasMin
```

Número mínimo de calorias que o paciente deve consumir

4.5.1.3 datafim

```
char datafim[N]
```

Data de fim do plano nutricional

4.5.1.4 datainicio

```
char datainicio[N]
```

Data de início do plano nutricional

4.5.1.5 numpaciente

```
int numpaciente
```

Número de paciente

4.5.1.6 tp

```
Refeicao tp
```

Tipo da refeição (Pequeno-almoço;Almoço;Jantar)

The documentation for this struct was generated from the following file:

- /home/carolina/Desktop/Trabalho Prático/d-18/src/Dados.h

Chapter 5

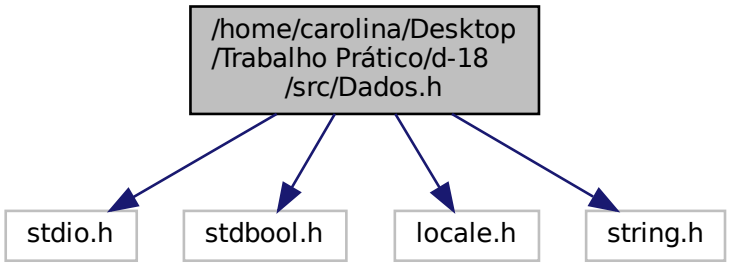
File Documentation

5.1 /home/carolina/Desktop/Trabalho Prático/d-18/src/Dados.h File Reference

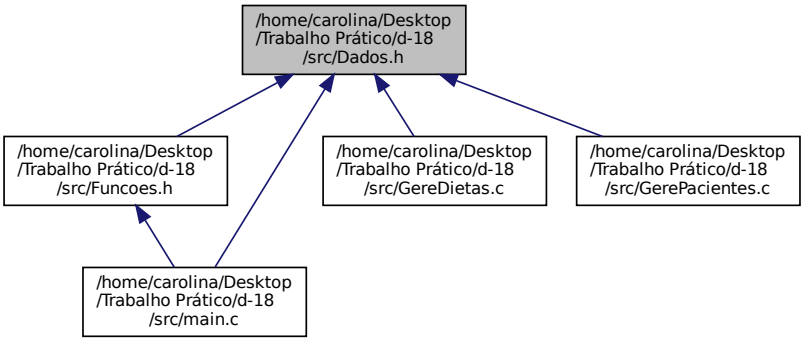
Fornece estruturas de dados que podem ser usadas para representar informações sobre pacientes, dietas, planos nutricionais e métricas relacionadas à ingestão de calorias.

```
#include <stdio.h>
#include <stdbool.h>
#include <locale.h>
#include <string.h>
```

Include dependency graph for Dados.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- struct Dieta
- struct Paciente
- struct PlanoNutri
- struct NaoCumpPaciente
- struct MediaPaciente

Macros

- #define **N** 30
- #define **M** 40
- #define **L** 50
- #define **E** 5

Typedefs

- typedef struct Dieta **Dieta**
- typedef struct Paciente **Paciente**
- typedef struct PlanoNutri **PlanoNutri**
- typedef struct NaoCumpPaciente **NaoCumpPaciente**
- typedef struct MediaPaciente **MediaPaciente**

Enumerations

- enum **Refeicao** { **PEQUENO_ALMOCO** , **ALMOCO** , **JANTAR** }

5.1.1 Detailed Description

Fornece estruturas de dados que podem ser usadas para representar informações sobre pacientes, dietas, planos nutricionais e métricas relacionadas à ingestão de calorias.

Author

Pedro Ribeiro, Ricardo Fernandes, Carolina Branco (a27960@alunos.ipca.pt a279861@alunos.↩ opca.pt a27983@alunos.ipca.pt)

Version

0.1

Date

2023-12-27

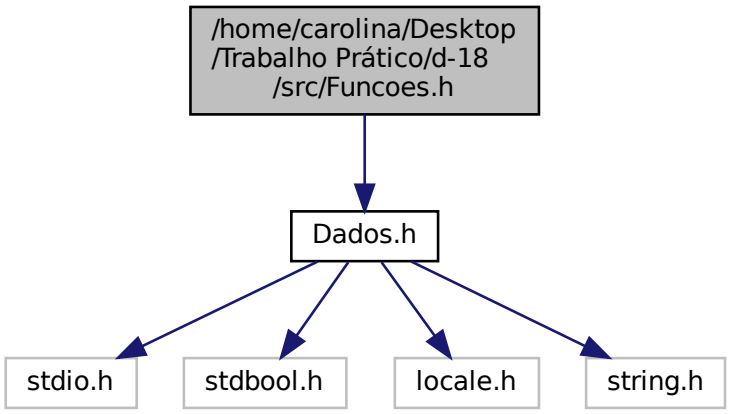
Copyright

5.2 /home/carolina/Desktop/Trabalho Prático/d-18/src/Funcoes.h File Reference

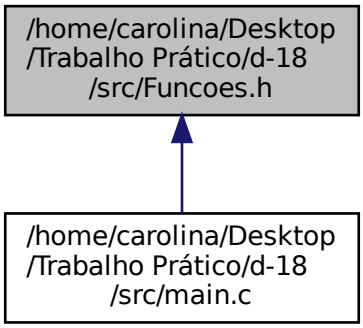
Declaração das funções.

```
#include "Dados.h"
```

Include dependency graph for Funcoes.h:



This graph shows which files directly or indirectly include this file:



Functions

- void MostraAjuda ()

  Exibe informações de ajuda sobre o programa.
- bool ExportaDadosPacientes (char nomeFicheiro[ ], Paciente pacientes[ ], int maximoPacientes)

  Exporta os dados dos pacientes para um arquivo.
- bool LeDadosPacientes (char separador, char fileName[ ], Paciente ∗dados, int maximoplanos)

  Lê os dados dos pacientes de um arquivo.
- bool ExportaDadosDieta (char fileName[ ], Dieta dietas[ ], int maximodietas)

  Exporta os dados das dietas para um arquivo CSV ou TXT.
- bool LeDadosDieta (char separador, char fileName[ ], Dieta ∗dietas, int maximodietas)

  Lê os dados das dietas de um arquivo CSV ou TXT.
- bool ExportaDadosPlanoNutri (char fileName[ ], PlanoNutri plano[ ], int maximodietas)

  Exporta os dados do plano nutricional para um arquivo CSV ou TXT.
- bool LeDadosPlanoNutri (char separador, char fileName[ ], PlanoNutri ∗nutri, int maximodados)

  Lê os dados dos planos nutricionais de um arquivo e armazena-os em um array de estruturas PlanoNutri.
- int NumPacientesUltrapassamCal (PlanoNutri plano[ ], Dieta dietas[ ], int maximopacientes)

  Conta o número de pacientes cujo consumo de calorias ultrapassa o limite definido no plano nutricional.
- void IdentificaForaIntervalo (PlanoNutri plano[ ], Dieta dietas[ ], Paciente dados[ ], NaoCumpPaciente dadospacientes[ ], int maxpaciente)

  Identifica os pacientes cujo consumo de calorias está fora do intervalo definido no plano nutricional.
- void OrdenaPacientesForaIntervalo (NaoCumpPaciente dadospacientes[ ], int maxpaciente)

  Ordena os pacientes fora do intervalo com base no número do paciente.
- void ListaPlanoNutricional (PlanoNutri planos[ ], int maxplanos, int numpaciente, int refeicao, char data↩ Inicio[ ], char dataFim[ ])

  Lista o plano nutricional para um paciente em uma refeição específica durante um determinado período.
- void CalcularMediasRefeicoes (Dieta dietas[ ], MediaPaciente dados[ ], int maxdietas, int maxpacientes, char dataInicio[ ], char dataFim[ ])

  Calcula as médias de calorias consumidas em diferentes refeições para cada paciente.
- void GerarTabelaRefeicoes (Paciente pacientes[ ], Dieta dietas[ ], PlanoNutri planos[ ], int maxPacientes, int maxDietas, int maxPlanos)

  Gera uma tabela de refeições com informações sobre dietas e planos nutricionais.

5.2.1 Detailed Description

Declaração das funções.

Author

Pedro Ribeiro, Ricardo Fernandes, Carolina Branco (a27960@alunos.ipca.pt a279861@alunos.↩ opca.pt a27983@alunos.ipca.pt)

Version

0.1

Date

2023-12-27

Copyright

Copyright (c) 2023

5.2.2 Function Documentation

5.2.2.1 CalcularMediasRefeicoes()

```
void CalcularMediasRefeicoes (
            Dieta dietas[],
            MediaPaciente dados[],
            int maxdietas,
            int maxpacientes,
            char dataInicio[],
            char dataFim[] )
```

Calcula as médias de calorias consumidas em diferentes refeições para cada paciente.

Parameters

dietas	Um array de estruturas Dieta contendo informações sobre as dietas.
dados	Um array de estruturas MediaPaciente para armazenar os resultados das médias.
maxdietas	O número máximo de elementos no array de dietas.
maxpacientes	O número máximo de pacientes.
dataInicio	A data de início do período desejado para o cálculo das médias.
dataFim	A data de fim do período desejado para o cálculo das médias.

```
314
                                    {
315
316     for (int i = 0; i < maxpacientes; i++) {
317             double auxcontadorPequenoAlmoco = 0.0;
```

```
318             double auxcontadorAlmoco = 0.0;
319             double auxcontadorJantar = 0.0;
320             int numPequenoAlmoco = 0;
321             int numAlmoco = 0;
322             int numJantar = 0;
323 
324     for (int j = 0; j < maxdietas; j++) {
325                 if (strcmp(dietas[j].datainicio, dataInicio) <= 0 && strcmp(dietas[j].datafim, dataFim)
    >= 0 &&
326                     dietas[j].numpaciente == i + 1) {
327 
328                     switch (dietas[j].tp) {
329                         case 0: // Pequeno Almoço
330                             auxcontadorPequenoAlmoco += dietas[j].calorias;
331                             numPequenoAlmoco++;
332                             break;
333                         case 1: // Almoço
334                             auxcontadorAlmoco += dietas[j].calorias;
335                             numAlmoco++;
336                             break;
337                         case 2: // Jantar
338                             auxcontadorJantar += dietas[j].calorias;
339                             numJantar++;
340                             break;
341             }
342         }
343     }
344 
345     dados[i].numpaciente = i + 1;
346     dados[i].mediapequenoalmoco = (numPequenoAlmoco > 0) ? auxcontadorPequenoAlmoco / numPequenoAlmoco :
    0;
347     dados[i].mediaalmoco = (numAlmoco > 0) ? auxcontadorAlmoco / numAlmoco : 0;
348     dados[i].mediajantar = (numJantar > 0) ? auxcontadorJantar / numJantar : 0;
349         }
350     }
```

5.2.2.2 ExportaDadosDieta()

```
bool ExportaDadosDieta (
              char fileName[],
              Dieta dietas[],
              int maximodietas )
```

Exporta os dados das dietas para um arquivo CSV ou TXT.

Esta função recebe um nome de um arquivo e um array de structs Dieta, e exporta os dados das dietas para o arquivo.

Parameters

fileName	é o nome do arquivo que vai ser feita a exportação.
dietas	é o array de structs Dieta contendo os dados a serem exportados.
maximodietas	Número máximo de dietas no array.

Returns

Retorna true se a exportação for bem sucedida, false caso contrário.

```
46                                                                                  {
47     FILE* fp;
48     fp = fopen(fileName, "w"); //Abre o ficheiro para escrita, usado o caracter "w" para apagar o que
       estiver escrito e escrever depois o pedido
49     if (fp == NULL) return false; //Caso não seja possível abrir o ficheiro retorna false
50     for (int i = 0; i < maximodietas; i++) {
51             // O ciclo for itera sobre o array dietas e vai escrevendo o mesmo no ficheiro, no formato
       CSV.
52             fprintf(fp, "%d;%s;%s;%d;%s;%d\n", dietas[i].numpaciente, dietas[i].datainicio,
       dietas[i].datafim, dietas[i].tp, dietas[i].alimento, dietas[i].calorias);
```

```
53
54         }
55     fclose(fp);
56     return true;
57     //Fecha o ficheiro e devolve true
58 }
```

5.2.2.3 ExportaDadosPacientes()

```
bool ExportaDadosPacientes (
             char nomeFicheiro[],
             Paciente pacientes[],
             int maximoPacientes )
```

Exporta os dados dos pacientes para um arquivo.

Esta função exporta os dados dos pacientes para um arquivo no formato CSV ou TXT.

Parameters

nomeFicheiro	é o nome do arquivo para exportar os dados.
pacientes	é o array de structs Paciente que contem os dados dos pacientes.
maximoPacientes	é o número máximo de pacientes no array.

Returns

Retorna true se a exportação for bem sucedida, caso contrário, retorna false.

```
30                                                                                                    {
31     FILE* fp;
32     fp = fopen(nomeFicheiro, "w");  //Abre o ficheiro para escrita, usado o caracter "w" para apagar o
       que estiver escrito e escrever depois o pedido
33     if (fp == NULL) return 0; //Caso não seja possível abrir o ficheiro retorna false
34         for (int i = 0; i < maximoPacientes; i++) {
35             // O ciclo for itera sobre o array pacientes e vai escrevendo o mesmo no ficheiro, no formato
       CSV.
36             fprintf(fp, "%d;%s;%d\n", pacientes[i].numpaciente, pacientes[i].nome,
       pacientes[i].telefone);
37         }
38     fclose(fp);
39     return true;
40     //Fecha o ficheiro e devolve true
41 }
```

5.2.2.4 ExportaDadosPlanoNutri()

```
bool ExportaDadosPlanoNutri (
             char fileName[],
             PlanoNutri plano[],
             int maximodietas )
```

Exporta os dados do plano nutricional para um arquivo CSV ou TXT.

Esta função recebe um nome de arquivo e exporta os dados do plano nutricional para o arquivo no formato CSV ou TXT.

Parameters

fileName	é o nome do arquivo para exportação.
plano	é o array de structs PlanoNutri contendo os dados a serem exportados.
maximodietas	é o número máximo de planos nutricionais no array.

Returns

Retorna true se a exportação for bem sucedida, false caso contrário.

```
119                                                                                                 {
120     FILE* fp;
121     fp = fopen(fileName, "w");
122     if (fp == NULL) return false;
123     for (int i = 0; i < maximodietas; i++) {
124         fprintf(fp, "%d;%s;%s;%d;%d;%d\n", plano[i].numpaciente, plano[i].datainicio, plano[i].datafim,
        plano[i].tp, plano[i].caloriasMin, plano[i].caloriasMax);
125     }
126     fclose(fp);
127     return true;
128 }
```

5.2.2.5 GerarTabelaRefeicoes()

```
void GerarTabelaRefeicoes (
             Paciente pacientes[],
             Dieta dietas[],
             PlanoNutri planos[],
             int maxPacientes,
             int maxDietas,
             int maxPlanos )
```

Gera uma tabela de refeições com informações sobre dietas e planos nutricionais.

Esta função imprime uma tabela que contem informações sobre as refeições, incluindo dados do paciente, tipo de refeição, datas de início e término, calorias mínimas e máximas do plano nutricional, e consumo de calorias.

Parameters

pacientes	é o array de structs Paciente que contem os dados dos pacientes.
dietas	é o array de structs Dieta que contem os dados das dietas.
planos	é o array de structs PlanoNutri que contem os dados dos planos nutricionais.
maxPacientes	é o número máximo de pacientes no array.
maxDietas	é o número máximo de dietas no array.
maxPlanos	é o número máximo de planos nutricionais no array.

```
101                                 {
102     printf("%s\t %-10s\t %-20s %-10s\t %-15s\t %-15s\t %-15s\t %-15s\n\n",
103     "NP", "Paciente", "Tipo Refeição", "Data Início", "Data Fim", "Calorias Mínimo", "Calorias Máximo",
        "Consumo");
104
105     for (int i = 0; i < maxDietas; i++) {
106         char aux[N];
107         if (planos[i].tp == 0) {
108             strcpy(aux, "peq. almoço");
109         }
110         if (planos[i].tp == 1) {
111             strcpy(aux, "almoço");
112         }
```

```
113         if (planos[i].tp == 2) {
114             strcpy(aux, "jantar");
115         }
116 
117         //Criação de uma variável e colocação temporária da string correspondente a cada índice para ser
        imprimido na consola o valor em string
118 
119         printf("%04d\t %-10s\t %-20s %-10s\t %-15s\t %-15d\t %-15d\t %-15d\n",
120         dietas[i].numpaciente,
121         pacientes[dietas[i].numpaciente - 1].nome,
122         aux,
123         planos[i].datainicio,
124         planos[i].datafim,
125         planos[dietas[i].numpaciente - 1].caloriasMin,
126         planos[dietas[i].numpaciente - 1].caloriasMax,
127         dietas[i].calorias);
128     }
129 }
```

5.2.2.6 IdentificaForaIntervalo()

```
void IdentificaForaIntervalo (
              PlanoNutri plano[],
              Dieta dietas[],
              Paciente dados[],
              NaoCumpPaciente dadospacientes[],
              int maxpaciente )
```

Identifica os pacientes cujo consumo de calorias está fora do intervalo definido no plano nutricional.

Esta função percorre os arrays de planos nutricionais, dietas e dados dos pacientes, identificando os pacientes cujo consumo de calorias está fora do intervalo definido pelo plano nutricional. Os pacientes identificados são armazenados na struct NaoCumpPaciente.

Parameters

plano	é o array de structs PlanoNutri contendo os dados dos planos nutricionais.
dietas	é o array de structs Dieta contendo os dados das dietas.
dados	é o array de structs Paciente contendo os dados dos pacientes.
dadospacientes	é o array de structs NaoCumpPaciente que armazenará os dados dos pacientes que não cumprem.
maxpaciente	é o número máximo de pacientes nos arrays.

```
215 
                                    {
216     int num = 0;
217 
218     for (int i = 0; i < maxpaciente; i++) {
219     // Verifica se o consumo de calorias está fora do intervalo definido no plano
220     if (dietas[i].calorias > plano[i].caloriasMax || dietas[i].calorias < plano[i].caloriasMin) {
221         // Armazena os dados dos pacientes que não cumprem as condições
222         dadospacientes[num].numpaciente = dados[i].numpaciente;
223         strcpy(dadospacientes[num].nome, dados[i].nome);
224         dadospacientes[num].telefone = dados[i].telefone;
225         num++;
226         }
227     }
228 }
```

5.2.2.7 LeDadosDieta()

```
bool LeDadosDieta (
```

```
            char separador,
            char fileName[],
            Dieta * dietas,
            int maximodietas )
```

Lê os dados das dietas de um arquivo CSV ou TXT.

Esta função recebe um nome de arquivo e lê os dados das dietas do arquivo no formato CSV ou TXT. Os dados lidos são armazenados em um array de structs Dieta.

Parameters

separador	
fileName	é o nome do arquivo para leitura.
dietas	é o array de structs Dieta para armazenar os dados lidos.
maximodietas	é p número máximo de dietas no array.

Returns

Retorna true se a leitura for bem sucedida, false caso contrário.

```
78                                                                                  {
79      FILE * fp;
80      fp = fopen (fileName,"r");
81      if (fp == NULL) {
82      return false; // Retorna false se o apontador do ficheiro for nulo
83      }
84      if (separador == ';') {
85          for (int i = 0; i < maximodietas; i++) {
86              fscanf(fp, "%d;%99[^;];%99[^;];%d;%99[^;];%d", &dietas[i].numpaciente, dietas[i].datainicio,
    dietas[i].datafim, &dietas[i].tp, dietas[i].alimento, &dietas[i].calorias);
87              //Para verifcar o que foi lido:
88              //printf("%d;%s;%s;%d;%s;%d\n", dietas[i].numpaciente, dietas[i].datainicio,
    dietas[i].datafim, dietas[i].tp, dietas[i].alimento, dietas[i].calorias);
89
90          }
91      }
92      if (separador == '\t') {
93          for (int i = 0; i < maximodietas; i++) {
94              fscanf(fp, "%d\t%99[^\t]\t%99[^\t]\t%d\t%99[^\t]\t%d", &dietas[i].numpaciente,
    dietas[i].datainicio, dietas[i].datafim, &dietas[i].tp, dietas[i].alimento, &dietas[i].calorias);
95              //Para verifcar o que foi lido:
96              //printf("%d\t%s\t%s\t%d\t%s\t%d\n", dietas[i].numpaciente, dietas[i].datainicio,
    dietas[i].datafim, dietas[i].tp, dietas[i].alimento, dietas[i].calorias);
97          }
98      }
99      fclose(fp);
100      return true;
101      }
```

5.2.2.8 LeDadosPacientes()

```
bool LeDadosPacientes (
            char separador,
            char fileName[],
            Paciente * dados,
            int maximoplanos )
```

Lê os dados dos pacientes de um arquivo.

Esta função lê os dados dos pacientes de um arquivo no formato CSV.

Parameters

nomeFicheiro	é o nome do arquivo para ler os dados dos pacientes.
pacientes	é o array de structs Paciente para armazenar os dados lidos.
maximoPacientes	é o número máximo de pacientes no array.

Returns

Retorna true se a leitura for bem sucedida, caso contrário, retorna false.

```
58                                                                                                  {
59
60      FILE* fp;
61      fp = fopen(fileName, "r");
62      if (fp == NULL) {
63          return false;
64      }
65      if (separador == ';') {
66          for (int i = 0; i < maximoplanos; i++) {
67              fscanf(fp, "%d;%99[^;];%d", &dados[i].numpaciente, dados[i].nome, &dados[i].telefone);
68              //Para verifcar o que foi lido:
69              //printf("dados lidos :%d;%s;%d\n", dados[i].numpaciente, dados[i].nome, dados[i].telefone);
70          }
71      }
72      if (separador == '\t') {
73          for (int i = 0; i < maximoplanos; i++) {
74              fscanf(fp, "%d\t%99[^\t]\t%d", &dados[i].numpaciente, dados[i].nome, &dados[i].telefone);
75              //Para verifcar o que foi lido:
76              //printf("lidos : %d\t%s\t%d\n", dados[i].numpaciente, dados[i].nome, dados[i].telefone);
77          }
78      }
79      fclose(fp);
80      return true;
81  }
```

5.2.2.9 LeDadosPlanoNutri()

```
bool LeDadosPlanoNutri (
              char separador,
              char fileName[],
              PlanoNutri * nutri,
              int maximodados )
```

Lê os dados dos planos nutricionais de um arquivo e armazena-os em um array de estruturas PlanoNutri.

Esta função recebe um nome de arquivo e lê os dados das dietas do arquivo no formato CSV ou TXT. Os dados lidos são armazenados em um array de structs PlanoNutri.

Parameters

fileName	é o nome do arquivo a ser lido.
plano	é o array de estruturas PlanoNutri para armazenar os dados lidos.
maximodietas	é o número máximo de dietas a serem lidas do arquivo.

Returns

Retorna true se a leitura for bem sucedida, false caso contrário.

```
143                                                                                                 {
144     FILE * fp;
```

```
145     fp = fopen(fileName,"r");
146     if (fp == NULL) {
147         return false; // Retorna um dado vazio se o apontador do ficheiro for nulo
148     }
149 
150     if (separador == ';') {
151         for (int i = 0; i < maximodados; i++) {
152             fscanf(fp, "%d;%99[^;];%99[^;];%d;%d;%d", &nutri[i].numpaciente, nutri[i].datainicio,
    nutri[i].datafim, &nutri[i].tp, &nutri[i].caloriasMin, &nutri[i].caloriasMax);
153             //Para verifcar o que foi lido:
154             //printf("%d;%s;%s;%d;%d;%d\n", nutri[i].numpaciente, nutri[i].datainicio, nutri[i].datafim,
    nutri[i].tp, nutri[i].caloriasMin, nutri[i].caloriasMax);
155         }
156     }
157 
158     if (separador == '\t') {
159         for (int i = 0; i < maximodados; i++) {
160             fscanf(fp, "%d\t%99[^\t]\t%99[^\t]\t%d\t%d\t%d", &nutri[i].numpaciente, nutri[i].datainicio,
    nutri[i].datafim, &nutri[i].tp, &nutri[i].caloriasMin, &nutri[i].caloriasMax);
161             //Para verifcar o que foi lido:
162             //printf("%d\t%s\t%s\t%d\t%d\t%d\n", nutri[i].numpaciente, nutri[i].datainicio,
    nutri[i].datafim, nutri[i].tp, nutri[i].caloriasMin, nutri[i].caloriasMax);
163         }
164     }
165     fclose(fp);
166     return true;
167 }
```

5.2.2.10 ListaPlanoNutricional()

```
void ListaPlanoNutricional (
              PlanoNutri planos[],
              int maxplanos,
              int numpaciente,
              int refeicao,
              char dataInicio[],
              char dataFim[] )
```

Lista o plano nutricional para um paciente em uma refeição específica durante um determinado período.

Esta função imprime na tela o plano nutricional para um paciente em uma refeição específica durante um período definido.

Parameters

planos	é o array de structs PlanoNutri que contem os dados dos planos nutricionais.
maxplanos	é o número máximo de planos nutricionais no array.
numpaciente	é o número do paciente para o qual o plano nutricional será listado.
refeicao	é o indice da refeição (0 para 'peq. almoço', 1 para 'almoço', 2 para 'jantar').
dataInicio	é a data de início do período desejado no formato "DD/MM/AAAA".
dataFim	é a data de término do período desejado no formato "DD/MM/AAAA".

```
271 
                            {
272 
273 char aux[N];
274     if (refeicao == 0) {
275         strcpy(aux, "peq. almoço");
276     }
277     if (refeicao == 1) {
278         strcpy(aux, "almoço");
279     }
280     if (refeicao == 2) {
281         strcpy(aux, "jantar");
282     }
283 
```

```
284     //Criação de uma variável e colocação da string correspondente a cada índice para ser imprimido na
        consola o valor em string
285
286     printf("Plano Nutricional para o Paciente %d na Refeicao '%s' no periodo de %s a %s:\n\n",
        numpaciente, aux, dataInicio, dataFim);
287
288     for (int i = 0; i < maxplanos; i++) {
289         if (planos[i].numpaciente == numpaciente && planos[i].tp == refeicao &&
        strcmp(planos[i].datainicio, dataInicio) <= 0 && strcmp(planos[i].datafim, dataFim) >= 0) {
290             printf("O paciente de número: %d\n", planos[i].numpaciente);
291             printf("O plano dura de %s a %s\n", planos[i].datainicio, planos[i].datafim);
292             printf("Na refeição %s deve ingerir:\n", aux);
293             printf("Calorias Mínimas: %d\n", planos[i].caloriasMin);
294             printf("Calorias Máximas: %d\n\n", planos[i].caloriasMax);
295         }
296     }
297 }
```

5.2.2.11 MostraAjuda()

```
void MostraAjuda ( )
```

Exibe informações de ajuda sobre o programa.

Esta função imprime mensagens de ajuda na consola, explicando o funcionamento do programa. Inclui instruções sobre como utilizar o programa.

```
24                  {
25     printf("Para compilar o programa utilize uma das seguintes opções:\n\n");
26     printf("-ajuda: É utilizada para fornecer ajuda ao utilizador sobre os comandos a utilizar\n");
27     printf("-tab : É utilizada para compilar o programa com ficheiros passados por argumento com os dados
       separados por tabs\n");
28 }
```

5.2.2.12 NumPacientesUltrapassamCal()

```
int NumPacientesUltrapassamCal (
            PlanoNutri plano[],
            Dieta dietas[],
            int maximopacientes )
```

Conta o número de pacientes cujo consumo de calorias ultrapassa o limite definido no plano nutricional.

Esta função percorre os arrays de planos nutricionais e dietas e conta quantos pacientes ultrapassaram o limite de calorias definido no plano nutricional para a respectiva refeição.

Parameters

plano	é o array de structs PlanoNutri contendo os dados dos planos nutricionais.
dietas	é o array de structs Dieta contendo os dados das dietas.
maximopacientes	é o número máximo de pacientes nos arrays.

Returns

Retorna o número de pacientes cujo consumo de calorias ultrapassa o limite definido no plano nutricional.

```
185                                                                                 {
```

```
186     int contador = 0;
187         for (int i = 0; i < maximopacientes; i++) {  // Itera sobre o array de pacientes
188             if (dietas[i].calorias > plano[i].caloriasMax) {
189                 contador++; // Se o consumo de calorias na dieta atual excede o limite definido no plano
    é somada 1 unidade à variável contador
190
191         }
192     }
193 return contador;
194     // Retorna o número total de pacientes cujo consumo de calorias ultrapassa o limite
195 }
```

5.2.2.13 OrdenaPacientesForaIntervalo()

```
void OrdenaPacientesForaIntervalo (
            NaoCumpPaciente dadospacientes[],
            int maxpaciente )
```

Ordena os pacientes fora do intervalo com base no número do paciente.

Esta função ordena o array de pacientes fora do intervalo na struct NaoCumpPaciente com base no número do paciente em ordem decrescente.

Parameters

dadospacientes	é o array de structs NaoCumpPaciente a ser ordenado.
maxpaciente	é o número máximo de pacientes no array.

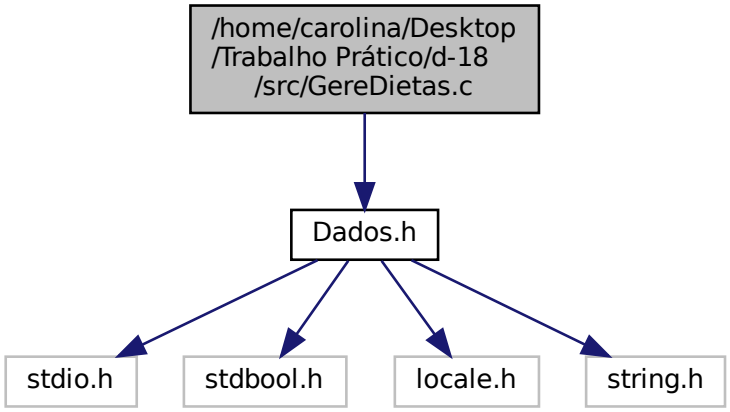
```
241                                                                                 {
242     for (int i = 0; i < maxpaciente - 1; i++) {
243         for (int j = i + 1; j < maxpaciente; j++) {
244             if (dadospacientes[i].numpaciente < dadospacientes[j].numpaciente) {  // Compara o número
    dos pacientes para determinar a ordem
245             NaoCumpPaciente temp = dadospacientes[i];
246             dadospacientes[i] = dadospacientes[j];
247             dadospacientes[j] = temp;
248             // Troca os elementos se estiverem fora de ordem
249             }
250         }
251     }
252 }
```

5.3 /home/carolina/Desktop/Trabalho Prático/d-18/src/GereDietas.c File Reference

Leitura, escrita e manipulação de informações relacionadas com dietas e planos nutricionais.

```
#include "Dados.h"
```

Include dependency graph for GereDietas.c:



Functions

- void MostraAjuda ()

  Exibe informações de ajuda sobre o programa.
- bool ExportaDadosDieta (char fileName[ ], Dieta dietas[ ], int maximodietas)

  Exporta os dados das dietas para um arquivo CSV ou TXT.
- bool LeDadosDieta (char separador, char fileName[ ], Dieta ∗dietas, int maximodietas)

  Lê os dados das dietas de um arquivo CSV ou TXT.
- bool ExportaDadosPlanoNutri (char fileName[ ], PlanoNutri plano[ ], int maximodietas)

  Exporta os dados do plano nutricional para um arquivo CSV ou TXT.
- bool LeDadosPlanoNutri (char separador, char fileName[ ], PlanoNutri ∗nutri, int maximodados)

  Lê os dados dos planos nutricionais de um arquivo e armazena-os em um array de estruturas PlanoNutri.
- int NumPacientesUltrapassamCal (PlanoNutri plano[ ], Dieta dietas[ ], int maximopacientes)

  Conta o número de pacientes cujo consumo de calorias ultrapassa o limite definido no plano nutricional.
- void IdentificaForaIntervalo (PlanoNutri plano[ ], Dieta dietas[ ], Paciente dados[ ], NaoCumpPaciente dadospacientes[ ], int maxpaciente)

  Identifica os pacientes cujo consumo de calorias está fora do intervalo definido no plano nutricional.
- void OrdenaPacientesForaIntervalo (NaoCumpPaciente dadospacientes[ ], int maxpaciente)

  Ordena os pacientes fora do intervalo com base no número do paciente.
- void ListaPlanoNutricional (PlanoNutri planos[ ], int maxplanos, int numpaciente, int refeicao, char data↩ Inicio[ ], char dataFim[ ])

  Lista o plano nutricional para um paciente em uma refeição específica durante um determinado período.
- void CalcularMediasRefeicoes (Dieta dietas[ ], MediaPaciente dados[ ], int maxdietas, int maxpacientes, char dataInicio[ ], char dataFim[ ])

  Calcula as médias de calorias consumidas em diferentes refeições para cada paciente.

5.3.1 Detailed Description

Leitura, escrita e manipulação de informações relacionadas com dietas e planos nutricionais.

Author

Pedro Ribeiro, Ricardo Fernandes, Carolina Branco (a27960@alunos.ipca.pt a279861@alunos.ipca.pt a27983@alunos.ipca.pt)

Version

0.1

Date

2023-12-27

Copyright

Copyright (c) 2023

5.3.2 Function Documentation

5.3.2.1 CalcularMediasRefeicoes()

```
void CalcularMediasRefeicoes (
            Dieta dietas[],
            MediaPaciente dados[],
            int maxdietas,
            int maxpacientes,
            char dataInicio[],
            char dataFim[] )
```

Calcula as médias de calorias consumidas em diferentes refeições para cada paciente.

Parameters

dietas	Um array de estruturas Dieta contendo informações sobre as dietas.
dados	Um array de estruturas MediaPaciente para armazenar os resultados das médias.
maxdietas	O número máximo de elementos no array de dietas.
maxpacientes	O número máximo de pacientes.
dataInicio	A data de início do período desejado para o cálculo das médias.
dataFim	A data de fim do período desejado para o cálculo das médias.

```
314
                                    {
315
316     for (int i = 0; i < maxpacientes; i++) {
317             double auxcontadorPequenoAlmoco = 0.0;
```

```
318             double auxcontadorAlmoco = 0.0;
319             double auxcontadorJantar = 0.0;
320             int numPequenoAlmoco = 0;
321             int numAlmoco = 0;
322             int numJantar = 0;
323 
324     for (int j = 0; j < maxdietas; j++) {
325                 if (strcmp(dietas[j].datainicio, dataInicio) <= 0 && strcmp(dietas[j].datafim, dataFim)
    >= 0 &&
326                     dietas[j].numpaciente == i + 1) {
327 
328                     switch (dietas[j].tp) {
329                         case 0: // Pequeno Almoço
330                             auxcontadorPequenoAlmoco += dietas[j].calorias;
331                             numPequenoAlmoco++;
332                             break;
333                         case 1: // Almoço
334                             auxcontadorAlmoco += dietas[j].calorias;
335                             numAlmoco++;
336                             break;
337                         case 2: // Jantar
338                             auxcontadorJantar += dietas[j].calorias;
339                             numJantar++;
340                             break;
341             }
342         }
343     }
344 
345     dados[i].numpaciente = i + 1;
346     dados[i].mediapequenoalmoco = (numPequenoAlmoco > 0) ? auxcontadorPequenoAlmoco / numPequenoAlmoco :
    0;
347     dados[i].mediaalmoco = (numAlmoco > 0) ? auxcontadorAlmoco / numAlmoco : 0;
348     dados[i].mediajantar = (numJantar > 0) ? auxcontadorJantar / numJantar : 0;
349         }
350     }
```

5.3.2.2 ExportaDadosDieta()

```
bool ExportaDadosDieta (
              char fileName[],
              Dieta dietas[],
              int maximodietas )
```

Exporta os dados das dietas para um arquivo CSV ou TXT.

Esta função recebe um nome de um arquivo e um array de structs Dieta, e exporta os dados das dietas para o arquivo.

Parameters

fileName	é o nome do arquivo que vai ser feita a exportação.
dietas	é o array de structs Dieta contendo os dados a serem exportados.
maximodietas	Número máximo de dietas no array.

Returns

Retorna true se a exportação for bem sucedida, false caso contrário.

```
46                                                                                  {
47     FILE* fp;
48     fp = fopen(fileName, "w"); //Abre o ficheiro para escrita, usado o caracter "w" para apagar o que
       estiver escrito e escrever depois o pedido
49     if (fp == NULL) return false; //Caso não seja possível abrir o ficheiro retorna false
50     for (int i = 0; i < maximodietas; i++) {
51             // O ciclo for itera sobre o array dietas e vai escrevendo o mesmo no ficheiro, no formato
       CSV.
52             fprintf(fp, "%d;%s;%s;%d;%s;%d\n", dietas[i].numpaciente, dietas[i].datainicio,
       dietas[i].datafim, dietas[i].tp, dietas[i].alimento, dietas[i].calorias);
```

```
53
54         }
55     fclose(fp);
56     return true;
57     //Fecha o ficheiro e devolve true
58 }
```

5.3.2.3 ExportaDadosPlanoNutri()

```
bool ExportaDadosPlanoNutri (
             char fileName[],
             PlanoNutri plano[],
             int maximodietas )
```

Exporta os dados do plano nutricional para um arquivo CSV ou TXT.

Esta função recebe um nome de arquivo e exporta os dados do plano nutricional para o arquivo no formato CSV ou TXT.

Parameters

fileName	é o nome do arquivo para exportação.
plano	é o array de structs PlanoNutri contendo os dados a serem exportados.
maximodietas	é o número máximo de planos nutricionais no array.

Returns

Retorna true se a exportação for bem sucedida, false caso contrário.

```
119                                                                                    {
120     FILE* fp;
121     fp = fopen(fileName, "w");
122     if (fp == NULL) return false;
123     for (int i = 0; i < maximodietas; i++) {
124         fprintf(fp, "%d;%s;%s;%d;%d;%d\n", plano[i].numpaciente, plano[i].datainicio, plano[i].datafim,
       plano[i].tp, plano[i].caloriasMin, plano[i].caloriasMax);
125     }
126     fclose(fp);
127     return true;
128 }
```

5.3.2.4 IdentificaForaIntervalo()

```
void IdentificaForaIntervalo (
             PlanoNutri plano[],
             Dieta dietas[],
             Paciente dados[],
             NaoCumpPaciente dadospacientes[],
             int maxpaciente )
```

Identifica os pacientes cujo consumo de calorias está fora do intervalo definido no plano nutricional.

Esta função percorre os arrays de planos nutricionais, dietas e dados dos pacientes, identificando os pacientes cujo consumo de calorias está fora do intervalo definido pelo plano nutricional. Os pacientes identificados são armazenados na struct NaoCumpPaciente.

Parameters

plano	é o array de structs PlanoNutri contendo os dados dos planos nutricionais.
dietas	é o array de structs Dieta contendo os dados das dietas.
dados	é o array de structs Paciente contendo os dados dos pacientes.
dadospacientes	é o array de structs NaoCumpPaciente que armazenará os dados dos pacientes que não cumprem.
maxpaciente	é o número máximo de pacientes nos arrays.

```
215                                     {
216     int num = 0;
217
218     for (int i = 0; i < maxpaciente; i++) {
219     // Verifica se o consumo de calorias está fora do intervalo definido no plano
220     if (dietas[i].calorias > plano[i].caloriasMax || dietas[i].calorias < plano[i].caloriasMin) {
221         // Armazena os dados dos pacientes que não cumprem as condições
222         dadospacientes[num].numpaciente = dados[i].numpaciente;
223         strcpy(dadospacientes[num].nome, dados[i].nome);
224         dadospacientes[num].telefone = dados[i].telefone;
225         num++;
226         }
227     }
228 }
```

5.3.2.5 LeDadosDieta()

```
bool LeDadosDieta (
              char separador,
              char fileName[],
              Dieta * dietas,
              int maximodietas )
```

Lê os dados das dietas de um arquivo CSV ou TXT.

Esta função recebe um nome de arquivo e lê os dados das dietas do arquivo no formato CSV ou TXT. Os dados lidos são armazenados em um array de structs Dieta.

Parameters

separador	
fileName	é o nome do arquivo para leitura.
dietas	é o array de structs Dieta para armazenar os dados lidos.
maximodietas	é p número máximo de dietas no array.

Returns

Retorna true se a leitura for bem sucedida, false caso contrário.

```
78                                                                                  {
79     FILE * fp;
80     fp = fopen (fileName,"r");
81     if (fp == NULL) {
82     return false; // Retorna false se o apontador do ficheiro for nulo
83     }
84     if (separador == ';') {
85         for (int i = 0; i < maximodietas; i++) {
86             fscanf(fp, "%d;%99[^;];%99[^;];%d;%99[^;];%d", &dietas[i].numpaciente, dietas[i].datainicio,
       dietas[i].datafim, &dietas[i].tp, dietas[i].alimento, &dietas[i].calorias);
87             //Para verifcar o que foi lido:
```

```
88              //printf("%d;%s;%s;%d;%s;%d\n", dietas[i].numpaciente, dietas[i].datainicio,
        dietas[i].datafim, dietas[i].tp, dietas[i].alimento, dietas[i].calorias);
89
90          }
91      }
92      if (separador == '\t') {
93          for (int i = 0; i < maximodietas; i++) {
94              fscanf(fp, "%d\t%99[^\t]\t%99[^\t]\t%d\t%99[^\t]\t%d", &dietas[i].numpaciente,
        dietas[i].datainicio, dietas[i].datafim, &dietas[i].tp, dietas[i].alimento, &dietas[i].calorias);
95              //Para verifcar o que foi lido:
96              //printf("%d\t%s\t%s\t%d\t%s\t%d\n", dietas[i].numpaciente, dietas[i].datainicio,
        dietas[i].datafim, dietas[i].tp, dietas[i].alimento, dietas[i].calorias);
97          }
98      }
99      fclose(fp);
100      return true;
101      }
```

5.3.2.6 LeDadosPlanoNutri()

```
bool LeDadosPlanoNutri (
            char separador,
            char fileName[],
            PlanoNutri * nutri,
            int maximodados )
```

Lê os dados dos planos nutricionais de um arquivo e armazena-os em um array de estruturas PlanoNutri.

Esta função recebe um nome de arquivo e lê os dados das dietas do arquivo no formato CSV ou TXT. Os dados lidos são armazenados em um array de structs PlanoNutri.

Parameters

fileName	é o nome do arquivo a ser lido.
plano	é o array de estruturas PlanoNutri para armazenar os dados lidos.
maximodietas	é o número máximo de dietas a serem lidas do arquivo.

Returns

Retorna true se a leitura for bem sucedida, false caso contrário.

```
143                                                                                          {
144      FILE * fp;
145      fp = fopen(fileName,"r");
146      if (fp == NULL) {
147          return false; // Retorna um dado vazio se o apontador do ficheiro for nulo
148      }
149
150      if (separador == ';') {
151          for (int i = 0; i < maximodados; i++) {
152              fscanf(fp, "%d;%99[^;];%99[^;];%d;%d;%d", &nutri[i].numpaciente, nutri[i].datainicio,
        nutri[i].datafim, &nutri[i].tp, &nutri[i].caloriasMin, &nutri[i].caloriasMax);
153              //Para verifcar o que foi lido:
154              //printf("%d;%s;%s;%d;%d;%d\n", nutri[i].numpaciente, nutri[i].datainicio, nutri[i].datafim,
        nutri[i].tp, nutri[i].caloriasMin, nutri[i].caloriasMax);
155          }
156      }
157
158      if (separador == '\t') {
159          for (int i = 0; i < maximodados; i++) {
160              fscanf(fp, "%d\t%99[^\t]\t%99[^\t]\t%d\t%d\t%d", &nutri[i].numpaciente, nutri[i].datainicio,
        nutri[i].datafim, &nutri[i].tp, &nutri[i].caloriasMin, &nutri[i].caloriasMax);
161              //Para verifcar o que foi lido:
162              //printf("%d\t%s\t%s\t%d\t%d\t%d\n", nutri[i].numpaciente, nutri[i].datainicio,
        nutri[i].datafim, nutri[i].tp, nutri[i].caloriasMin, nutri[i].caloriasMax);
163          }
164      }
```

```
165     fclose(fp);
166     return true;
167 }
```

5.3.2.7 ListaPlanoNutricional()

```
void ListaPlanoNutricional (
             PlanoNutri planos[],
             int maxplanos,
             int numpaciente,
             int refeicao,
             char dataInicio[],
             char dataFim[] )
```

Lista o plano nutricional para um paciente em uma refeição específica durante um determinado período.

Esta função imprime na tela o plano nutricional para um paciente em uma refeição específica durante um período definido.

Parameters

planos	é o array de structs PlanoNutri que contem os dados dos planos nutricionais.
maxplanos	é o número máximo de planos nutricionais no array.
numpaciente	é o número do paciente para o qual o plano nutricional será listado.
refeicao	é o indice da refeição (0 para 'peq. almoço', 1 para 'almoço', 2 para 'jantar').
dataInicio	é a data de início do período desejado no formato "DD/MM/AAAA".
dataFim	é a data de término do período desejado no formato "DD/MM/AAAA".

```
271
                              {
272
273 char aux[N];
274     if (refeicao == 0) {
275         strcpy(aux, "peq. almoço");
276     }
277     if (refeicao == 1) {
278         strcpy(aux, "almoço");
279     }
280     if (refeicao == 2) {
281         strcpy(aux, "jantar");
282     }
283
284     //Criação de uma variável e colocação da string correspondente a cada índice para ser imprimido na
      consola o valor em string
285
286      printf("Plano Nutricional para o Paciente %d na Refeicao '%s' no periodo de %s a %s:\n\n",
      numpaciente, aux, dataInicio, dataFim);
287
288     for (int i = 0; i < maxplanos; i++) {
289         if (planos[i].numpaciente == numpaciente && planos[i].tp == refeicao &&
      strcmp(planos[i].datainicio, dataInicio) <= 0 && strcmp(planos[i].datafim, dataFim) >= 0) {
290             printf("O paciente de número: %d\n", planos[i].numpaciente);
291             printf("O plano dura de %s a %s\n", planos[i].datainicio, planos[i].datafim);
292             printf("Na refeição %s deve ingerir:\n", aux);
293             printf("Calorias Mínimas: %d\n", planos[i].caloriasMin);
294             printf("Calorias Máximas: %d\n\n", planos[i].caloriasMax);
295         }
296     }
297 }
```

5.3.2.8 MostraAjuda()

```
void MostraAjuda ( )
```

Exibe informações de ajuda sobre o programa.

Esta função imprime mensagens de ajuda na consola, explicando o funcionamento do programa. Inclui instruções sobre como utilizar o programa.

```
24                  {
25     printf("Para compilar o programa utilize uma das seguintes opções:\n\n");
26     printf("-ajuda: É utilizada para fornecer ajuda ao utilizador sobre os comandos a utilizar\n");
27     printf("-tab : É utilizada para compilar o programa com ficheiros passados por argumento com os dados
       separados por tabs\n");
28 }
```

5.3.2.9 NumPacientesUltrapassamCal()

```
int NumPacientesUltrapassamCal (
            PlanoNutri plano[],
            Dieta dietas[],
            int maximopacientes )
```

Conta o número de pacientes cujo consumo de calorias ultrapassa o limite definido no plano nutricional.

Esta função percorre os arrays de planos nutricionais e dietas e conta quantos pacientes ultrapassaram o limite de calorias definido no plano nutricional para a respectiva refeição.

Parameters

plano	é o array de structs PlanoNutri contendo os dados dos planos nutricionais.
dietas	é o array de structs Dieta contendo os dados das dietas.
maximopacientes	é o número máximo de pacientes nos arrays.

Returns

Retorna o número de pacientes cujo consumo de calorias ultrapassa o limite definido no plano nutricional.

```
185                                                                                 {
186     int contador = 0;
187         for (int i = 0; i < maximopacientes; i++) {  // Itera sobre o array de pacientes
188             if (dietas[i].calorias > plano[i].caloriasMax) {
189                 contador++; // Se o consumo de calorias na dieta atual excede o limite definido no plano
       é somada 1 unidade à variável contador
190
191         }
192     }
193 return contador;
194     // Retorna o número total de pacientes cujo consumo de calorias ultrapassa o limite
195 }
```

5.3.2.10 OrdenaPacientesForaIntervalo()

```
void OrdenaPacientesForaIntervalo (
            NaoCumpPaciente dadospacientes[],
            int maxpaciente )
```

Ordena os pacientes fora do intervalo com base no número do paciente.

Esta função ordena o array de pacientes fora do intervalo na struct NaoCumpPaciente com base no número do paciente em ordem decrescente.

Parameters

dadospacientes	é o array de structs NaoCumpPaciente a ser ordenado.
maxpaciente	é o número máximo de pacientes no array.

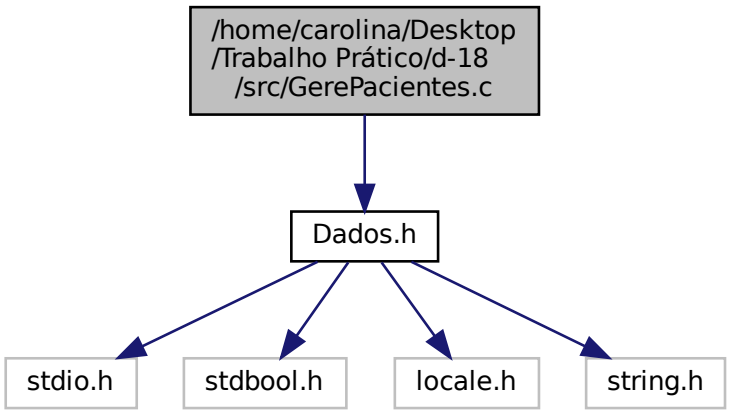
```
241                                                                                 {
242     for (int i = 0; i < maxpaciente - 1; i++) {
243         for (int j = i + 1; j < maxpaciente; j++) {
244             if (dadospacientes[i].numpaciente < dadospacientes[j].numpaciente) {  // Compara o número
        dos pacientes para determinar a ordem
245             NaoCumpPaciente temp = dadospacientes[i];
246             dadospacientes[i] = dadospacientes[j];
247             dadospacientes[j] = temp;
248             // Troca os elementos se estiverem fora de ordem
249             }
250         }
251     }
252 }
```

5.4 /home/carolina/Desktop/Trabalho Prático/d-18/src/GerePacientes.c File Reference

Leitura, escrita e manipulação de informações relacionadas a pacientes.

```
#include "Dados.h"
```

Include dependency graph for GerePacientes.c:



Functions

- bool ExportaDadosPacientes (char nomeFicheiro[ ], Paciente pacientes[ ], int maximoPacientes)

  Exporta os dados dos pacientes para um arquivo.

- bool LeDadosPacientes (char separador, char fileName[ ], Paciente ∗dados, int maximoplanos)

  Lê os dados dos pacientes de um arquivo.
- void GerarTabelaRefeicoes (Paciente pacientes[ ], Dieta dietas[ ], PlanoNutri planos[ ], int maxPacientes, int maxDietas, int maxPlanos)

  Gera uma tabela de refeições com informações sobre dietas e planos nutricionais.

5.4.1 Detailed Description

Leitura, escrita e manipulação de informações relacionadas a pacientes.

Author

Pedro Ribeiro, Ricardo Fernandes, Carolina Branco (a27960@alunos.ipca.pt a279861@alunos.↩ opca.pt a27983@alunos.ipca.pt)

Version

0.1

Date

2023-12-27

Copyright

Copyright (c) 2023

5.4.2 Function Documentation

5.4.2.1 ExportaDadosPacientes()

```
bool ExportaDadosPacientes (
            char nomeFicheiro[],
            Paciente pacientes[],
            int maximoPacientes )
```

Exporta os dados dos pacientes para um arquivo.

Esta função exporta os dados dos pacientes para um arquivo no formato CSV ou TXT.

Parameters

nomeFicheiro	é o nome do arquivo para exportar os dados.
pacientes	é o array de structs Paciente que contem os dados dos pacientes.
maximoPacientes	é o número máximo de pacientes no array.

Returns

Retorna true se a exportação for bem sucedida, caso contrário, retorna false.

```
30                                                                                  {
31     FILE* fp;
32     fp = fopen(nomeFicheiro, "w");  //Abre o ficheiro para escrita, usado o caracter "w" para apagar o
       que estiver escrito e escrever depois o pedido
33     if (fp == NULL) return 0; //Caso não seja possível abrir o ficheiro retorna false
34         for (int i = 0; i < maximoPacientes; i++) {
35             // O ciclo for itera sobre o array pacientes e vai escrevendo o mesmo no ficheiro, no formato
       CSV.
36             fprintf(fp, "%d;%s;%d\n", pacientes[i].numpaciente, pacientes[i].nome,
       pacientes[i].telefone);
37         }
38     fclose(fp);
39     return true;
40     //Fecha o ficheiro e devolve true
41 }
```

5.4.2.2 GerarTabelaRefeicoes()

```
void GerarTabelaRefeicoes (
            Paciente pacientes[],
            Dieta dietas[],
            PlanoNutri planos[],
            int maxPacientes,
            int maxDietas,
            int maxPlanos )
```

Gera uma tabela de refeições com informações sobre dietas e planos nutricionais.

Esta função imprime uma tabela que contem informações sobre as refeições, incluindo dados do paciente, tipo de refeição, datas de início e término, calorias mínimas e máximas do plano nutricional, e consumo de calorias.

Parameters

pacientes	é o array de structs Paciente que contem os dados dos pacientes.
dietas	é o array de structs Dieta que contem os dados das dietas.
planos	é o array de structs PlanoNutri que contem os dados dos planos nutricionais.
maxPacientes	é o número máximo de pacientes no array.
maxDietas	é o número máximo de dietas no array.
maxPlanos	é o número máximo de planos nutricionais no array.

```
101
                                    {
102     printf("%s\t %-10s\t %-20s %-10s\t %-15s\t %-15s\t %-15s\t %-15s\n\n",
103     "NP", "Paciente", "Tipo Refeição", "Data Início", "Data Fim", "Calorias Mínimo", "Calorias Máximo",
       "Consumo");
104
105     for (int i = 0; i < maxDietas; i++) {
106         char aux[N];
107         if (planos[i].tp == 0) {
108             strcpy(aux, "peq. almoço");
109         }
110         if (planos[i].tp == 1) {
111             strcpy(aux, "almoço");
112         }
113         if (planos[i].tp == 2) {
114             strcpy(aux, "jantar");
115         }
116
117         //Criação de uma variável e colocação temporária da string correspondente a cada índice para ser
       imprimido na consola o valor em string
118
119         printf("%04d\t %-10s\t %-20s %-10s\t %-15s\t %-15d\t %-15d\t %-15d\n",
```

```
120         dietas[i].numpaciente,
121         pacientes[dietas[i].numpaciente - 1].nome,
122         aux,
123         planos[i].datainicio,
124         planos[i].datafim,
125         planos[dietas[i].numpaciente - 1].caloriasMin,
126         planos[dietas[i].numpaciente - 1].caloriasMax,
127         dietas[i].calorias);
128     }
129 }
```

5.4.2.3 LeDadosPacientes()

```
bool LeDadosPacientes (
              char separador,
              char fileName[],
              Paciente * dados,
              int maximoplanos )
```

Lê os dados dos pacientes de um arquivo.

Esta função lê os dados dos pacientes de um arquivo no formato CSV.

Parameters

nomeFicheiro	é o nome do arquivo para ler os dados dos pacientes.
pacientes	é o array de structs Paciente para armazenar os dados lidos.
maximoPacientes	é o número máximo de pacientes no array.

Returns

Retorna true se a leitura for bem sucedida, caso contrário, retorna false.

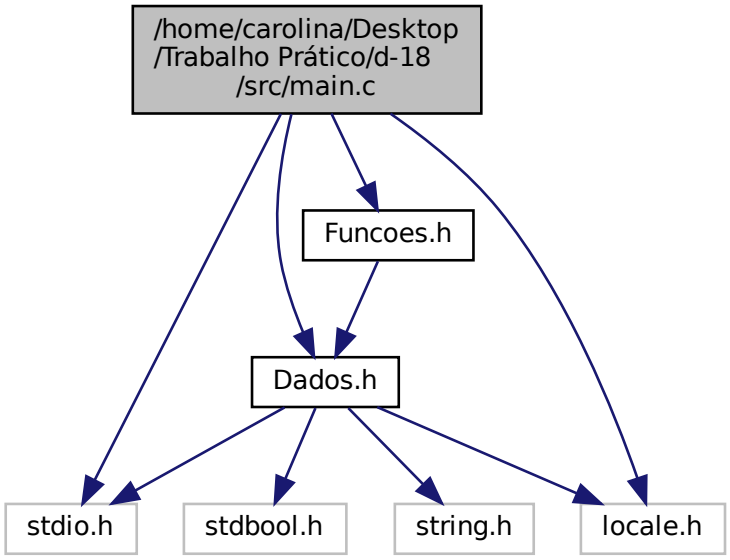
```
58                                                                                                  {
59
60     FILE* fp;
61     fp = fopen(fileName, "r");
62     if (fp == NULL) {
63         return false;
64     }
65     if (separador == ';') {
66         for (int i = 0; i < maximoplanos; i++) {
67             fscanf(fp, "%d;%99[^;];%d", &dados[i].numpaciente, dados[i].nome, &dados[i].telefone);
68             //Para verifcar o que foi lido:
69             //printf("dados lidos :%d;%s;%d\n", dados[i].numpaciente, dados[i].nome, dados[i].telefone);
70         }
71     }
72     if (separador == '\t') {
73         for (int i = 0; i < maximoplanos; i++) {
74             fscanf(fp, "%d\t%99[^\t]\t%d", &dados[i].numpaciente, dados[i].nome, &dados[i].telefone);
75             //Para verifcar o que foi lido:
76             //printf("lidos : %d\t%s\t%d\n", dados[i].numpaciente, dados[i].nome, dados[i].telefone);
77         }
78     }
79     fclose(fp);
80     return true;
81 }
```

5.5 /home/carolina/Desktop/Trabalho Prático/d-18/src/main.c File Reference

Local de implementação das funções.

```
#include <stdio.h>
#include <locale.h>
#include "Dados.h"
#include "Funcoes.h"
```

Include dependency graph for main.c:



Functions

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

  Inserção de parametros na função main.

5.5.1 Detailed Description

Local de implementação das funções.

Author

Pedro Ribeiro, Ricardo Fernandes, Carolina Branco (a27960@alunos.ipca.pt a279861@alunos.ipca.pt a27983@alunos.ipca.pt)

Version

0.1

Date

2023-12-27

Copyright

5.5.2 Function Documentation

5.5.2.1 main()

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

Inserção de parametros na função main.

Parameters

argc	número de argumentos
argv	array de argumentos, indica a posição do argumento (começa no 0)

Returns

int

```
24                                  {
25
26      setlocale(LC_ALL, "Portuguese");
27      char separador = ';';
28      char fpacientes[N], fdietas[N], fplanos[N];
29      int aux = 0;
30      Paciente dados[E];
31      Dieta dietas[E];
32      PlanoNutri nutri[E];
33
34      if (argc < 3 && strcmp(argv[1], "-ajuda")!=0){
35          printf ("Número insuficiente de argumentos\n");
36          return 0;
37      }
38      if (strcmp(argv[1], "-ajuda") == 0) {
39          MostraAjuda();
40          return 0;
41      }
42      if (strcmp(argv[1], "-tab") == 0) {
43          if(argc < 4){
44              printf ("Número insuficiente de argumentos\n");
45              return 0;
46          }else{
47              separador = '\t';
48              strcpy(fpacientes, argv[2]);
49              strcpy(fdietas, argv[3]);
50              if (argc == 5) {
51                  strcpy (fplanos, argv[4]);
52                  aux = 1;
53              }
54          }
55      }
56      else {
57          if(argc < 3){
58              printf ("Número insuficiente de argumentos\n");
59              return 0;
60          }else{
61              strcpy(fpacientes, argv[1]);
62              strcpy(fdietas, argv[2]);
63              if (argc == 4) {
64                  strcpy (fplanos, argv[3]);
65                  aux = 1;
66              }
67          }
68      }
69      if (aux == 0) {
70          char linha[E][1000];
71          int i=0;
72          while (i < E && fgets(linha[i], 1000, stdin) != NULL) {
73              if (linha[i][0] == '\n') {
```

```
74                 printf("Fim da entrada.\n");
75                 break;
76             }
77         i++;
78         }
79 
80         for (int j = 0; j < i; j++) {
81             printf("Dados lidos: %s", linha[j]);
82         }
83 
84         FILE *fplanosFILE;
85 
86         if (separador == ';') {
87             fplanosFILE = fopen("planosstdin.csv", "w");
88             strcpy(fplanos,"planosstdin.csv");
89         } else if (separador == '\t') {
90             fplanosFILE = fopen("tplanosstdin.csv", "w");
91             strcpy(fplanos,"tplanosstdin.csv");
92         } else {
93             printf("Separador não suportado.\n");
94             return 1;
95         }
96 
97         if (fplanosFILE == NULL) {
98             perror("Erro ao abrir o ficheiro de planos");
99             return 1;
100         }
101 
102         for (int j = 0; j < i; j++) {
103             fputs(linha[j], fplanosFILE);
104         }
105 
106         fclose(fplanosFILE);
107 
108     }
109 
110     LeDadosPacientes(separador, fpacientes, dados, E);
111     LeDadosDieta(separador, fdietas, dietas, E);
112     LeDadosPlanoNutri(separador, fplanos, nutri, E);
113 
114     NaoCumpPaciente nCump[E];
115 
116     MediaPaciente mediaPc[E];
117 
118 #pragma region TÓPICO 1. a. (main)
119 
120     if (ExportaDadosPacientes("ExpPacientes.csv", dados, E)) {
121         printf("Dados dos pacientes exportados com sucesso...\n\n");
122     }
123     else {
124         printf("Erro ao exportar os dados dos pacientes...\n\n");
125     }
126 
127 #pragma endregion
128 
129 #pragma region TÓPICO 1. b. (main)
130 
131     if (ExportaDadosDieta("ExpDietas.csv", dietas, E)) {
132         printf("Dados das dietas exportados com sucesso...\n\n");
133     }
134     else {
135         printf("Erro ao exportar os dados das dietas...\n\n");
136     }
137 
138 #pragma endregion
139 
140 #pragma region TÓPICO 1. c. (main)
141 
142     if (ExportaDadosPlanoNutri("ExpPlanoNutricional.csv", nutri, E)) {
143         printf("Dados dos planos nutricionais exportados com sucesso...\n\n");
144     }
145     else {
146         printf("Erro ao exportar os dados dos planos nutricionais...\n\n");
147     }
148 
149 #pragma endregion
150 
151 #pragma region TÓPICO 2
152 
153     int pacientesUltrapassam = NumPacientesUltrapassamCal(nutri, dietas, E);
154     printf("Número de pacientes que ultrapassaram o limite de calorias: %d\n\n", pacientesUltrapassam);
155 
156 #pragma endregion
157 
158 #pragma region TÓPICO 3
159 
160     IdentificaForaIntervalo(nutri, dietas, dados, nCump, E);
```

```
161     OrdenaPacientesForaIntervalo(nCump, E);
162
163     // Exibir resultados
164     printf("Pacientes fora do intervalo ordenados:\n");
165     for (int i = 0; i < E; i++) {
166         if (nCump[i].numpaciente < -1) { //em caso do array ter espaços ocupados com lixo
167             break;
168         }
169         else {
170             printf("%d - %s, %d\n", nCump[i].numpaciente, nCump[i].nome, nCump[i].telefone);
171         }
172     }
173     printf("\n");
174
175 #pragma endregion
176
177 #pragma region TÓPICO 4
178
179     ListaPlanoNutricional(nutri, E, 1, 0, "10-05-2023", "12-05-2023");
180
181 #pragma endregion
182
183 #pragma region TÓPICO 5
184
185     CalcularMediasRefeicoes(dietas, mediaPc, E, E, "10-05-2023", "12-05-2023");
186
187     for (int i = 0; i < E; i++) {
188         printf("Paciente %d:\n", mediaPc[i].numpaciente);
189         printf("Média Pequeno Almoço: %.2f\n", mediaPc[i].mediapequenoalmoco);
190         printf("Média Almoço: %.2f\n", mediaPc[i].mediaalmoco);
191         printf("Média Jantar: %.2f\n\n", mediaPc[i].mediajantar);
192     }
193
194 #pragma endregion
195
196 #pragma region TÓPICO 6
197
198     GerarTabelaRefeicoes(dados, dietas, nutri, 3, 3, 3);
199
200 #pragma endregion
201
202
203
204
205
206 }
```