LESI_PI_TP_26017_29851_27981

AUTHOR
Versão 1.0

Índice

Table of contents

Índice das estruturas de dados

Estruturas de dados

Lista das estruturas de dados com uma breve descrição:

Índice dos ficheiros

Lista de ficheiros

Lista de todos os ficheiros com uma breve descrição:

Documentação da estruturas de dados

Referência à estrutura Dieta

`#include <Header.h>`

Campos de Dados

- int **numPaciente**
- **Refeicao refeicao**
- char **ali** [**M**]
- int **cal**
- time_t **data**

Documentação dos campos e atributos

char ali[M]

int cal

time_t data

int numPaciente

Refeicao refeicao

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

- C:/AulasProgramacaoImperativa/TrabalhosPraticos/TrabalhoPratico1/**Header.h**

Referência à estrutura MediaCalorias

`#include <Header.h>`

Campos de Dados

- int **numPaciente**
- float **mediaCal**
- **Refeicao refeicao**

Documentação dos campos e atributos

float mediaCal

int numPaciente

Refeicao refeicao

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

- C:/AulasProgramacaoImperativa/TrabalhosPraticos/TrabalhoPratico1/**Header.h**

Referência à estrutura Paciente

`#include <Header.h>`

Campos de Dados

- int **numPaciente**
- char **nome** [**N**]
- long **tel**
- **Plano planos** [**N**]

Documentação dos campos e atributos

char nome[N]

int numPaciente

Plano planos[N]

long tel

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

- C:/AulasProgramacaoImperativa/TrabalhosPraticos/TrabalhoPratico1/**Header.h**

Referência à estrutura Plano

`#include <Header.h>`

Campos de Dados

- int **numPaciente**
- **Refeicao refeicao**
- int **minCal**
- int **maxCal**
- time_t **dataInicio**
- time_t **dataFim**

Documentação dos campos e atributos

time_t dataFim

time_t dataInicio

int maxCal

int minCal

int numPaciente

Refeicao refeicao

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

- C:/AulasProgramacaoImperativa/TrabalhosPraticos/TrabalhoPratico1/**Header.h**

Documentação do ficheiro

Referência ao ficheiro C:/AulasProgramacaoImperativa/TrabalhosPraticos/TrabalhoPratico1/Funcoes.c

```
#include <stdio.h>
#include <string.h>
#include "Header.h"
```

Funções

- int **ImportarPacientes** (**Paciente** pacientes[], char filename[])
  Importa pacientes de um ficheiro CSV.

- int **DietaPaciente** (**Dieta** dietas[], char filename[])
  Importa dieta dos pacientes de um ficheiro CSV.

- int **ImportarPlanos** (**Plano** planos[], char filename[], **Paciente** pacientes[])
  Importa planos nutricionais de um ficheiro CSV.

- int **ConverteDataString** (time_t data, char dataString[], int tamString)
  Converte time_t para string.

- time_t **ConverteDataTimet** (char dataString[])
  Converte string para time_t.

- int **ConverteRefeicaoString** (int refeicao, char refeicaoString[])
  Converte enum refeição para string.

- int **NumPacientesPassaLim** (**Dieta** dietas[], int tamDietas, **Paciente** pacientes[], int tamPacientes, int calMax, time_t dataDieta)
  Conta o nº de pacientes que ultrapassaram um limite de calorias num determinado dia.

- int **ListaPacientesCalMais** (**Plano** planos[], int tamPlanos, **Dieta** dietas[], int tamDietas, **Paciente** pacientes[], int tamPacientes, **Paciente** arrayOrdPacientes[])
  Conta o nº de pacientes que ultrapassaram o limite de calorias do plano.

- bool **ExisteNmrPaciente** (**Paciente** pacientes[], int tamPacientes, int numPaciente)
  Verifica se já existe um paciente com o mesmo nº

- int **AssociaPlano** (**Plano** plano, **Paciente** pacientes[])
  Adiciona o plano a um paciente.

- int **ListarPlanoPorRefeicao** (**Plano** planos[], **Plano** detalhes[], int tamPlanos, int numPaciente, **Refeicao** refeicao, time_t dataMin, time_t dataMax)
  Lista plano por refeicao.

- int **calcularMediaCalorias** (**Dieta** dietas[], int numDietas, **Paciente** pacientes[], int numPacientes, **Refeicao** refeicao, **MediaCalorias** mc[], time_t dataMin, time_t dataMax)
  Calcula media das calorias de uma refeição.

- void **TabelaDietas** (**Dieta** dietas[], int tamDietas, **Paciente** pacientes[], int tamPacientes, **Plano** planos[], int tamPlanos)
  Desenha a tabela para cada dieta com plano associado registada (FULLSCREEN)

Documentação das funções

int AssociaPlano (Plano *plano*, Paciente *pacientes*[])

Adiciona o plano a um paciente.

Parâmetros

plano	
pacientes	

```
310                                                 {
311     for (int i = 0; i < N; i++)
312     {
313         if(plano.numPaciente == pacientes[i].numPaciente) {
314             //Procura um sitio vazio no array de planos para preencher com um plano
novo
315             for (int j = 0; j < N; j++)
316             {
317                 if (pacientes[i].planos[j].numPaciente <= 0)
318                 {
319                     pacientes[i].planos[j] = plano;
320                     return 1;
321                 }
322             }
323             return 0;
324         }
325     }
326     return 0;
327 }
```

int calcularMediaCalorias (Dieta *dietas*[], int *numDietas*, Paciente *pacientes*[], int *numPacientes*, Refeicao *refeicao*, MediaCalorias *mc*[], time_t *dataMin*, time_t *dataMax*)

Calcula media das calorias de uma refeição.

Parâmetros

dietas	
numDietas	
pacientes	
numPacientes	
refeicao	
mc	
dataMin	
dataMax	

Retorna

```
372
{
373     double somaCalorias = 0;
374     int numRefeicoes = 0;
375
376     int posMedia = 0;
377
378     for (int j = 0; j < numPacientes; j++)
379     {
380
381         // Calcule a soma das calorias para a refeição e paciente específicos
382         for (int i = 0; i < numDietas; i++) {
383             if (dietas[i].numPaciente == pacientes[j].numPaciente &&
dietas[i].refeicao == refeicao && dietas[i].data >= dataMin && dietas[i].data <=
dataMax) {
384                 somaCalorias += dietas[i].cal;
385                 numRefeicoes++;
386             }
387         }
388
389         // Calcule a média de calorias se houver pelo menos uma refeição
390         if (numRefeicoes > 0) {
391             mc[posMedia].numPaciente = pacientes[j].numPaciente;
392             mc[posMedia].mediaCal = somaCalorias/numRefeicoes;
393             mc[posMedia].refeicao = refeicao;
394
395             numRefeicoes = 0;
396             posMedia++;
397         }
398     }
399
400 }
```

int ConverteDataString (time_t *data*, char *dataString*[], int *tamString*)

Converte time_t para string.

Parâmetros

data	
dataString	
tamString	

Retorna

```
143                                                                     {
144
145     //converte data numa struct
146     struct tm *dataStruct = localtime(&data);
147
148     //traduz a struct numa string
149     if(strftime(dataString, tamString, "%d/%m/%Y", dataStruct) == 0) return 0;
150
151     return 1;
152 }
```

time_t ConverteDataTimet (char *dataString*[])

Converte string para time_t.

Parâmetros

data	
dataString	
tamString	

Retorna

```
161                                                 {
162
163     //converte data numa struct
164     struct tm dataStruct = { 0 };
165
166     time_t data;
167
168     //converte as strings struct tm
169     sscanf(dataString, "%d/%d/%d", &dataStruct.tm_mday, &dataStruct.tm_mon,
&dataStruct.tm_year);
170
171     //ajusta os valores para os valores esperados na struct tm
172     dataStruct.tm_mon -= 1; //Para o mês ficar 0-11
173     dataStruct.tm_year -= 1900; //Para o ano começar em 1900
174
175     //converter para a variável time_t
176     data = mktime(&dataStruct);
177
178     return data;
179 }
```

int ConverteRefeicaoString (int *refeicao*, char *refeicaoString*[])

Converte enum refeição para string.

Parâmetros

refeicao	
refeicaoString	

Retorna

```
187                                                                     {
188     if (refeicao == 0) strcpy(refeicaoString, "PA");
189     if (refeicao == 1) strcpy(refeicaoString, "A");
190     if (refeicao == 2) strcpy(refeicaoString, "J");
191
192     return 1;
193 }
```

int DietaPaciente (Dieta *dietas*[], char *filename*[])

Importa dieta dos pacientes de um ficheiro CSV.

Parâmetros

dieta	
filename	

Retorna

```
54                                                      {
55     FILE* fp;
56     fp = fopen(filename, "r");
57     if (fp == NULL) return 0;
58
59     char dataString[N];
60
61     struct tm data = { 0 };
62
63     int i = 0;
64     while (1)
65     {
```

```
66          fscanf(fp, "%d;%[^;];%d;%[^;];%d", &dietas[i].numPaciente, dataString,
&dietas[i].refeicao, dietas[i].ali, &dietas[i].cal);
67
68          //converte as strings struct tm
69          sscanf(dataString, "%d/%d/%d", &data.tm_mday, &data.tm_mon,
&data.tm_year);
70
71          //ajusta os valores para os valores esperados na struct tm
72          data.tm_mon -= 1; //Para o mês ficar 0-11
73          data.tm_year -= 1900; //Para o ano começar em 1900
74
75          //converter para a variável time_t
76          dietas[i].data = mktime(&data);
77
78          if (feof(fp)) break;
79          i++;
80      }
81      fclose(fp);
82      return 1;
83  }
```

bool ExisteNmrPaciente (Paciente *pacientes*[], int *tamPacientes*, int *numPaciente*)

Verifica se já existe um paciente com o mesmo nº

Parâmetros

pacientes	
tamPacientes	
numPaciente	

Retorna

```
295
{
296
297      for (int i = 0; i < tamPacientes; i++)
298      {
299          if (pacientes[i].numPaciente == numPaciente) return true;
300      }
301
302      return false;
303  }
```

int ImportarPacientes (Paciente *pacientes*[], char *filename*[])

Importa pacientes de um ficheiro CSV.

Parâmetros

pacientes	
filename	

Retorna

```
32                                                              {
33      FILE* fp;
34      fp = fopen(filename, "r");
35      if (fp == NULL) return 0;
36
37      int i = 0;
38      while (1)
39      {
40          fscanf(fp, "%d;%[^;];%ld\n", &pacientes[i].numPaciente,
pacientes[i].nome, &pacientes[i].tel);
41          if (feof(fp)) break;
42          i++;
```

```
43      }
44      fclose(fp);
45      return 1;
46  }
```

int ImportarPlanos (Plano *planos*[], char *filename*[], Paciente *pacientes*[])

Importa planos nutricionais de um ficheiro CSV.

Parâmetros

planos	
filename	
pacientes	

Retorna

```
92                                                                      {
93      FILE* fp;
94      fp = fopen(filename, "r");
95      if (fp == NULL) return 0;
96
97      //strings que serão recebidas para depois converter para time_t
98      char dataInicioString[N];
99      char dataFimString[N];
100
101      //structs tm auxiliares para converter para time_t
102      struct tm dataInicio = {0};
103      struct tm dataFim = {0};
104
105      int i = 0;
106      while (1)
107      {
108
109          fscanf(fp, "%d;%[^;];%[^;];%d;%d;%d\n", &planos[i].numPaciente,
dataInicioString, dataFimString, &planos[i].refeicao, &planos[i].minCal,
&planos[i].maxCal);
110
111          //converte as strings struct tm
112          sscanf(dataInicioString, "%d/%d/%d", &dataInicio.tm_mday,
&dataInicio.tm_mon, &dataInicio.tm_year);
113          sscanf(dataFimString, "%d/%d/%d", &dataFim.tm_mday, &dataFim.tm_mon,
&dataFim.tm_year);
114
115          //ajusta os valores para os valores esperados na struct tm
116          dataInicio.tm_mon -= 1; //Para o mês ficar 0-11
117          dataInicio.tm_year -= 1900; //Para o ano começar em 1900
118
119          dataFim.tm_mon -= 1; //Para o mês ficar 0-11
120          dataFim.tm_year -= 1900; //Para o ano começar em 1900
121
122          //converter para a variável time_t
123          planos[i].dataInicio = mktime(&dataInicio);
124          planos[i].dataFim = mktime(&dataFim);
125
126          //AssociaPlano(planos[i], pacientes);
127          if (feof(fp)) break;
128          i++;
129      }
130      fclose(fp);
131      return 1;
132  }
```

int ListaPacientesCalMais (Plano *planos*[], int *tamPlanos*, Dieta *dietas*[], int *tamDietas*, Paciente *pacientes*[], int *tamPacientes*, Paciente *arrayOrdPacientes*[])

Conta o nº de pacientes que ultrapassaram o limite de calorias do plano.

Parâmetros

planos	
tamPlanos	
dietas	
tamDietas	

Retorna

```
236
{
237     int pacientesCalMais = 0;
238
239     int posArray = 0;
240
241     int calPaciente = 0;
242
243     //Corre os planos
244     for (int i = 0; i < tamPlanos; i++)
245     {
246         calPaciente = 0;
247
248         //Corre as dietas (para somar todas as calorias)
249         for (int j = 0; j < tamDietas; j++)
250         {
251             if (planos[i].numPaciente == dietas[j].numPaciente &&
planos[i].refeicao == dietas[j].refeicao && planos[i].dataInicio <= dietas[j].data &&
planos[i].dataFim >= dietas[j].data)
252             {
253                 calPaciente = calPaciente + dietas[j].cal;
254             }
255         }
256
257
258         //corre os pacientes ignorando os duplicados(para guardar o paciente certo
no array)
259         if ((calPaciente > planos[i].maxCal || calPaciente < planos[i].minCal) &&
calPaciente != 0 && ExisteNmrPaciente(pacientes, tamPacientes,
planos[i].numPaciente)) {
260             for (int p = 0; p < tamPacientes; p++)
261             {
262                 if (planos[i].numPaciente == pacientes[p].numPaciente &&
!ExisteNmrPaciente(arrayOrdPacientes, posArray+1, planos[i].numPaciente)) {
263                     arrayOrdPacientes[posArray] = pacientes[p];
264                     posArray++;
265                     break;
266                 }
267             }
268         }
269     }
270
271     //ordena por ordem decrescente
272     for (int i = 0; i < posArray + 1; i++)
273     {
274         for (int j = 0; j < tamPacientes; j++) {
275             if (arrayOrdPacientes[i].numPaciente >
arrayOrdPacientes[j].numPaciente)
276             {
277                 Paciente aux = arrayOrdPacientes[i];
278                 arrayOrdPacientes[i] = arrayOrdPacientes[j];
279                 arrayOrdPacientes[j] = aux;
280             }
281         }
282     }
283
284     //pacientes;
285     return pacientesCalMais;
286 }
```

int ListarPlanoPorRefeicao (Plano *planos*[], Plano *detalhes*[], int *tamPlanos*, int *numPaciente*, Refeicao *refeicao*, time_t *dataMin*, time_t *dataMax*)

Lista plano por refeicao.

Parâmetros

planos	
detalhes	
tamPlanos	
numPaciente	
refeicao	
dataMin	
dataMax	

Retorna

```
342
{
343     int contaDetalhes = 0;
344     for (int i = 0; i < tamPlanos; i++) {
345         if (planos[i].numPaciente == numPaciente && planos[i].refeicao == refeicao
&& dataMin <= planos[i].dataInicio && dataMax >= planos[i].dataFim) {
346             detalhes[contaDetalhes].numPaciente = planos[i].numPaciente;
347             detalhes[contaDetalhes].dataInicio = planos[i].dataInicio;
348             detalhes[contaDetalhes].dataFim = planos[i].dataFim;
349             detalhes[contaDetalhes].minCal = planos[i].minCal;
350             detalhes[contaDetalhes].maxCal = planos[i].maxCal;
351             detalhes[contaDetalhes].refeicao = planos[i].refeicao;
352             contaDetalhes++;
353         }
354     }return 1;
355
356 }
```

int NumPacientesPassaLim (Dieta *dietas*[], int *tamDietas*, Paciente *pacientes*[], int *tamPacientes*, int *calMax*, time_t *dataDieta*)

Conta o nº de pacientes que ultrapassaram um limite de calorias num determinado dia.

Parâmetros

dietas	
tamDietas	
calMax	
dataDieta	

Retorna

```
206
{
207     int pacientesCalMais = 0;
208     int calPaciente = 0;
209     //Corre todas as dietas de um paciente
210     for (int i = 0; i < tamPacientes; i++)
211     {
212         calPaciente = 0;
213
214         for (int j = 0; j < tamDietas; j++)
215         {
216             if (dietas[j].numPaciente == pacientes[i].numPaciente &&
dietas[i].data == dataDieta)
217             {
218                 calPaciente += dietas[j].cal;
219             }
220         }
221         if (calPaciente > calMax) pacientesCalMais++;
222     }
223     return pacientesCalMais;
224 }
```

void TabelaDietas (Dieta *dietas*[], int *tamDietas*, Paciente *pacientes*[], int *tamPacientes*, Plano *planos*[], int *tamPlanos*)

Desenha a tabela para cada dieta com plano associado registada (FULLSCREEN)

Parâmetros

dietas	
tamDietas	
pacientes	
tamPacientes	
planos	
tamPlanos	

```
413
{
414
415     Paciente pacienteTemp;
416     Plano planoTemp;
417     int caloriasRefeicao = 0;
418
419     char refeicao[N];
420     char dataInicio[N];
421     char dataFim[N];
422
423     bool saltarDieta = false;
424     bool temPlano = false;
425
426     //Escreve o cabeçalho da tabela
427
printf("+-------------------------------------------------------------------------
-------------------------------------------------------------------------------
----+\n");
428     printf("|   NP  |   Paciente    |   Refeicao    |   Data Inicio |   Data Fim
|   Min |   Max |   Consumo |\n");
429
printf("|-------------------------------------------------------------------------
-------------------------------------------------------------------------------
----|\n");
430
431
432     for (int i = 0; i < tamDietas; i++)
433     {
434         caloriasRefeicao += dietas[i].cal;
435
436         //Se existir outra dieta no mesmo dia e na mesma refeicao, incrementa
calorias
437         for (int j = 0; j < tamDietas; j++)
438         {
439             if (dietas[i].numPaciente == dietas[j].numPaciente &&
dietas[i].refeicao == dietas[j].refeicao && dietas[i].data == dietas[j].data)
440             {
441                 if (i == j) continue;
442
443                 //Se já escreveu esta linha, salta para a próxima
444                 if (i > j) {
445                     saltarDieta = true;
446                     break;
447                 }
448                 caloriasRefeicao += dietas[j].cal;
449             }
450         }
451
452         //Procura as infos do paciente na dieta atual
453         for (int j = 0; j < tamPacientes; j++)
454         {
455             if (dietas[i].numPaciente == pacientes[j].numPaciente) {
456                 pacienteTemp = pacientes[j];
457                 break;
458             }
459         }
460
```

```
461         temPlano = false;
462         //Procura as infos do plano a que corresponde a dieta
463         for (int j = 0; j < tamPlanos; j++)
464         {
465             if (dietas[i].numPaciente == planos[j].numPaciente &&
dietas[i].refeicao == planos[j].refeicao && planos[j].dataInicio <= dietas[i].data &&
planos[j].dataFim >= dietas[i].data) {
466                 planoTemp = planos[j];
467                 temPlano = true;
468                 break;
469             }
470         }
471 
472         // salta para a proxima linha caso já tenha escrito esta dieta ou a dieta
não tenha plano associado
473         if (saltarDieta || !temPlano) {
474             saltarDieta = false;
475             caloriasRefeicao = 0;
476             continue;
477         }
478 
479         // converte a refeição para string para ser mostrada
480         ConverteRefeicaoString(dietas[i].refeicao, refeicao);
481 
482         //organiza as datas para dia/mes/ano
483         ConverteDataString(planoTemp.dataInicio, dataInicio, N);
484 
485         ConverteDataString(planoTemp.dataFim, dataFim, N);
486 
487         printf("|   %d  |   %s      |   %s      |   %s  |   %s  |   %d  |   %d
|   %d  |\n", dietas[i].numPaciente, pacienteTemp.nome, refeicao, dataInicio,
dataFim, planoTemp.minCal, planoTemp.maxCal, caloriasRefeicao);
488         caloriasRefeicao = 0;
489     }
490 
printf("+-------------------------------------------------------------------------
-------------------------------------------------------------------------------
----+\n");
491 }
```

Referência ao ficheiro C:/AulasProgramacaoImperativa/TrabalhosPraticos/TrabalhoPratico1/Header.h

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

Estruturas de Dados

- struct **Plano**struct **Dieta**
- struct **Paciente**
- struct **MediaCalorias**

Macros

- #define **N** 25
- #define **M** 50

Definições de tipos

- typedef enum **Refeicao Refeicao**
- typedef struct **Plano Plano**
- typedef struct **Dieta Dieta**
- typedef struct **Paciente Paciente**
- typedef struct **MediaCalorias MediaCalorias**

Enumerações

- enum **Refeicao** { **PequenoAlmoco**, **Almoco**, **Jantar** }

Funções

- int **ImportarPacientes** (**Paciente** pacientes[], char filename[])
  Importa pacientes de um ficheiro CSV.

- int **ImportarPlanos** (**Plano** planos[], char filename[], **Paciente** pacientes[])
  Importa planos nutricionais de um ficheiro CSV.

- int **DietaPaciente** (**Dieta** dietas[], char filename[])
  Importa dieta dos pacientes de um ficheiro CSV.

- int **ConverteDataString** (time_t data, char dataString[], int tamString)
  Converte time_t para string.

- time_t **ConverteDataTimet** (char dataString[])
  Converte string para time_t.

- int **ListarPlanoPorRefeicao** (**Plano** planos[], **Plano** detalhes[], int tamPlanos, int numPaciente, **Refeicao** refeicao, time_t dataMin, time_t dataMax)
  Lista plano por refeicao.

- int **calcularMediaCalorias** (**Dieta** dietas[], int numDietas, **Paciente** pacientes[], int numPacientes, **Refeicao** refeicao, **MediaCalorias** mc[], time_t dataMin, time_t dataMax)
  Calcula media das calorias de uma refeição.

- int **NumPacientesPassaLim** (**Dieta** dietas[], int tamDietas, **Paciente** pacientes[], int tamPacientes, int calMax, time_t dataDieta)
  Conta o nº de pacientes que ultrapassaram um limite de calorias num determinado dia.

- bool **ExisteNmrPaciente** (**Paciente** pacientes[], int tamPacientes, int numPaciente)
  Verifica se já existe um paciente com o mesmo nº

- int **ListaPacientesCalMais** (**Plano** planos[], int tamPlanos, **Dieta** dietas[], int tamDietas, **Paciente** pacientes[], int tamPacientes, **Paciente** arrayOrdPacientes[])
  Conta o nº de pacientes que ultrapassaram o limite de calorias do plano.

- void **TabelaDietas** (**Dieta** dietas[], int tamDietas, **Paciente** pacientes[], int tamPacientes, **Plano** planos[], int tamPlanos)
  Desenha a tabela para cada dieta com plano associado registada (FULLSCREEN)

- int **ConverteRefeicaoString** (int refeicao, char refeicaoString[])
  Converte enum refeição para string.

Documentação das macros

#define M 50

#define N 25

Documentação dos tipos

typedef struct Dieta Dieta

typedef struct MediaCalorias MediaCalorias

typedef struct Paciente Paciente

typedef struct Plano Plano

typedef enum Refeicao Refeicao

Documentação dos valores da enumeração

enum Refeicao

Valores de enumerações:

PequenoAlmoco	
Almoco	
Jantar	

```
25                      {
26     PequenoAlmoco,
27     Almoco,
```

```
28     Jantar
29 } Refeicao;
```

Documentação das funções

int calcularMediaCalorias (Dieta *dietas*[], int *numDietas*, Paciente *pacientes*[], int *numPacientes*, Refeicao *refeicao*, MediaCalorias *mc*[], time_t *dataMin*, time_t *dataMax*)

Calcula media das calorias de uma refeição.

Parâmetros

dietas	
numDietas	
pacientes	
numPacientes	
refeicao	
mc	
dataMin	
dataMax	

Retorna

```
372
{
373     double somaCalorias = 0;
374     int numRefeicoes = 0;
375
376     int posMedia = 0;
377
378     for (int j = 0; j < numPacientes; j++)
379     {
380
381         // Calcule a soma das calorias para a refeição e paciente específicos
382         for (int i = 0; i < numDietas; i++) {
383             if (dietas[i].numPaciente == pacientes[j].numPaciente &&
dietas[i].refeicao == refeicao && dietas[i].data >= dataMin && dietas[i].data <=
dataMax) {
384                 somaCalorias += dietas[i].cal;
385                 numRefeicoes++;
386             }
387         }
388
389         // Calcule a média de calorias se houver pelo menos uma refeição
390         if (numRefeicoes > 0) {
391             mc[posMedia].numPaciente = pacientes[j].numPaciente;
392             mc[posMedia].mediaCal = somaCalorias/numRefeicoes;
393             mc[posMedia].refeicao = refeicao;
394
395             numRefeicoes = 0;
396             posMedia++;
397         }
398     }
399
400 }
```

int ConverteDataString (time_t *data*, char *dataString*[], int *tamString*)

Converte time_t para string.

Parâmetros

data	
dataString	
tamString	

Retorna

```
143                                                                     {
144
145     //converte data numa struct
146     struct tm *dataStruct = localtime(&data);
147
148     //traduz a struct numa string
149     if(strftime(dataString, tamString, "%d/%m/%Y", dataStruct) == 0) return 0;
150
151     return 1;
152 }
```

time_t ConverteDataTimet (char *dataString*[])

Converte string para time_t.

Parâmetros

data	
dataString	
tamString	

Retorna

```
161                                             {
162
163     //converte data numa struct
164     struct tm dataStruct = { 0 };
165
166     time_t data;
167
168     //converte as strings struct tm
169     sscanf(dataString, "%d/%d/%d", &dataStruct.tm_mday, &dataStruct.tm_mon,
&dataStruct.tm_year);
170
171     //ajusta os valores para os valores esperados na struct tm
172     dataStruct.tm_mon -= 1; //Para o mês ficar 0-11
173     dataStruct.tm_year -= 1900; //Para o ano começar em 1900
174
175     //converter para a variável time_t
176     data = mktime(&dataStruct);
177
178     return data;
179 }
```

int ConverteRefeicaoString (int *refeicao*, char *refeicaoString*[])

Converte enum refeição para string.

Parâmetros

refeicao	
refeicaoString	

Retorna

```
187                                                             {
188     if (refeicao == 0) strcpy(refeicaoString, "PA");
189     if (refeicao == 1) strcpy(refeicaoString, "A");
```

```
190     if (refeicao == 2) strcpy(refeicaoString, "J");
191
192     return 1;
193 }
```

int DietaPaciente (Dieta *dietas*[], char *filename*[])

Importa dieta dos pacientes de um ficheiro CSV.

Parâmetros

dieta	
filename	

Retorna

```
54                                                      {
55     FILE* fp;
56     fp = fopen(filename, "r");
57     if (fp == NULL) return 0;
58
59     char dataString[N];
60
61     struct tm data = { 0 };
62
63     int i = 0;
64     while (1)
65     {
66         fscanf(fp, "%d;%[^;];%d;%[^;];%d", &dietas[i].numPaciente, dataString,
&dietas[i].refeicao, dietas[i].ali, &dietas[i].cal);
67
68         //converte as strings struct tm
69         sscanf(dataString, "%d/%d/%d", &data.tm_mday, &data.tm_mon,
&data.tm_year);
70
71         //ajusta os valores para os valores esperados na struct tm
72         data.tm_mon -= 1; //Para o mês ficar 0-11
73         data.tm_year -= 1900; //Para o ano começar em 1900
74
75         //converter para a variável time_t
76         dietas[i].data = mktime(&data);
77
78         if (feof(fp)) break;
79         i++;
80     }
81     fclose(fp);
82     return 1;
83 }
```

bool ExisteNmrPaciente (Paciente *pacientes*[], int *tamPacientes*, int *numPaciente*)

Verifica se já existe um paciente com o mesmo nº

Parâmetros

pacientes	
tamPacientes	
numPaciente	

Retorna

```
295
{
296
297     for (int i = 0; i < tamPacientes; i++)
298     {
299         if (pacientes[i].numPaciente == numPaciente) return true;
```

```
300     }
301
302     return false;
303 }
```

int ImportarPacientes (Paciente *pacientes*[], char *filename*[])

Importa pacientes de um ficheiro CSV.

Parâmetros

pacientes	
filename	

Retorna

```
32                                                                  {
33     FILE* fp;
34     fp = fopen(filename, "r");
35     if (fp == NULL) return 0;
36
37     int i = 0;
38     while (1)
39     {
40         fscanf(fp, "%d;%[^;];%ld\n", &pacientes[i].numPaciente,
pacientes[i].nome, &pacientes[i].tel);
41         if (feof(fp)) break;
42         i++;
43     }
44     fclose(fp);
45     return 1;
46 }
```

int ImportarPlanos (Plano *planos*[], char *filename*[], Paciente *pacientes*[])

Importa planos nutricionais de um ficheiro CSV.

Parâmetros

planos	
filename	
pacientes	

Retorna

```
92                                                                      {
93     FILE* fp;
94     fp = fopen(filename, "r");
95     if (fp == NULL) return 0;
96
97     //strings que serão recebidas para depois converter para time_t
98     char dataInicioString[N];
99     char dataFimString[N];
100
101     //structs tm auxiliares para converter para time t
102     struct tm dataInicio = {0};
103     struct tm dataFim = {0};
104
105     int i = 0;
106     while (1)
107     {
108
109         fscanf(fp, "%d;%[^;];%[^;];%d;%d;%d\n", &planos[i].numPaciente,
dataInicioString, dataFimString, &planos[i].refeicao, &planos[i].minCal,
&planos[i].maxCal);
110
111         //converte as strings struct tm
```

```
112         sscanf(dataInicioString, "%d/%d/%d", &dataInicio.tm_mday,
&dataInicio.tm_mon, &dataInicio.tm_year);
113         sscanf(dataFimString, "%d/%d/%d", &dataFim.tm_mday, &dataFim.tm_mon,
&dataFim.tm_year);
114
115         //ajusta os valores para os valores esperados na struct tm
116         dataInicio.tm_mon -= 1; //Para o mês ficar 0-11
117         dataInicio.tm_year -= 1900; //Para o ano começar em 1900
118
119         dataFim.tm_mon -= 1; //Para o mês ficar 0-11
120         dataFim.tm_year -= 1900; //Para o ano começar em 1900
121
122         //converter para a variável time_t
123         planos[i].dataInicio = mktime(&dataInicio);
124         planos[i].dataFim = mktime(&dataFim);
125
126         //AssociaPlano(planos[i], pacientes);
127         if (feof(fp)) break;
128         i++;
129     }
130     fclose(fp);
131     return 1;
132 }
```

int ListaPacientesCalMais (Plano *planos*[], int *tamPlanos*, Dieta *dietas*[], int *tamDietas*, Paciente *pacientes*[], int *tamPacientes*, Paciente *arrayOrdPacientes*[])

Conta o nº de pacientes que ultrapassaram o limite de calorias do plano.

Parâmetros

planos	
tamPlanos	
dietas	
tamDietas	

Retorna

```
236
{
237     int pacientesCalMais = 0;
238
239     int posArray = 0;
240
241     int calPaciente = 0;
242
243     //Corre os planos
244     for (int i = 0; i < tamPlanos; i++)
245     {
246         calPaciente = 0;
247
248         //Corre as dietas (para somar todas as calorias)
249         for (int j = 0; j < tamDietas; j++)
250         {
251             if (planos[i].numPaciente == dietas[j].numPaciente &&
planos[i].refeicao == dietas[j].refeicao && planos[i].dataInicio <= dietas[j].data &&
planos[i].dataFim >= dietas[j].data)
252             {
253                 calPaciente = calPaciente + dietas[j].cal;
254             }
255         }
256
257
258         //corre os pacientes ignorando os duplicados(para guardar o paciente certo
no array)
259         if ((calPaciente > planos[i].maxCal || calPaciente < planos[i].minCal) &&
calPaciente != 0 && ExisteNmrPaciente(pacientes, tamPacientes,
planos[i].numPaciente)) {
260             for (int p = 0; p < tamPacientes; p++)
261             {
```

```
262                 if (planos[i].numPaciente == pacientes[p].numPaciente &&
!ExisteNmrPaciente(arrayOrdPacientes, posArray+1, planos[i].numPaciente)) {
263                     arrayOrdPacientes[posArray] = pacientes[p];
264                     posArray++;
265                     break;
266                 }
267             }
268         }
269     }
270 
271     //ordena por ordem decrescente
272     for (int i = 0; i < posArray + 1; i++)
273     {
274         for (int j = 0; j < tamPacientes; j++) {
275             if (arrayOrdPacientes[i].numPaciente >
arrayOrdPacientes[j].numPaciente)
276             {
277                 Paciente aux = arrayOrdPacientes[i];
278                 arrayOrdPacientes[i] = arrayOrdPacientes[j];
279                 arrayOrdPacientes[j] = aux;
280             }
281         }
282     }
283 
284     //pacientes;
285     return pacientesCalMais;
286 }
```

int ListarPlanoPorRefeicao (Plano *planos*[], Plano *detalhes*[], int *tamPlanos*, int *numPaciente*, Refeicao *refeicao*, time_t *dataMin*, time_t *dataMax*)

Lista plano por refeicao.

Parâmetros

planos	
detalhes	
tamPlanos	
numPaciente	
refeicao	
dataMin	
dataMax	

Retorna

```
342 
{
343     int contaDetalhes = 0;
344     for (int i = 0; i < tamPlanos; i++) {
345         if (planos[i].numPaciente == numPaciente && planos[i].refeicao == refeicao
&& dataMin <= planos[i].dataInicio && dataMax >= planos[i].dataFim) {
346             detalhes[contaDetalhes].numPaciente = planos[i].numPaciente;
347             detalhes[contaDetalhes].dataInicio = planos[i].dataInicio;
348             detalhes[contaDetalhes].dataFim = planos[i].dataFim;
349             detalhes[contaDetalhes].minCal = planos[i].minCal;
350             detalhes[contaDetalhes].maxCal = planos[i].maxCal;
351             detalhes[contaDetalhes].refeicao = planos[i].refeicao;
352             contaDetalhes++;
353         }
354     }return 1;
355 
356 }
```

int NumPacientesPassaLim (Dieta *dietas*[], int *tamDietas*, Paciente *pacientes*[], int *tamPacientes*, int *calMax*, time_t *dataDieta*)

Conta o nº de pacientes que ultrapassaram um limite de calorias num determinado dia.

Parâmetros

dietas	
tamDietas	
calMax	
dataDieta	

Retorna

```
206
{
207     int pacientesCalMais = 0;
208     int calPaciente = 0;
209     //Corre todas as dietas de um paciente
210     for (int i = 0; i < tamPacientes; i++)
211     {
212         calPaciente = 0;
213
214         for (int j = 0; j < tamDietas; j++)
215         {
216             if (dietas[j].numPaciente == pacientes[i].numPaciente &&
dietas[i].data == dataDieta)
217             {
218                 calPaciente += dietas[j].cal;
219             }
220         }
221         if (calPaciente > calMax) pacientesCalMais++;
222     }
223     return pacientesCalMais;
224 }
```

void TabelaDietas (Dieta *dietas*[], int *tamDietas*, Paciente *pacientes*[], int *tamPacientes*, Plano *planos*[], int *tamPlanos*)

Desenha a tabela para cada dieta com plano associado registada (FULLSCREEN)

Parâmetros

dietas	
tamDietas	
pacientes	
tamPacientes	
planos	
tamPlanos	

```
413
{
414
415     Paciente pacienteTemp;
416     Plano planoTemp;
417     int caloriasRefeicao = 0;
418
419     char refeicao[N];
420     char dataInicio[N];
421     char dataFim[N];
422
423     bool saltarDieta = false;
424     bool temPlano = false;
425
426     //Escreve o cabeçalho da tabela
427
printf("+-------------------------------------------------------------------------
-------------------------------------------------------------------------------
----+\n");
428     printf("|   NP  |   Paciente    |   Refeicao    |   Data Inicio |   Data Fim
|   Min |   Max |   Consumo |\n");
429
printf("|-------------------------------------------------------------------------
```

```
----------------------------------------------------------------------------
----|\n");
430
431
432     for (int i = 0; i < tamDietas; i++)
433     {
434         caloriasRefeicao += dietas[i].cal;
435
436         //Se existir outra dieta no mesmo dia e na mesma refeicao, incrementa
calorias
437         for (int j = 0; j < tamDietas; j++)
438         {
439             if (dietas[i].numPaciente == dietas[j].numPaciente &&
dietas[i].refeicao == dietas[j].refeicao && dietas[i].data == dietas[j].data)
440             {
441                 if (i == j) continue;
442
443                 //Se já escreveu esta linha, salta para a próxima
444                 if (i > j) {
445                     saltarDieta = true;
446                     break;
447                 }
448                 caloriasRefeicao += dietas[j].cal;
449             }
450         }
451
452         //Procura as infos do paciente na dieta atual
453         for (int j = 0; j < tamPacientes; j++)
454         {
455             if (dietas[i].numPaciente == pacientes[j].numPaciente) {
456                 pacienteTemp = pacientes[j];
457                 break;
458             }
459         }
460
461         temPlano = false;
462         //Procura as infos do plano a que corresponde a dieta
463         for (int j = 0; j < tamPlanos; j++)
464         {
465             if (dietas[i].numPaciente == planos[j].numPaciente &&
dietas[i].refeicao == planos[j].refeicao && planos[j].dataInicio <= dietas[i].data &&
planos[j].dataFim >= dietas[i].data) {
466                 planoTemp = planos[j];
467                 temPlano = true;
468                 break;
469             }
470         }
471
472         // salta para a proxima linha caso já tenha escrito esta dieta ou a dieta
não tenha plano associado
473         if (saltarDieta || !temPlano) {
474             saltarDieta = false;
475             caloriasRefeicao = 0;
476             continue;
477         }
478
479         // converte a refeição para string para ser mostrada
480         ConverteRefeicaoString(dietas[i].refeicao, refeicao);
481
482         //organiza as datas para dia/mes/ano
483         ConverteDataString(planoTemp.dataInicio, dataInicio, N);
484
485         ConverteDataString(planoTemp.dataFim, dataFim, N);
486
487         printf("|   %d  |   %s      |   %s      |   %s  |   %s  |   %d  |   %d
|   %d  |\n", dietas[i].numPaciente, pacienteTemp.nome, refeicao, dataInicio,
dataFim, planoTemp.minCal, planoTemp.maxCal, caloriasRefeicao);
488         caloriasRefeicao = 0;
489     }
490
printf("+----------------------------------------------------------------------------
----------------------------------------------------------------------------
----+\n");
491 }
```

Header.h

Ir para a documentação deste ficheiro.

```c
/*
 * Paulo Alexandre Rodrigues da Costa
 * a29851@alunos.ipca.pt
 *
 * Afonso Teixeira Ferreira
 * a27981@alunos.ipca.pt
 *
 * David João Martins Lopes
 * a26017@alunos.ipca.pt
 *
 * 22/11/2023
 *
 * Trabalho Prático 1
 *
 * UC: PI
 */
#pragma once

#include <stdbool.h>
#include <time.h>

#define N 25
#define M 50

typedef enum Refeicao {
    PequenoAlmoco,
    Almoco,
    Jantar
} Refeicao;

// Struct do plano nutricional
typedef struct Plano {
    int numPaciente;
    Refeicao refeicao;
    int minCal;
    int maxCal;
    time_t dataInicio;
    time_t dataFim;
} Plano;

typedef struct Dieta {
    int numPaciente;
    Refeicao refeicao;
    char ali[M];
    int cal;
    time_t data;
} Dieta;

// struct da informação do Paciente
typedef struct Paciente {
    int numPaciente;
    char nome[N];
    long tel;
    Plano planos[N];
} Paciente;

//struct das medias das calorias dos pacientes
typedef struct MediaCalorias {
    int numPaciente;
    float mediaCal;
    Refeicao refeicao;
} MediaCalorias;


// Importa pacientes de um ficheiro CSV
int ImportarPacientes(Paciente pacientes[], char filename[]);

// Importa planos nutricionais de um ficheiro CSV
int ImportarPlanos(Plano planos[], char filename[], Paciente pacientes[]);

// Importa dietas de um ficheiro CSV
int DietaPaciente(Dieta dietas[], char filename[]);
```

```
73
74 //Converte time_t para string
75 int ConverteDataString(time_t data, char dataString[], int tamString);
76
77 //Converte string para time_t
78 time_t ConverteDataTimet(char dataString[]);
79
80 // Lista o Plano por refeicao de um paciente
81 int ListarPlanoPorRefeicao(Plano planos[], Plano detalhes[], int tamPlanos, int
numPaciente, Refeicao refeicao, time_t dataMin, time_t dataMax);
82
83 //Calcula media das calorias de uma refeição
84 int calcularMediaCalorias(Dieta dietas[], int numDietas, Paciente pacientes[], int
numPacientes, Refeicao refeicao, MediaCalorias mc[], time_t dataMin, time_t dataMax);
85
86 // Conta o nº de pacientes que ultrapassaram um limite de calorias
87 int NumPacientesPassaLim(Dieta dietas[], int tamDietas, Paciente pacientes[], int
tamPacientes, int calMax, time_t dataDieta);
88
89 // Verifica se um paciente com o mesmo nº existe num array
90 bool ExisteNmrPaciente(Paciente pacientes[], int tamPacientes, int numPaciente);
91
92 // Lista por ordem decrescente os pacientes que ultrapassaram os limites do plano
93 int ListaPacientesCalMais(Plano planos[], int tamPlanos, Dieta dietas[], int tamDietas,
Paciente pacientes[], int tamPacientes, Paciente arrayOrdPacientes[]);
94
95 // Desenha a tabela para cada dieta com plano associado registada
96 void TabelaDietas(Dieta dietas[], int tamDietas, Paciente pacientes[], int
tamPacientes, Plano planos[], int tamPlanos);
97
98 // Converte enum refeição para string
99 int ConverteRefeicaoString(int refeicao, char refeicaoString[]);
```

Referência ao ficheiro C:/AulasProgramacaoImperativa/TrabalhosPraticos/TrabalhoPratico1/Source.c

```
#include "Header.h"
#include <stdio.h>
```

Funções

- int **main** ()

Documentação das funções

int main ()

```
21            {
22
23 }
```

Índice

INDEX