Grupo 30

Generated by Doxygen 1.8.17

Chapter 1

Trabalho Prático

Licenciatura em Engenharia de Sistemas Informáticos 2023-24

Laboratórios de Informática

1.1 Grupo 30

Número	Nome
a23010	Hugo Cruz
a23015	Lino Azevedo
a23016	Dani Cruz

1.2 Organização

`doc/` documentação com o relatório

`src/` Código da solução desenvolvida

1.3 Relatórios e Código

- **Relatório LaTeX:** `doc/relatorio/d-30-doc.pdf`
- **Relatório Doxygen:** `doc/latex/d-30-refman.pdf`
- **Código Fonte:** `src/`

1.4 Descrição do Projeto

O projeto consiste em desenvolver um programa em C para gerenciar informações relacionadas à saúde e nutrição de pacientes. O programa permite carregar dados de pacientes, informações sobre suas dietas e planos nutricionais a partir de arquivos de texto. Além disso, oferece funcionalidades como apresentar estatísticas sobre a ingestão de calorias, listar pacientes com refeições fora do intervalo prescrito e gerar tabelas de refeições planeadas e realizadas

1.5 Estratégia de Desenvolvimento

- **Hugo Cruz (a23010):** Implementação do modo de utilização, argumentos, listar o paciente do plano nutricional, tabelas ficheiros binários, e divisão em ficheiros .c, .h, CSV, desenvolvimento do relatório.
- **Lino Azevedo (a23015):** Implementação do contador, listagem de cliente fora intervalo de calorias, calculo da média de calorias por cliente, num determinado período, desenvolvimento do relatório.
- **Dani Cruz (a23016):** Implementação de ficheiros CSV e TSV, divisão em ficheiros .c .h para as funções TSV, e ficheiro.h contendo as struct, desenvolvimento do relatório.

1.6 Decisões Coletivas

-No âmbito da nossa colaboração no projeto em grupo, começámos por realizar uma seleção criteriosa de ficheiros CSV. Durante a análise, constatámos que esses ficheiros utilizam tanto pontos e vírgulas como vírgulas para separar as palavras. Esta escolha revelou-se fundamental, uma vez que verificámos que as informações presentes no enunciado seguiam precisamente essa estrutura. Optámos, assim, por utilizar esses ficheiros, com o propósito de assegurar uma correspondência exata com o formato das informações fornecidas, conclusão que considerámos ser a mais apropriada para atingir os nossos objetivos.

-Paralelamente, no decurso do desenvolvimento do projeto, deparámo-nos com um desafio relacionado à utilização de variáveis em funções. Após uma deliberação conjunta, decidimos adotar a abordagem estruturada através da utilização de "struct", o que nos possibilita a utilização das variáveis em qualquer momento do código.

-Por fim, decidimos ajustar a nossa estratégia em relação aos argumentos. Inicialmente, optámos por utilizar 1 argumento apenas para deferir o modo onde nos deparamos poder vir a ser um problema relativamente aos ficheiros binários, então colocamos 5 argumentos podendo definir os ficheiros que pretendíamos utilizar.

1.7 Compilação e Execução:

Para compilar este projeto, é necessário ter o comando "make" instalado. Após a instalação, um arquivo Makefile está disponível no diretório. Este arquivo é responsável por compilar automaticamente todos os arquivos. Basta executar make para realizar a compilação.

```
make
```

1.8 Comando com Sintaxe de Utilização:

- Ficheiros (CSV): `./prog -csv dados.csv dietas.csv planos.csv`
- Ficheiros (TSV): `./prog -tsv dados.tsv dietas.tsv planos.tsv`
- Ficheiros (CSV|bin): `./prog -csv dados.csv dietas_csv.bin planos.csv`
- Ficheiros (TSV|bin): `./prog -tsv dados.tsv dietas_tsv.bin planos_tsv.bin`

1.9 Conclusão

O presente relatório é parte integrante da avaliação do trabalho prático da Unidade Curricular Laboratórios de Informática.

Ao longo do projeto, enfrentamos a decisão crucial de não apenas utilizar ficheiros CSV, mas também de incorporar ficheiros TSV, que são delimitados por tabulações, e posteriormente, ficheiros no formato binário.

Durante o desenvolvimento, deparamo-nos com a necessidade de uma abordagem estruturada para manipular variáveis em funções. A decisão coletiva de adotar "struct" conferiu-nos flexibilidade, permitindo o acesso a variáveis em qualquer ponto do código. Esta escolha contribuiu significativamente para a clareza e organização do nosso código.

Uma das etapas que se revelou particularmente desafiadora foi a elaboração do relatório em LaTeX, uma tarefa que conseguimos concluir com êxito.

Este projeto não apenas aprimorou as nossas habilidades técnicas, mas também destacou a importância de decisões ponderadas e flexibilidade durante o processo de desenvolvimento. Cada desafio superado contribuiu para uma compreensão mais profunda das melhores práticas de programação e promoveu um ambiente colaborativo e eficiente na execução do projeto em grupo.

Embora tenha sido uma etapa desafiadora, acreditamos que alcançamos não só os objetivos propostos, mas também os prazos definidos.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Data Structure Documentation

4.1 Dados Struct Reference

Esta estrutura contém arrays para armazenar o número do cliente, o nome do cliente e o número de telefone associado a cada cliente.

`#include <struct.h>`

Data Fields

- int n_cliente [min]
- char nome_cliente [max][max]
- int num_telefone [min]

4.1.1 Detailed Description

Esta estrutura contém arrays para armazenar o número do cliente, o nome do cliente e o número de telefone associado a cada cliente.

4.1.2 Field Documentation

4.1.2.1 n_cliente

`int n_cliente[min]`

Array para armazenar o número do cliente.

4.1.2.2 nome_cliente

`char nome_cliente[max][max]`

Matriz para armazenar o nome do cliente.

4.1.2.3 num_telefone

```
int num_telefone[min]
```

Array para armazenar o número de telefone.

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

- /home/hugoc/d-30-1/src/struct.h

4.2 Dieta Struct Reference

Esta estrutura contém arrays para armazenar o número do cliente, tipos de refeição, tipos de alimento, calorias, e a data de cada refeição.

```
#include <struct.h>
```

Data Fields

- int n_cliente [min]
- char refeicao [max][max]
- char alimento [max][max]
- int calorias [max]
- int dia [max]
- int mes [max]
- int ano [max]

4.2.1 Detailed Description

Esta estrutura contém arrays para armazenar o número do cliente, tipos de refeição, tipos de alimento, calorias, e a data de cada refeição.

4.2.2 Field Documentation

4.2.2.1 alimento

```
char alimento[max][max]
```

Matriz para armazenar o tipo de alimento.

4.2.2.2 ano

```
int ano[max]
```

Array para armazenar o ano da entrada na dieta.

4.2.2.3 calorias

```
int calorias[max]
```

Array para armazenar a quantidade de calorias.

4.2.2.4 dia

```
int dia[max]
```

Array para armazenar o dia da entrada na dieta.

4.2.2.5 mes

```
int mes[max]
```

Array para armazenar o mês da entrada na dieta.

4.2.2.6 n_cliente

```
int n_cliente[min]
```

Array para armazenar o número do cliente.

4.2.2.7 refeicao

```
char refeicao[max][max]
```

Matriz para armazenar o tipo de refeição.

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

- /home/hugoc/d-30-1/src/struct.h

4.3 Plano Struct Reference

Esta estrutura contém arrays para armazenar o número do cliente, data tipos de refeição, calorias minimas e maximas.

```
#include <struct.h>
```

Data Fields

- int n_cliente [min]
- int dia [max]
- int mes [max]
- int ano [max]
- char refeicao [max][max]
- int calorias_min [min]
- int calorias_max [min]

4.3.1 Detailed Description

Esta estrutura contém arrays para armazenar o número do cliente, data tipos de refeição, calorias minimas e maximas.

4.3.2 Field Documentation

4.3.2.1 ano

```
int ano[max]
```

Array que armazena os anos do plano.

4.3.2.2 calorias_max

```
int calorias_max[min]
```

Array que armazena as calorias máximas permitidas.

4.3.2.3 calorias_min

```
int calorias_min[min]
```

Array que armazena as calorias mínimas permitidas.

4.3.2.4 dia

```
int dia[max]
```

Array que armazena os dias do plano.

4.3.2.5 mes

```
int mes[max]
```

Array que armazena os meses do plano.

4.3.2.6 n_cliente

```
int n_cliente[min]
```

Número do cliente associado ao plano.

4.3.2.7 refeicao

```
char refeicao[max][max]
```

Matriz que armazena as refeições do plano.

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

- /home/hugoc/d-30-1/src/struct.h

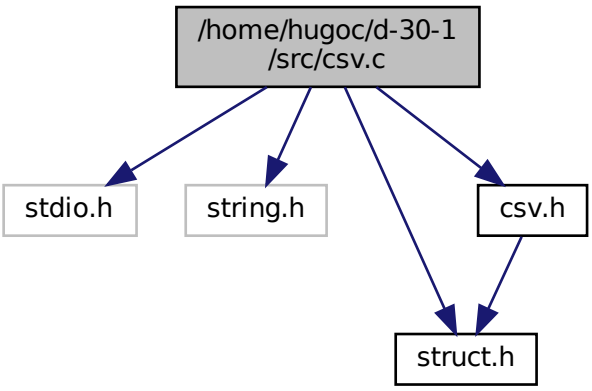
Chapter 5

File Documentation

5.1 /home/hugoc/d-30-1/src/csv.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "struct.h"
#include "csv.h"
```

Include dependency graph for csv.c:



Functions

- void ficheiro_csv_1 (char *arquivo, Dados *dados)

  Verifica os nomes dos ficheiros, caso sejam os ficheiros corretos, entra e verifica as condições, caso contrario, retorna uma mensagem de erro.
- void ficheiro_csv_2 (char *arquivo, Dieta *dieta)

  Lê dados de um arquivo CSV e os armazena na estrutura Dieta.
- void ficheiro_csv_3 (char *arquivo, Plano *plano)

  Lê dados de um arquivo CSV e bin e armazena os na estrutura Plano.

- void listar_plano_csv (Plano *plano, int diainicio, int diafim, int mesinicio, int mesfim)

  Está função é responsável por listar um cliente existente no plano, a escolha do utilizador para ficheiros CSV.
- void calcmedia_csv (Dados *dados, Dieta *dieta, int diainicio, int mesinicio, int diafim, int mesfim)

  Calcula a média de calorias consumidas por cliente num período de tempo para ficheiros CSV.
- void tabela_csv (Dados *dados, Dieta *dieta, Plano *plano)

  Tabela com todas as informações do plano nutricional para ficheiros CSV.

5.1.1 Function Documentation

5.1.1.1 calcmedia_csv()

```
void calcmedia_csv (
            Dados * dados,
            Dieta * dieta,
            int diainicio,
            int mesinicio,
            int diafim,
            int mesfim )
```

Calcula a média de calorias consumidas por cliente num período de tempo para ficheiros CSV.

Inicialmente, a mesma verifica se as datas estão dentro de intervalos válidos, restringindo assim alguns dos clientes, em seguida verifica o numero de cliente e a refeição, onde realiza a soma das calorias e a contagem de vezes que o cliente realizou a refeição. Por fim caso as calorias sejam diferentes de 0, realiza a média das calorias consumidas por cliente.

Parameters

plano	Invocação da struct plano na função.
diainicio	Invocação do dia de início defenido.
mesinicio	Invocação do mes de início defenido.
diafim	Invocação do dia de fim defenido.
mesfim	Invocação do mes de fim defenido.

```
316 {
317     int i = 0;                                                  //**< Responsável por percorrer o array*/
318     float c1_pequeno = 0, c1_almoco = 0, c1_jantar = 0;         //**< Armazena as contagens */
319     float cal_pequeno1 = 0, cal_almoco1 = 0, cal_jantar1 = 0; //**< Armazena as calorias */
320
321     float c2_pequeno = 0, c2_almoco = 0, c2_jantar = 0;         //**< Armazena as contagens */
322     float cal_pequeno2 = 0, cal_almoco2 = 0, cal_jantar2 = 0; //**< Armazena as calorias */
323
324     float c3_pequeno = 0, c3_almoco = 0, c3_jantar = 0;         //**< Armazena as contagens */
325     float cal_pequeno3 = 0, cal_almoco3 = 0, cal_jantar3 = 0; //**< Armazena as calorias */
326
327     printf("\nMédia de todos os clientes:\n\n");
328
329     for (i = 0; i < 4; i++)
330     {
331         if ((mesinicio < dieta->mes[i] && dieta->mes[i] < mesfim) || //**< Verifica o período  */
332             (mesinicio == dieta->mes[i] && diainicio <= dieta->dia[i] && dieta->dia[i] <= diafim) ||
333             (mesfim == dieta->mes[i] && diainicio <= dieta->dia[i] && dieta->dia[i] <= diafim))
334         {
335             if (dieta->n_cliente[i] == 1) //**< Verifica o numero de cliente */
336             {
337                 if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição  */
338                 {
339                     cal_pequeno1 = cal_pequeno1 + dieta->calorias[i]; //**< Acumula as calorias  */
```

```
340                     c1_pequeno++;                                       //**< Contador  */
341                 }
342                 if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição  */
343                 {
344                     cal_almoco1 = cal_almoco1 + dieta->calorias[i]; //**< Acumula as calorias  */
345                     c1_almoco++;                                        //**< Contador  */
346                 }
347                 if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição  */
348                 {
349                     cal_jantar1 = cal_jantar1 + dieta->calorias[i]; //**< Acumula as calorias  */
350                     c1_jantar++;                                        //**< Contador  */
351                 }
352             }
353 
354             if (dieta->n_cliente[i] == 2) //**< Verifica o numero de cliente */
355             {
356                 if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição  */
357                 {
358                     cal_pequeno2 = cal_pequeno2 + dieta->calorias[i]; //**< Acumula as calorias  */
359                     c2_pequeno++;                                       //**< Contador  */
360                 }
361                 if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição  */
362                 {
363                     cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Acumula as calorias  */
364                     c2_almoco++;                                        //**< Contador  */
365                 }
366                 if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição  */
367                 {
368                     cal_jantar2 = cal_jantar2 + dieta->calorias[i]; //**< Acumula as calorias  */
369                     c2_jantar++;                                        //**< Contador  */
370                 }
371             }
372 
373             if (dieta->n_cliente[i] == 3) //**< Verifica o numero de cliente */
374             {
375                 if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição  */
376                 {
377                     cal_pequeno3 = cal_pequeno3 + dieta->calorias[i]; //**< Acumula as calorias  */
378                     c3_pequeno++;                                       //**< Contador  */
379                 }
380                 if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição  */
381                 {
382                     cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Acumula as calorias  */
383                     c3_almoco++;                                        //**< Contador  */
384                 }
385                 if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição  */
386                 {
387                     cal_jantar3 = cal_jantar3 + dieta->calorias[i]; //**< Acumula as calorias  */
388                     c3_jantar++;                                        //**< Contador  */
389                 }
390             }
391         }
392     }
393 
394     // Cliente 1
395     if (cal_pequeno1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
396     {
397         float media_cal_pequeno1 = cal_pequeno1 / c1_pequeno;
              //**< Realiza a media   */
398         printf("\tMédia das calorias consumidas pelo cliente 1 ao pequeno almoço: %.2f\n",
    media_cal_pequeno1); //**< Apresenta a media  */
399     }
400 
401     if (cal_almoco1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
402     {
403         float media_cal_almoco1 = cal_almoco1 / c1_almoco;
    //**< Realiza a media   */
404         printf("\tMédia das calorias consumidas pelo cliente 1 ao almoço: %.2f\n", media_cal_almoco1);
    //**< Apresenta a media  */
405     }
406 
407     if (cal_jantar1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
408     {
409         float media_cal_jantar1 = cal_jantar1 / c1_jantar;
    //**< Realiza a media   */
410         printf("\tMédia das calorias consumidas pelo cliente 1 ao jantar: %.2f\n", media_cal_jantar1);
    //**< Apresenta a media  */
411     }
412 
413     // Cliente 2
414     if (cal_pequeno2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
415     {
416         float media_cal_pequeno2 = cal_pequeno2 / c2_pequeno;
              //**< Realiza a media   */
417         printf("\tMédia das calorias consumidas pelo cliente 2 ao pequeno almoço: %.2f\n",
    media_cal_pequeno2); //**< Apresenta a media  */
418     }
```

```
419
420     if (cal_almoco2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
421     {
422         float media_cal_almoco2 = cal_almoco2 / c2_almoco;
      //**< Realiza a media   */
423         printf("\tMédia das calorias consumidas pelo cliente 2 ao almoço: %.2f\n", media_cal_almoco2);
      //**< Apresenta a media  */
424     }
425
426     if (cal_jantar2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
427     {
428         float media_cal_jantar2 = cal_jantar2 / c2_jantar;
      //**< Realiza a media   */
429         printf("\tMédia das calorias consumidas pelo cliente 2 ao jantar: %.2f\n", media_cal_jantar2);
      //**< Apresenta a media  */
430     }
431
432     // Cliente 3
433     if (cal_pequeno3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
434     {
435         float media_cal_pequeno3 = cal_pequeno3 / c3_pequeno;
            //**< Realiza a media   */
436         printf("\tMédia das calorias consumidas pelo cliente 3 ao pequeno almoço: %.2f\n",
      media_cal_pequeno3); //**< Apresenta a media  */
437     }
438
439     if (cal_almoco3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
440     {
441         float media_cal_almoco3 = cal_almoco3 / c3_almoco;
      //**< Realiza a media   */
442         printf("\tMédia das calorias consumidas pelo cliente 3 ao almoço: %.2f\n", media_cal_almoco3);
      //**< Apresenta a media  */
443     }
444
445     if (cal_jantar3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
446     {
447         float media_cal_jantar3 = cal_jantar3 / c3_jantar;
      //**< Realiza a media   */
448         printf("\tMédia das calorias consumidas pelo cliente 3 ao jantar: %.2f\n", media_cal_jantar3);
      //**< Apresenta a media  */
449     }
450 }
```

5.1.1.2 ficheiro_csv_1()

```
void ficheiro_csv_1 (
            char * arquivo,
            Dados * dados )
```

Verifica os nomes dos ficheiros, caso sejam os ficheiros corretos, entra e verifica as condições, caso contrario, retorna uma mensagem de erro.

Parameters

arquivo	Nome do arquivo CSV e bin a ser lido na função.
dados	Invocação da struct dados na função.

While: percorre o arquivo CSV, lê os dados até ao ';' e armazena-os na struct Dados.

Parameters

ficheiro	Nome do arquivo CSV a ser lido na função.
dados	Invocação da struct dados na função.
i	Variável para percorrer o array.

For: vai realizar a leitura dos dados do arquivo binário e armazená-los na struct Dados.

For: Apresenta os dados armazenados na struct Dados.

```
17 {
18     if (strcmp(arquivo, "dados.csv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
       */
19     {
20
21         int i = 0;
22         FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, com o nome ficheiro.
       */
23
24         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
25         {
26             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
27             return;                                               //**< Retorno indicando falha na
       abertura do arquivo.*/
28         }
29
30         printf("\nDados clientes:\n\n");
31
40         while (fscanf(ficheiro, "%d;%[^;];%d\n", &dados->n_cliente[i], dados->nome_cliente[i],
       &dados->num_telefone[i]) == 3)
41         {
42             printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
       dados->num_telefone[i]);
43             i++;
44         }
45
46         fclose(ficheiro); //**< Fecha o arquivo. */
47     }
48
49     else if (strcmp(arquivo, "dados_csv.bin") == 0) //**< Compara as strings, caso sejam iguais entra
       neste loop */
50     {
51         int i = 0;
52
53         FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
       nome ficheiro. */
54
55         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
56         {
57             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
58             return;                                               //**< Retorno indicando falha na
       abertura do arquivo.*/
59         }
60         printf("\nDados clientes:\n\n");
61
66         for (int i = 0; i < 3; i++)
67         {
68             fread(&dados->n_cliente[i], sizeof(int), 1, ficheiro);      //**< Modo leitura relativamente
       a ficheiros binários */
69             fread(dados->nome_cliente[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente
       a dados binários do tipo string */
70             fread(&dados->num_telefone[i], sizeof(int), 1, ficheiro);   //**< Modo leitura relativamente
       a dados binários interios */
71         }
76         for (int i = 0; i < 3; i++)
77         {
78             printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
       dados->num_telefone[i]);
79         }
80
81         fclose(ficheiro); //**< Fecha o arquivo. */
82     }
83
84     else //**< Outros casos  */
85     {
86         printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro  */
87         return;
88     }
89 }
```

5.1.1.3 ficheiro_csv_2()

```
void ficheiro_csv_2 (
            char * arquivo,
            Dieta * dieta )
```

Lê dados de um arquivo CSV e os armazena na estrutura Dieta.

Parameters

arquivo	Nome do arquivo CSV a ser lido na função.
dieta	Invocação da struct dieta na função.

Extrai dados da linha formatada do arquivo CSV e os armazena na estrutura Dieta.

Parameters

linha	Nome da linha do arquivo CSV a ser lido na função.
dieta	Invocação da struct dieta na função.
i	Variável para percorrer o array.

For: vai realizar a leitura dos dados do arquivo binário e armazená-los na struct Dados.

For: Apresenta os dados armazenados na struct Dados.

```
97 {
98      if (strcmp(arquivo, "dietas.csv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
        */
99      {
100
101         FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
        ficheiro. */
102         char linha[max];                      //**< Tamanho para armazenar cada linha do arquivo. */
103
104         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
105         {
106             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
107             return;                                               //**< Retorno indicando falha na
        abertura do arquivo.*/
108         }
109
110         printf("\nDieta realizada pelo cliente:\n\n");
111
112         for (int i = 0; i < max; i++)
113         {
114             if (fgets(linha, sizeof(linha), ficheiro) == NULL) //**< Verifica se as linhas do arquivo
        CSV estão vazias.Verifica se as linhas do arquivo CSV estão vazias. */
115             {
116                 break; //**< Sai do loop se não houver mais linhas para ler. */
117             }
118
126             sscanf(linha, "%d;%d-%d-%d;%[^;];%[^;];%d ", &dieta->n_cliente[i], &dieta->dia[i],
        &dieta->mes[i], &dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], &dieta->calorias[i]);
127             printf("\t%04d %02d-%02d-%d %s %s %d \n", dieta->n_cliente[i], dieta->dia[i], dieta->mes[i],
        dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], dieta->calorias[i]);
128         }
129
130         fclose(ficheiro); //**< Fecha o arquivo após a leitura dos dados. */
131     }
132
133     else if (strcmp(arquivo, "dietas_csv.bin") == 0) //**< Compara as strings, caso sejam iguais entra
        neste loop */
134     {
135         FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
        nome ficheiro. */
136
137         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
138         {
139             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
140             return;                                               //**< Retorno indicando falha na
        abertura do arquivo.*/
141         }
142
143         printf("\nDieta realizada pelo cliente:\n\n");
144
149         for (int i = 0; i < 4; i++)
150         {
151             fread(&dieta->n_cliente[i], sizeof(int), 1, ficheiro);  //**< Modo leitura relativamente a
        dados binários interios */
152             fread(&dieta->dia[i], sizeof(int), 1, ficheiro);        //**< Modo leitura relativamente a
        dados binários interios */
153             fread(&dieta->mes[i], sizeof(int), 1, ficheiro);        //**< Modo leitura relativamente a
        dados binários interios */
```

```
154             fread(&dieta->ano[i], sizeof(int), 1, ficheiro);        //**< Modo leitura relativamente a
      dados binários interios */
155             fread(dieta->refeicao[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a
      dados binários do tipo string */
156             fread(dieta->alimento[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a
      dados binários do tipo string */
157             fread(&dieta->calorias[i], sizeof(int), 1, ficheiro);   //**< Modo leitura relativamente a
      dados binários interios */
158         }
163         for (int i = 0; i < 4; i++)
164         {
165             printf("\t%04d %02d-%02d-%d %s %s %d \n", dieta->n_cliente[i], dieta->dia[i], dieta->mes[i],
      dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], dieta->calorias[i]);
166         }
167 
168         fclose(ficheiro); //**< Fecha o arquivo. */
169         return;
170     }
171     else //**< Outros casos  */
172     {
173         printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro  */
174         return;
175     }
176 }
```

5.1.1.4 ficheiro_csv_3()

```
void ficheiro_csv_3 (
              char * arquivo,
              Plano * plano )
```

Lê dados de um arquivo CSV e bin e armazena os na estrutura Plano.

Parameters

arquivo	Nome do arquivo (CSV|.bin) a ser lido na função.
dieta	Invocação da struct dieta na função.

Extrai dados da linha formatada do arquivo CSV e os armazena na estrutura Dieta.

Parameters

linha	Nome da linha do arquivo CSV a ser lido na função.
dieta	Invocação da struct dieta na função.
i	Variável para percorrer o array.

For: vai realizar a leitura dos dados do arquivo binário e armazená-los na struct Dados.

For: Apresenta os dados armazenados na struct Dados.

```
185 {
186     if (strcmp(arquivo, "planos.csv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
      */
187     {
188         FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
      ficheiro. */
189         char linha[100];                      //**< Tamanho para armazenar cada linha do arquivo. */
190 
191         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
192         {
193             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
194             return;                                               //**< Retorno indicando falha na
      abertura do arquivo.*/
195         }
```

```
196
197         printf("\nPlano nutricional:\n\n");
198
199         for (int i = 0; i < 100; i++)
200         {
201             if (fgets(linha, sizeof(linha), ficheiro) == NULL) //**< Verifica se as linhas do arquivo
    CSV estão vazias.Verifica se as linhas do arquivo CSV estão vazias. */
202             {
203                 break; //**< Sai do loop se não houver mais linhas para ler. */
204             }
205
214             sscanf(linha, "%d;%d-%d-%d;%[^;];%d Cal, %d Cal", &plano->n_cliente[i], &plano->dia[i],
    &plano->mes[i], &plano->ano[i], plano->refeicao[i], &plano->calorias_min[i],
    &plano->calorias_max[i]);
215             printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal \n", plano->n_cliente[i], plano->dia[i],
    plano->mes[i], plano->ano[i], plano->refeicao[i], plano->calorias_min[i], plano->calorias_max[i]);
216         }
217         fclose(ficheiro); //**< Fecha o arquivo após a leitura dos dados. */
218     }
219     else if (strcmp(arquivo, "planos_csv.bin") == 0) //**< Compara as strings, caso sejam iguais entra
    neste loop */
220     {
221         FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
    nome ficheiro. */
222
223         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
224         {
225             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
226             return;                                               //**< Retorno indicando falha na
    abertura do arquivo.*/
227         }
228
229         printf("\nPlano nutricional:\n\n");
230
235         for (int i = 0; i < 3; i++)
236         {
237             fread(&plano->n_cliente[i], sizeof(int), 1, ficheiro);    //**< Modo leitura relativamente a
    dados binários interios */
238             fread(&plano->dia[i], sizeof(int), 1, ficheiro);          //**< Modo leitura relativamente a
    dados binários interios */
239             fread(&plano->mes[i], sizeof(int), 1, ficheiro);          //**< Modo leitura relativamente a
    dados binários interios */
240             fread(&plano->ano[i], sizeof(int), 1, ficheiro);          //**< Modo leitura relativamente a
    dados binários interios */
241             fread(plano->refeicao[i], sizeof(char), max, ficheiro);   //**< Modo leitura relativamente a
    dados binários do tipo string */
242             fread(&plano->calorias_min[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a
    dados binários interios */
243             fread(&plano->calorias_max[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a
    dados binários interios */
244         }
248         for (int i = 0; i < 3; i++)
249         {
250             printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal \n", plano->n_cliente[i], plano->dia[i],
    plano->mes[i], plano->ano[i], plano->refeicao[i], plano->calorias_min[i], plano->calorias_max[i]);
251         }
252
253         fclose(ficheiro); //**< Fecha o arquivo. */
254     }
255     else //**< Outros casos  */
256     {
257         printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro  */
258         return;
259     }
260 }
```

5.1.1.5 listar_plano_csv()

```
void listar_plano_csv (
            Plano * plano,
            int diainicio,
            int diafim,
            int mesinicio,
            int mesfim )
```

Está função é responsável por listar um cliente existente no plano, a escolha do utilizador para ficheiros CSV.

Realiza a leitura do número do cliente e da refeição que o utilizador pretende visualizar, verifica o periodo em seguida compra cliente e refeição com os dados do plano, caso sejam verdadeiros apresenta os dados do cliente caso sejam falsa aprenta uma mensagem de erro.

Parameters

plano	Invocação da struct plano na função.
diainicio	Invocação do dia de início defenido.
mesinicio	Invocação do mes de início defenido.
diafim	Invocação do dia de fim defenido.
mesfim	Invocação do mes de fim defenido.

```
276 {
277     int cliente;       //**< Armazena nº cliente*/
278     char refeicao[20]; //**< Armazena o nome do cliente*/
279 
280     printf("\nDigite o cliente e a refeição que pretende visualizar (numero refeição):");
281     scanf("%d%[^\n]", &cliente, refeicao); //**< Realiza a leitura */
282 
283     printf("\n  Plano nutricional do cliente %04d, refeição:%s\n\n", cliente, refeicao); //**< Apresenta
      os clientes selecionados */
284 
285     for (int i = 0; i < max; i++)
286     {
287         if ((mesinicio < plano->mes[i] && plano->mes[i] < mesfim) || //**< Verifica o período se for
      válido entra no loop */
288             (mesinicio == plano->mes[i] && diainicio <= plano->dia[i] && plano->dia[i] <= diafim) ||
289             (mesfim == plano->mes[i] && diainicio <= plano->dia[i] && plano->dia[i] <= diafim))
290         {
291             if (cliente == plano->n_cliente[i]) //**< Verifica se o cliente existe */
292             {
293                 if (strcmp(refeicao, plano->refeicao[i]) == 0) //**< Verifica se a refeição existe */
294                 {
295                     printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal\n", plano->n_cliente[i],
      plano->dia[i], plano->mes[i], plano->ano[i], plano->refeicao[i], plano->calorias_min[i],
      plano->calorias_max[i]); //**< Apresenta o plano do cliente selecionado */
296                 }
297             }
298         }
299     }
300 }
```

5.1.1.6 tabela_csv()

```
void tabela_csv (
             Dados * dados,
             Dieta * dieta,
             Plano * plano )
```

Tabela com todas as informações do plano nutricional para ficheiros CSV.

Inicialmente, a mesma verifica o numero de cliente e a refeição, dentro dessas condições soma as calorias e por cliente e refeição. Por fim verifica o numero de cliente e a refeição, e apresenta as calorias acumuladas por cliente.

Parameters

dados	Invocação da struct dados na função.
dieta	Invocação da struct dieta na função.
plano	Invocação da struct plano na função.

```
463 {
464     float cal_pequeno1 = 0, cal_almoco1 = 0, cal_jantar1 = 0; //**< Armazena as calorias cliente 1*/
465     float cal_pequeno2 = 0, cal_almoco2 = 0, cal_jantar2 = 0; //**< Armazena as calorias cliente 2*/
```

```
466     float cal_pequeno3 = 0, cal_almoco3 = 0, cal_jantar3 = 0; //**< Armazena as calorias cliente 3*/
467
468     for (int i = 0; i < 6; i++)
469     {
470
471         if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */
472         {
473             if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição */
474             {
475                 cal_pequeno1 = cal_pequeno1 + dieta->calorias[i]; //**< Soma as calorias */
476             }
477             if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
478             {
479                 cal_almoco1 = cal_almoco1 + dieta->calorias[i]; //**< Soma as calorias */
480             }
481             if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */
482             {
483                 cal_jantar1 = cal_jantar1 + dieta->calorias[i]; //**< Soma as calorias */
484             }
485         }
486
487         if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */
488         {
489             if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição */
490             {
491                 cal_pequeno2 = cal_pequeno2 + dieta->calorias[i]; //**< Soma as calorias */
492             }
493             if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
494             {
495                 cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Soma as calorias */
496             }
497             if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */
498             {
499                 cal_jantar2 = cal_jantar2 + dieta->calorias[i]; //**< Soma as calorias */
500             }
501         }
502
503         if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */
504         {
505             if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição */
506             {
507                 cal_pequeno3 = cal_pequeno3 + dieta->calorias[i]; //**< Soma as calorias */
508             }
509             if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
510             {
511                 cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Soma as calorias */
512             }
513             if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */
514             {
515                 cal_jantar3 = cal_jantar3 + dieta->calorias[i]; //**< Soma as calorias */
516             }
517         }
518     }
519
520     printf("\nTabela:\n\n");
521
    printf("\t-------------------------------------------------------------------------------------------------\n");
522     printf("\t|   NP   | Paciente |   Tipo Refeição  |   Inicio   |    Fim     | Mínimo | Máximo  |
    Consumo |\n");
523
    printf("\t|--------|----------|------------------|------------|------------|--------|---------|---------|\n");
524
525     for (int i = 0; i < 3; i++)
526     {
527         for (int j = 0; j < 3; j++)
528         {
529             if (dados->n_cliente[i] == plano->n_cliente[j]) //**< Responsável por verificar os clientes
    que contém planos, para realizar a aprsentação do nome correto  */
530             {
531
532                 if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */
533                 {
534                     if (strcmp(plano->refeicao[j], " jantar") == 0) //**< Verifica a refeição */
535                     {
536                         printf("\t|  %04d  |   %s   | %-16s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f   |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_jantar1); //**< Apresenta linha da tabela */
537                     }
538                     else if (strcmp(plano->refeicao[j], " almoço") == 0) //**< Verifica a refeição */
539                     {
540                         printf("\t|  %04d  |   %s   | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_almoco1); //**< Apresenta linha da tabela */
541                     }
542                     else if (strcmp(plano->refeicao[j], " pequeno almoço") == 0) //**< Verifica a
```

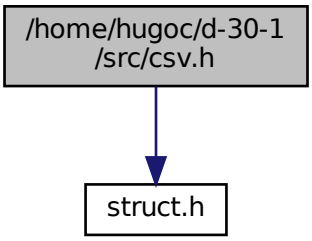
```
    refeição */
543                     {
544                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_pequeno1); //**< Apresenta linha da tabela */
545                     }
546                     else //**< Outros casos */
547                     {
548                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   | -----\n ", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j]); //**<
    Apresenta linha da tabela */
549                     }
550                 }
551                 if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */
552                 {
553                     if (strcmp(plano->refeicao[j], " jantar") == 0) //**< Verifica a refeição */
554                     {
555
556                         printf("\t|  %04d  |   %s  | %-16s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f   |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_jantar2); //**< Apresenta linha da tabela */
557                     }
558                     else if (strcmp(plano->refeicao[j], " almoço") == 0) //**< Verifica a refeição */
559                     {
560                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_almoco2); //**< Apresenta linha da tabela */
561                     }
562                     else if (strcmp(plano->refeicao[j], " pequeno almoço") == 0) //**< Verifica a
    refeição */
563                     {
564                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_pequeno2); //**< Apresenta linha da tabela */
565                     }
566                     else //**< Outros casos */
567                     {
568                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   | -----\n ", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j]); //**<
    Apresenta linha da tabela */
569                     }
570                 }
571
572                 if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */
573                 {
574                     if (strcmp(plano->refeicao[j], " jantar") == 0) //**< Verifica a refeição */
575                     {
576
577                         printf("\t|  %04d  |   %s  | %-16s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f   |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_jantar3); //**< Apresenta linha da tabela */
578                     }
579                     else if (strcmp(plano->refeicao[j], " almoço") == 0) //**< Verifica a refeição */
580                     {
581                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_almoco3); //**< Apresenta linha da tabela */
582                     }
583                     else if (strcmp(plano->refeicao[j], " pequeno almoço") == 0) //**< Verifica a
    refeição */
584                     {
585                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_pequeno3); //**< Apresenta linha da tabela */
586                     }
587                     else //**< Outros casos */
588                     {
589                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   | -----\n ", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j]); //**<
    Apresenta linha da tabela */
590                     }
591                 }
592             }
593         }
594     }
595
    printf("\t-------------------------------------------------------------------------------------------\n");
```

```
596 }
```

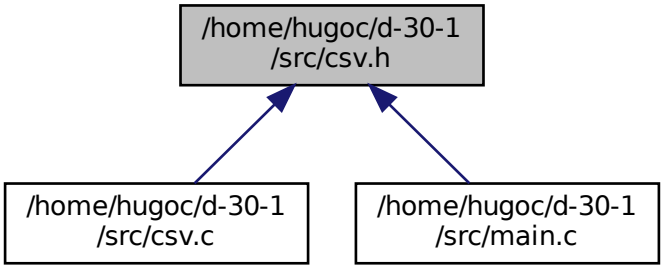
5.2 /home/hugoc/d-30-1/src/csv.h File Reference

```
#include "struct.h"
```

Include dependency graph for csv.h:



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



Functions

- void ficheiro_csv_1 (char ∗arquivo, Dados ∗dados)

  Verifica os nomes dos ficheiros, caso sejam os ficheiros corretos, entra e verifica as condições, caso contrario, retorna uma mensagem de erro.
- void ficheiro_csv_2 (char ∗arquivo, Dieta ∗dieta)

  Lê dados de um arquivo CSV e os armazena na estrutura Dieta.
- void ficheiro_csv_3 (char ∗arquivo, Plano ∗plano)

  Lê dados de um arquivo CSV e bin e armazena os na estrutura Plano.
- void listar_plano_csv (Plano ∗plano, int diainicio, int diafim, int mesinicio, int mesfim)

Está função é responsável por listar um cliente existente no plano, a escolha do utilizador para ficheiros CSV.

- void calcmedia_csv (Dados *dados, Dieta *dieta, int diainicio, int mesinicio, int diafim, int mesfim)

  Calcula a média de calorias consumidas por cliente num período de tempo para ficheiros CSV.

- void tabela_csv (Dados *dados, Dieta *dieta, Plano *plano)

  Tabela com todas as informações do plano nutricional para ficheiros CSV.

5.2.1 Function Documentation

5.2.1.1 calcmedia_csv()

```
void calcmedia_csv (
            Dados * dados,
            Dieta * dieta,
            int diainicio,
            int mesinicio,
            int diafim,
            int mesfim )
```

Calcula a média de calorias consumidas por cliente num período de tempo para ficheiros CSV.

Inicialmente, a mesma verifica se as datas estão dentro de intervalos válidos, restringindo assim alguns dos clientes, em seguida verifica o numero de cliente e a refeição, onde realiza a soma das calorias e a contagem de vezes que o cliente realizou a refeição. Por fim caso as calorias sejam diferentes de 0, realiza a média das calorias consumidas por cliente.

Parameters

plano	Invocação da struct plano na função.
diainicio	Invocação do dia de início defenido.
mesinicio	Invocação do mes de início defenido.
diafim	Invocação do dia de fim defenido.
mesfim	Invocação do mes de fim defenido.

```
316 {
317     int i = 0;                                                  //**< Responsável por percorrer o array*/
318     float c1_pequeno = 0, c1_almoco = 0, c1_jantar = 0;         //**< Armazena as contagens */
319     float cal_pequeno1 = 0, cal_almoco1 = 0, cal_jantar1 = 0; //**< Armazena as calorias */
320
321     float c2_pequeno = 0, c2_almoco = 0, c2_jantar = 0;         //**< Armazena as contagens */
322     float cal_pequeno2 = 0, cal_almoco2 = 0, cal_jantar2 = 0; //**< Armazena as calorias */
323
324     float c3_pequeno = 0, c3_almoco = 0, c3_jantar = 0;         //**< Armazena as contagens */
325     float cal_pequeno3 = 0, cal_almoco3 = 0, cal_jantar3 = 0; //**< Armazena as calorias */
326
327     printf("\nMédia de todos os clientes:\n\n");
328
329     for (i = 0; i < 4; i++)
330     {
331         if ((mesinicio < dieta->mes[i] && dieta->mes[i] < mesfim) || //**< Verifica o período  */
332             (mesinicio == dieta->mes[i] && diainicio <= dieta->dia[i] && dieta->dia[i] <= diafim) ||
333             (mesfim == dieta->mes[i] && diainicio <= dieta->dia[i] && dieta->dia[i] <= diafim))
334         {
335             if (dieta->n_cliente[i] == 1) //**< Verifica o numero de cliente */
336             {
337                 if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição  */
338                 {
339                     cal_pequeno1 = cal_pequeno1 + dieta->calorias[i]; //**< Acumula as calorias  */
340                     c1_pequeno++;                                     //**< Contador  */
341                 }
```

```
342                 if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição  */
343                 {
344                     cal_almoco1 = cal_almoco1 + dieta->calorias[i]; //**< Acumula as calorias  */
345                     c1_almoco++;                                    //**< Contador  */
346                 }
347                 if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição  */
348                 {
349                     cal_jantar1 = cal_jantar1 + dieta->calorias[i]; //**< Acumula as calorias  */
350                     c1_jantar++;                                    //**< Contador  */
351                 }
352             }
353 
354             if (dieta->n_cliente[i] == 2) //**< Verifica o numero de cliente */
355             {
356                 if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição  */
357                 {
358                     cal_pequeno2 = cal_pequeno2 + dieta->calorias[i]; //**< Acumula as calorias  */
359                     c2_pequeno++;                                     //**< Contador  */
360                 }
361                 if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição  */
362                 {
363                     cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Acumula as calorias  */
364                     c2_almoco++;                                    //**< Contador  */
365                 }
366                 if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição  */
367                 {
368                     cal_jantar2 = cal_jantar2 + dieta->calorias[i]; //**< Acumula as calorias  */
369                     c2_jantar++;                                    //**< Contador  */
370                 }
371             }
372 
373             if (dieta->n_cliente[i] == 3) //**< Verifica o numero de cliente */
374             {
375                 if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição  */
376                 {
377                     cal_pequeno3 = cal_pequeno3 + dieta->calorias[i]; //**< Acumula as calorias  */
378                     c3_pequeno++;                                     //**< Contador  */
379                 }
380                 if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição  */
381                 {
382                     cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Acumula as calorias  */
383                     c3_almoco++;                                    //**< Contador  */
384                 }
385                 if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição  */
386                 {
387                     cal_jantar3 = cal_jantar3 + dieta->calorias[i]; //**< Acumula as calorias  */
388                     c3_jantar++;                                    //**< Contador  */
389                 }
390             }
391         }
392     }
393 
394     // Cliente 1
395     if (cal_pequeno1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
396     {
397         float media_cal_pequeno1 = cal_pequeno1 / c1_pequeno;
          //**< Realiza a media   */
398         printf("\tMédia das calorias consumidas pelo cliente 1 ao pequeno almoço: %.2f\n",
    media_cal_pequeno1); //**< Apresenta a media  */
399     }
400 
401     if (cal_almoco1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
402     {
403         float media_cal_almoco1 = cal_almoco1 / c1_almoco;
    //**< Realiza a media   */
404         printf("\tMédia das calorias consumidas pelo cliente 1 ao almoço: %.2f\n", media_cal_almoco1);
    //**< Apresenta a media  */
405     }
406 
407     if (cal_jantar1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
408     {
409         float media_cal_jantar1 = cal_jantar1 / c1_jantar;
    //**< Realiza a media   */
410         printf("\tMédia das calorias consumidas pelo cliente 1 ao jantar: %.2f\n", media_cal_jantar1);
    //**< Apresenta a media  */
411     }
412 
413     // Cliente 2
414     if (cal_pequeno2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
415     {
416         float media_cal_pequeno2 = cal_pequeno2 / c2_pequeno;
          //**< Realiza a media   */
417         printf("\tMédia das calorias consumidas pelo cliente 2 ao pequeno almoço: %.2f\n",
    media_cal_pequeno2); //**< Apresenta a media  */
418     }
419 
420     if (cal_almoco2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
```

```
421     {
422         float media_cal_almoco2 = cal_almoco2 / c2_almoco;
      //**< Realiza a media   */
423         printf("\tMédia das calorias consumidas pelo cliente 2 ao almoço: %.2f\n", media_cal_almoco2);
      //**< Apresenta a media  */
424     }
425
426     if (cal_jantar2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
427     {
428         float media_cal_jantar2 = cal_jantar2 / c2_jantar;
      //**< Realiza a media   */
429         printf("\tMédia das calorias consumidas pelo cliente 2 ao jantar: %.2f\n", media_cal_jantar2);
      //**< Apresenta a media  */
430     }
431
432     // Cliente 3
433     if (cal_pequeno3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
434     {
435         float media_cal_pequeno3 = cal_pequeno3 / c3_pequeno;
            //**< Realiza a media   */
436         printf("\tMédia das calorias consumidas pelo cliente 3 ao pequeno almoço: %.2f\n",
      media_cal_pequeno3); //**< Apresenta a media  */
437     }
438
439     if (cal_almoco3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
440     {
441         float media_cal_almoco3 = cal_almoco3 / c3_almoco;
      //**< Realiza a media   */
442         printf("\tMédia das calorias consumidas pelo cliente 3 ao almoço: %.2f\n", media_cal_almoco3);
      //**< Apresenta a media  */
443     }
444
445     if (cal_jantar3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
446     {
447         float media_cal_jantar3 = cal_jantar3 / c3_jantar;
      //**< Realiza a media   */
448         printf("\tMédia das calorias consumidas pelo cliente 3 ao jantar: %.2f\n", media_cal_jantar3);
      //**< Apresenta a media  */
449     }
450 }
```

5.2.1.2 ficheiro_csv_1()

```
void ficheiro_csv_1 (
            char * arquivo,
            Dados * dados )
```

Verifica os nomes dos ficheiros, caso sejam os ficheiros corretos, entra e verifica as condições, caso contrario, retorna uma mensagem de erro.

Parameters

arquivo	Nome do arquivo CSV e bin a ser lido na função.
dados	Invocação da struct dados na função.

While: percorre o arquivo CSV, lê os dados até ao ';' e armazena-os na struct Dados.

Parameters

ficheiro	Nome do arquivo CSV a ser lido na função.
dados	Invocação da struct dados na função.
i	Variável para percorrer o array.

For: vai realizar a leitura dos dados do arquivo binário e armazená-los na struct Dados.

For: Apresenta os dados armazenados na struct Dados.

```
17 {
18     if (strcmp(arquivo, "dados.csv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
       */
19     {
20
21         int i = 0;
22         FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, com o nome ficheiro.
       */
23
24         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
25         {
26             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
27             return;                                               //**< Retorno indicando falha na
       abertura do arquivo.*/
28         }
29
30         printf("\nDados clientes:\n\n");
31
40         while (fscanf(ficheiro, "%d;%[^;];%d\n", &dados->n_cliente[i], dados->nome_cliente[i],
       &dados->num_telefone[i]) == 3)
41         {
42             printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
       dados->num_telefone[i]);
43             i++;
44         }
45
46         fclose(ficheiro); //**< Fecha o arquivo. */
47     }
48
49     else if (strcmp(arquivo, "dados_csv.bin") == 0) //**< Compara as strings, caso sejam iguais entra
       neste loop */
50     {
51         int i = 0;
52
53         FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
       nome ficheiro. */
54
55         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
56         {
57             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
58             return;                                               //**< Retorno indicando falha na
       abertura do arquivo.*/
59         }
60         printf("\nDados clientes:\n\n");
61
66         for (int i = 0; i < 3; i++)
67         {
68             fread(&dados->n_cliente[i], sizeof(int), 1, ficheiro);      //**< Modo leitura relativamente
       a ficheiros binários */
69             fread(dados->nome_cliente[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente
       a dados binários do tipo string */
70             fread(&dados->num_telefone[i], sizeof(int), 1, ficheiro);   //**< Modo leitura relativamente
       a dados binários interios */
71         }
76         for (int i = 0; i < 3; i++)
77         {
78             printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
       dados->num_telefone[i]);
79         }
80
81         fclose(ficheiro); //**< Fecha o arquivo. */
82     }
83
84     else //**< Outros casos  */
85     {
86         printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro  */
87         return;
88     }
89 }
```

5.2.1.3 ficheiro_csv_2()

```
void ficheiro_csv_2 (
            char * arquivo,
            Dieta * dieta )
```

Lê dados de um arquivo CSV e os armazena na estrutura Dieta.

Parameters

arquivo	Nome do arquivo CSV a ser lido na função.
dieta	Invocação da struct dieta na função.

Extrai dados da linha formatada do arquivo CSV e os armazena na estrutura Dieta.

Parameters

linha	Nome da linha do arquivo CSV a ser lido na função.
dieta	Invocação da struct dieta na função.
i	Variável para percorrer o array.

For: vai realizar a leitura dos dados do arquivo binário e armazená-los na struct Dados.

For: Apresenta os dados armazenados na struct Dados.

```
97 {
98      if (strcmp(arquivo, "dietas.csv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
        */
99      {
100
101         FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
        ficheiro. */
102         char linha[max];                      //**< Tamanho para armazenar cada linha do arquivo. */
103
104         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
105         {
106             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
107             return;                                               //**< Retorno indicando falha na
        abertura do arquivo.*/
108         }
109
110         printf("\nDieta realizada pelo cliente:\n\n");
111
112         for (int i = 0; i < max; i++)
113         {
114             if (fgets(linha, sizeof(linha), ficheiro) == NULL) //**< Verifica se as linhas do arquivo
        CSV estão vazias.Verifica se as linhas do arquivo CSV estão vazias. */
115             {
116                 break; //**< Sai do loop se não houver mais linhas para ler. */
117             }
118
126             sscanf(linha, "%d;%d-%d-%d;%[^;];%[^;];%d ", &dieta->n_cliente[i], &dieta->dia[i],
        &dieta->mes[i], &dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], &dieta->calorias[i]);
127             printf("\t%04d %02d-%02d-%d %s %s %d \n", dieta->n_cliente[i], dieta->dia[i], dieta->mes[i],
        dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], dieta->calorias[i]);
128         }
129
130         fclose(ficheiro); //**< Fecha o arquivo após a leitura dos dados. */
131     }
132
133     else if (strcmp(arquivo, "dietas_csv.bin") == 0) //**< Compara as strings, caso sejam iguais entra
        neste loop */
134     {
135         FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
        nome ficheiro. */
136
137         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
138         {
139             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
140             return;                                               //**< Retorno indicando falha na
        abertura do arquivo.*/
141         }
142
143         printf("\nDieta realizada pelo cliente:\n\n");
144
149         for (int i = 0; i < 4; i++)
150         {
151             fread(&dieta->n_cliente[i], sizeof(int), 1, ficheiro);  //**< Modo leitura relativamente a
        dados binários interios */
152             fread(&dieta->dia[i], sizeof(int), 1, ficheiro);        //**< Modo leitura relativamente a
        dados binários interios */
153             fread(&dieta->mes[i], sizeof(int), 1, ficheiro);        //**< Modo leitura relativamente a
        dados binários interios */
```

```
154             fread(&dieta->ano[i], sizeof(int), 1, ficheiro);        //**< Modo leitura relativamente a
    dados binários interios */
155             fread(dieta->refeicao[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a
    dados binários do tipo string */
156             fread(dieta->alimento[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a
    dados binários do tipo string */
157             fread(&dieta->calorias[i], sizeof(int), 1, ficheiro);   //**< Modo leitura relativamente a
    dados binários interios */
158         }
163         for (int i = 0; i < 4; i++)
164         {
165             printf("\t%04d %02d-%02d-%d %s %s %d \n", dieta->n_cliente[i], dieta->dia[i], dieta->mes[i],
    dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], dieta->calorias[i]);
166         }
167
168         fclose(ficheiro); //**< Fecha o arquivo. */
169         return;
170     }
171     else //**< Outros casos  */
172     {
173         printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro  */
174         return;
175     }
176 }
```

5.2.1.4 ficheiro_csv_3()

```
void ficheiro_csv_3 (
            char * arquivo,
            Plano * plano )
```

Lê dados de um arquivo CSV e bin e armazena os na estrutura Plano.

Parameters

arquivo	Nome do arquivo (CSV|.bin) a ser lido na função.
dieta	Invocação da struct dieta na função.

Extrai dados da linha formatada do arquivo CSV e os armazena na estrutura Dieta.

Parameters

linha	Nome da linha do arquivo CSV a ser lido na função.
dieta	Invocação da struct dieta na função.
i	Variável para percorrer o array.

For: vai realizar a leitura dos dados do arquivo binário e armazená-los na struct Dados.

For: Apresenta os dados armazenados na struct Dados.

```
185 {
186     if (strcmp(arquivo, "planos.csv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
    */
187     {
188         FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
    ficheiro. */
189         char linha[100];                      //**< Tamanho para armazenar cada linha do arquivo. */
190
191         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
192         {
193             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
194             return;                                               //**< Retorno indicando falha na
    abertura do arquivo.*/
195         }
```

```
196
197         printf("\nPlano nutricional:\n\n");
198
199         for (int i = 0; i < 100; i++)
200         {
201             if (fgets(linha, sizeof(linha), ficheiro) == NULL) //**< Verifica se as linhas do arquivo
    CSV estão vazias.Verifica se as linhas do arquivo CSV estão vazias. */
202             {
203                 break; //**< Sai do loop se não houver mais linhas para ler. */
204             }
205
214             sscanf(linha, "%d;%d-%d-%d;%[^;];%d Cal, %d Cal", &plano->n_cliente[i], &plano->dia[i],
    &plano->mes[i], &plano->ano[i], plano->refeicao[i], &plano->calorias_min[i],
    &plano->calorias_max[i]);
215             printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal \n", plano->n_cliente[i], plano->dia[i],
    plano->mes[i], plano->ano[i], plano->refeicao[i], plano->calorias_min[i], plano->calorias_max[i]);
216         }
217         fclose(ficheiro); //**< Fecha o arquivo após a leitura dos dados. */
218     }
219     else if (strcmp(arquivo, "planos_csv.bin") == 0) //**< Compara as strings, caso sejam iguais entra
    neste loop */
220     {
221         FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
    nome ficheiro. */
222
223         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
224         {
225             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
226             return;                                               //**< Retorno indicando falha na
    abertura do arquivo.*/
227         }
228
229         printf("\nPlano nutricional:\n\n");
230
235         for (int i = 0; i < 3; i++)
236         {
237             fread(&plano->n_cliente[i], sizeof(int), 1, ficheiro);    //**< Modo leitura relativamente a
    dados binários interios */
238             fread(&plano->dia[i], sizeof(int), 1, ficheiro);          //**< Modo leitura relativamente a
    dados binários interios */
239             fread(&plano->mes[i], sizeof(int), 1, ficheiro);          //**< Modo leitura relativamente a
    dados binários interios */
240             fread(&plano->ano[i], sizeof(int), 1, ficheiro);          //**< Modo leitura relativamente a
    dados binários interios */
241             fread(plano->refeicao[i], sizeof(char), max, ficheiro);   //**< Modo leitura relativamente a
    dados binários do tipo string */
242             fread(&plano->calorias_min[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a
    dados binários interios */
243             fread(&plano->calorias_max[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a
    dados binários interios */
244         }
248         for (int i = 0; i < 3; i++)
249         {
250             printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal \n", plano->n_cliente[i], plano->dia[i],
    plano->mes[i], plano->ano[i], plano->refeicao[i], plano->calorias_min[i], plano->calorias_max[i]);
251         }
252
253         fclose(ficheiro); //**< Fecha o arquivo. */
254     }
255     else //**< Outros casos  */
256     {
257         printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro  */
258         return;
259     }
260 }
```

5.2.1.5 listar_plano_csv()

```
void listar_plano_csv (
            Plano * plano,
            int diainicio,
            int diafim,
            int mesinicio,
            int mesfim )
```

Está função é responsável por listar um cliente existente no plano, a escolha do utilizador para ficheiros CSV.

Realiza a leitura do número do cliente e da refeição que o utilizador pretende visualizar, verifica o periodo em seguida compra cliente e refeição com os dados do plano, caso sejam verdadeiros apresenta os dados do cliente caso sejam falsa aprenta uma mensagem de erro.

Parameters

plano	Invocação da struct plano na função.
diainicio	Invocação do dia de início defenido.
mesinicio	Invocação do mes de início defenido.
diafim	Invocação do dia de fim defenido.
mesfim	Invocação do mes de fim defenido.

```
276 {
277     int cliente;       //**< Armazena nº cliente*/
278     char refeicao[20]; //**< Armazena o nome do cliente*/
279 
280     printf("\nDigite o cliente e a refeição que pretende visualizar (numero refeição):");
281     scanf("%d%[^\n]", &cliente, refeicao); //**< Realiza a leitura */
282 
283     printf("\n  Plano nutricional do cliente %04d, refeição:%s\n\n", cliente, refeicao); //**< Apresenta
      os clientes selecionados */
284 
285     for (int i = 0; i < max; i++)
286     {
287         if ((mesinicio < plano->mes[i] && plano->mes[i] < mesfim) || //**< Verifica o período se for
      válido entra no loop */
288             (mesinicio == plano->mes[i] && diainicio <= plano->dia[i] && plano->dia[i] <= diafim) ||
289             (mesfim == plano->mes[i] && diainicio <= plano->dia[i] && plano->dia[i] <= diafim))
290         {
291             if (cliente == plano->n_cliente[i]) //**< Verifica se o cliente existe */
292             {
293                 if (strcmp(refeicao, plano->refeicao[i]) == 0) //**< Verifica se a refeição existe */
294                 {
295                     printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal\n", plano->n_cliente[i],
      plano->dia[i], plano->mes[i], plano->ano[i], plano->refeicao[i], plano->calorias_min[i],
      plano->calorias_max[i]); //**< Apresenta o plano do cliente selecionado */
296                 }
297             }
298         }
299     }
300 }
```

5.2.1.6 tabela_csv()

```
void tabela_csv (
             Dados * dados,
             Dieta * dieta,
             Plano * plano )
```

Tabela com todas as informações do plano nutricional para ficheiros CSV.

Inicialmente, a mesma verifica o numero de cliente e a refeição, dentro dessas condições soma as calorias e por cliente e refeição. Por fim verifica o numero de cliente e a refeição, e apresenta as calorias acumuladas por cliente.

Parameters

dados	Invocação da struct dados na função.
dieta	Invocação da struct dieta na função.
plano	Invocação da struct plano na função.

```
463 {
464     float cal_pequeno1 = 0, cal_almoco1 = 0, cal_jantar1 = 0; //**< Armazena as calorias cliente 1*/
465     float cal_pequeno2 = 0, cal_almoco2 = 0, cal_jantar2 = 0; //**< Armazena as calorias cliente 2*/
```

```
466     float cal_pequeno3 = 0, cal_almoco3 = 0, cal_jantar3 = 0; //**< Armazena as calorias cliente 3*/
467
468     for (int i = 0; i < 6; i++)
469     {
470
471         if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */
472         {
473             if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição */
474             {
475                 cal_pequeno1 = cal_pequeno1 + dieta->calorias[i]; //**< Soma as calorias */
476             }
477             if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
478             {
479                 cal_almoco1 = cal_almoco1 + dieta->calorias[i]; //**< Soma as calorias */
480             }
481             if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */
482             {
483                 cal_jantar1 = cal_jantar1 + dieta->calorias[i]; //**< Soma as calorias */
484             }
485         }
486
487         if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */
488         {
489             if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição */
490             {
491                 cal_pequeno2 = cal_pequeno2 + dieta->calorias[i]; //**< Soma as calorias */
492             }
493             if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
494             {
495                 cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Soma as calorias */
496             }
497             if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */
498             {
499                 cal_jantar2 = cal_jantar2 + dieta->calorias[i]; //**< Soma as calorias */
500             }
501         }
502
503         if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */
504         {
505             if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição */
506             {
507                 cal_pequeno3 = cal_pequeno3 + dieta->calorias[i]; //**< Soma as calorias */
508             }
509             if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
510             {
511                 cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Soma as calorias */
512             }
513             if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */
514             {
515                 cal_jantar3 = cal_jantar3 + dieta->calorias[i]; //**< Soma as calorias */
516             }
517         }
518     }
519
520     printf("\nTabela:\n\n");
521
    printf("\t-------------------------------------------------------------------------------------------------\n");
522     printf("\t|   NP   | Paciente |   Tipo Refeição  |   Inicio   |    Fim     | Mínimo | Máximo  |
    Consumo |\n");
523
    printf("\t|--------|----------|------------------|------------|------------|--------|---------|---------|\n");
524
525     for (int i = 0; i < 3; i++)
526     {
527         for (int j = 0; j < 3; j++)
528         {
529             if (dados->n_cliente[i] == plano->n_cliente[j]) //**< Responsável por verificar os clientes
    que contém planos, para realizar a aprsentação do nome correto  */
530             {
531
532                 if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */
533                 {
534                     if (strcmp(plano->refeicao[j], " jantar") == 0) //**< Verifica a refeição */
535                     {
536                         printf("\t|  %04d  |   %s   | %-16s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f   |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_jantar1); //**< Apresenta linha da tabela */
537                     }
538                     else if (strcmp(plano->refeicao[j], " almoço") == 0) //**< Verifica a refeição */
539                     {
540                         printf("\t|  %04d  |   %s   | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_almoco1); //**< Apresenta linha da tabela */
541                     }
542                     else if (strcmp(plano->refeicao[j], " pequeno almoço") == 0) //**< Verifica a
```

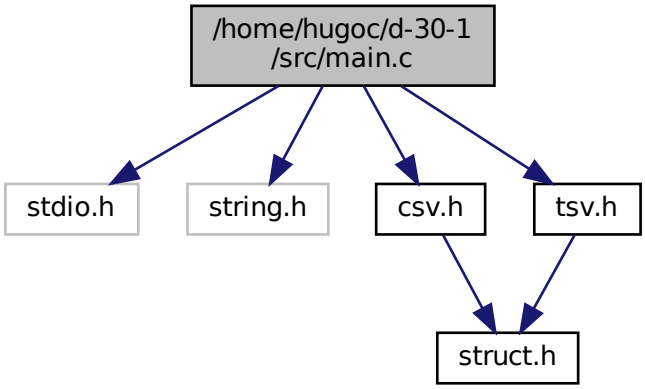
```
refeição */
543                     {
544                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_pequeno1); //**< Apresenta linha da tabela */
545                     }
546                     else //**< Outros casos */
547                     {
548                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   | -----\n ", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j]); //**<
    Apresenta linha da tabela */
549                     }
550                 }
551                 if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */
552                 {
553                     if (strcmp(plano->refeicao[j], " jantar") == 0) //**< Verifica a refeição */
554                     {
555
556                         printf("\t|  %04d  |   %s  | %-16s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f   |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_jantar2); //**< Apresenta linha da tabela */
557                     }
558                     else if (strcmp(plano->refeicao[j], " almoço") == 0) //**< Verifica a refeição */
559                     {
560                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_almoco2); //**< Apresenta linha da tabela */
561                     }
562                     else if (strcmp(plano->refeicao[j], " pequeno almoço") == 0) //**< Verifica a
    refeição */
563                     {
564                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_pequeno2); //**< Apresenta linha da tabela */
565                     }
566                     else //**< Outros casos */
567                     {
568                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   | -----\n ", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j]); //**<
    Apresenta linha da tabela */
569                     }
570                 }
571
572                 if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */
573                 {
574                     if (strcmp(plano->refeicao[j], " jantar") == 0) //**< Verifica a refeição */
575                     {
576
577                         printf("\t|  %04d  |   %s  | %-16s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f   |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_jantar3); //**< Apresenta linha da tabela */
578                     }
579                     else if (strcmp(plano->refeicao[j], " almoço") == 0) //**< Verifica a refeição */
580                     {
581                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_almoco3); //**< Apresenta linha da tabela */
582                     }
583                     else if (strcmp(plano->refeicao[j], " pequeno almoço") == 0) //**< Verifica a
    refeição */
584                     {
585                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_pequeno3); //**< Apresenta linha da tabela */
586                     }
587                     else //**< Outros casos */
588                     {
589                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   | -----\n ", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j]); //**<
    Apresenta linha da tabela */
590                     }
591                 }
592             }
593         }
594     }
595
    printf("\t-------------------------------------------------------------------------------------------\n");
```

```
596 }
```

5.3 /home/hugoc/d-30-1/src/main.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "csv.h"
#include "tsv.h"
```

Include dependency graph for main.c:



Functions

- void periodo (int *diainicio, int *mesinicio, int *diafim, int *mesfim)

  Define um período especificado pelos usuários.
- void contador_calorias (Dieta *dieta, int diainicio, int diafim, int mesinicio, int mesfim)

  Realiza a contagem de quem ultrapassou certas calorias num período de tempo.
- void listagem (Dieta *dieta, Plano *plano, Dados *dados, int diainicio, int diafim, int mesinicio, int mesfim)

  Está função é responsável por ordenar os clientes por ordem decrescente e apresentar os clientes fora do intervalo de calorias.
- int main (int argc, char *argv[ ])

  Função principal do programa.

5.3.1 Detailed Description

Author

Grupo 30

Version

0.73

Date

2023-12-29

Copyright

5.3.2 Function Documentation

5.3.2.1 contador_calorias()

```
void contador_calorias (
            Dieta * dieta,
            int diainicio,
            int diafim,
            int mesinicio,
            int mesfim )
```

Realiza a contagem de quem ultrapassou certas calorias num período de tempo.

Esta função solicita ao usuário que insira um valor de calorias e um período de tempo (no formato dd-mm), e verifica se as datas estão dentro de intervalos válidos (dia: 1-31, mês: 1-12), e se as calorias do ficheiro dietas forem maiores que as inseridas, realiza a contagem de quantos utilizadores ultrapassaram as calorias.

Parameters

dieta	Invocação da struct dieta na função.
diainicio	Invocação do dia de início defenido.
mesinicio	Invocação do mes de início defenido.
diafim	Invocação do dia de fim defenido.
mesfim	Invocação do mes de fim defenido.

```
58 {
59     float calorias;   //**< Armazena as calorias inseridas pelo utilizador  */
60     int contador = 0; //**< Realiza a contagem dos clientes que ultrapassaram as calorias inseridas */
61 
62     printf("\nDigite o valor de calorias que pretende ver quantos utilizadores ultrapassaram:");
63     scanf("%f", &calorias);
64 
65     for (int i = 0; i < 4; i++)
66     {
67         if ((mesinicio < dieta->mes[i] && dieta->mes[i] < mesfim) || //**< Verifica o período se for
       válido entra no loop */
68             (mesinicio == dieta->mes[i] && diainicio <= dieta->dia[i] && dieta->dia[i] <= diafim) ||
69             (mesfim == dieta->mes[i] && diainicio <= dieta->dia[i] && dieta->dia[i] <= diafim))
70         {
71             if (dieta->calorias[i] > calorias) //**< Verifica se as calorias são maiores que as inseridas
       pelo utilizador */
72             {
73                 contador++; //**< Contador de utilizadores que ultrapassaram as calorias
74             }
75         }
76     }
77 
78     printf("\n\tDe 4 utilizadores %d ultrapassaram as %0.01f calorias. \n", contador, calorias);
79 }
```

5.3.2.2 listagem()

```
void listagem (
              Dieta * dieta,
              Plano * plano,
              Dados * dados,
              int diainicio,
              int diafim,
              int mesinicio,
              int mesfim )
```

Está função é responsável por ordenar os clientes por ordem decrescente e apresentar os clientes fora do intervalo de calorias.

Inicialmente, a mesma verifica se as datas estão dentro de intervalos válidos, restringindo assim alguns dos clientes, caso as datas estejam dfora dos intervalos válidos, compara os números de cliente da struct dieta com os números de cliente da struct plano, garantindo que os clientes que não estão no plano não são apresentados. Por fim, compara as refeições, garantindo que os clientes realizem as refeições planeadas no plano, e valida se estão fora do intervalo de calorias.

Parameters

dieta	Invocação da struct dieta na função.
plano	Invocação da struct plano na função.
diainicio	Invocação do dia de início defenido.
mesinicio	Invocação do mes de início defenido.
diafim	Invocação do dia de fim defenido.
mesfim	Invocação do mes de fim defenido.

```
97 {
98     int j, k, temp;                    //**< Variáveis auxiliares */
99     int clientes_verificados[4] = {0}; //**< Array para armazenar os clientes verificados */
100     char nomes_ordenados[max][max];    //**< Matriz para armazenar os nomes dos clientes verificados */
101
102     for (j = 0; j < 4; j++)
103     {
104         if ((mesinicio < dieta->mes[j] && dieta->mes[j] < mesfim) || //**< Verifica o período se for
    válido entra no loop */
105             (mesinicio == dieta->mes[j] && diainicio <= dieta->dia[j] && dieta->dia[j] <= diafim) ||
106             (mesfim == dieta->mes[j] && diainicio <= dieta->dia[j] && dieta->dia[j] <= diafim))
107         {
108             for (k = 0; k < 4; k++)
109             {
110                 if (dieta->n_cliente[j] == plano->n_cliente[k]) //**< Verifica se os clientes existem em
    planos */
111                 {
112                     if (strcmp(dieta->refeicao[j], plano->refeicao[k]) == 0) //**< Verifica se as
    refeições existem em planos */
113                     {
114                         if (dieta->calorias[j] < plano->calorias_min[k] || dieta->calorias[j] >
    plano->calorias_max[k]) //**< Verifica se as calorias estão fora do intervalo defenido */
115                         {
116                             if (dados->n_cliente[j] != 0)
117                             {
118                                 clientes_verificados[j] = dados->n_cliente[j];      //**< Armazena os
    clientes verificados num variavel da função*/
119                                 strcpy(nomes_ordenados[j], dados->nome_cliente[j]); //**< Armazena os
    nomes dos clientes verificados num variavel da função*/
120                             }
121                         }
122                     }
123                 }
124             }
125         }
126     }
127
128     for (j = 0; j < 4; j++)
129     {
130         for (k = j + 1; k < 4; k++)
131         {
```

```
132             if (clientes_verificados[j] < clientes_verificados[k]) //**< Ordena os clientes por ordem
      decrescente */
133             {
134                 temp = clientes_verificados[j];
135                 clientes_verificados[j] = clientes_verificados[k];
136                 clientes_verificados[k] = temp;
137 
138                 char temp_nome[50];
139                 strcpy(temp_nome, nomes_ordenados[j]);
140                 strcpy(nomes_ordenados[j], nomes_ordenados[k]);
141                 strcpy(nomes_ordenados[k], temp_nome);
142             }
143         }
144     }
145 
146     // Imprimir os clientes ordenados
147     printf("\nListagem dos pacientes ordenada por ordem decrescente:\n\n");
148     for (j = 0; j < 4; j++)
149     {
150         if (clientes_verificados[j] != 0)
151         {
152             printf("\t%04d %s\n", clientes_verificados[j], nomes_ordenados[j]); //**< Apresenta os
      clientes ordenados */
153         }
154     }
155 }
```

5.3.2.3 main()

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

Função principal do programa.

Parameters

argc	Número de argumentos de linha de comando.
argv	Array de strings contendo os argumentos de linha de comando.

```
164 {
165     int modo;
166     int diainicio = 0, diafim = 0; //**< Armazena dias */
167     int mesinicio = 0, mesfim = 0; //**< Armazena meses */
168 
169     if (argc < 2) //**< Verifica se o numero de argumentos */
170     {
171         printf("Caso necessite de ajuda: %s -ajuda\n", argv[0]); //**< Apresenta a mensagem como usar
      modo -ajuda */
172         return 1;
173     }
174 
175     if (strcmp(argv[1], "-csv") == 0) //**< Verifica se o argumento é -csv */
176         modo = 1;
177     else if (strcmp(argv[1], "-tsv") == 0) //**< Verifica se o argumento é -tsv */
178         modo = 2;
179     else if (strcmp(argv[1], "-ajuda") == 0) //**< Verifica se o argumento é -ajuda */
180         modo = 3;
181     else //**< Outros casos */
182     {
183         printf("Caso necessite de ajuda: %s -ajuda\n", argv[0]);
184         return 1;
185     }
186 
187     Dados dados; //**< Invocação da struct dados */
188     Dieta dieta; //**< Invocação da struct dieta */
189     Plano plano; //**< Invocação da struct plano */
190 
191     switch (modo)
192     {
193     case 1: //**< Caso o argumento seja -csv */
194 
195         ficheiro_csv_1(argv[2], &dados);                                    //**< Invoca a função
```

```
      ficheiro_csv_1*/
196         ficheiro_csv_2(argv[3], &dieta);                                    //**< Invoca a função
      ficheiro_csv_2*/
197         ficheiro_csv_3(argv[4], &plano);                                    //**< Invoca a função
      ficheiro_csv_3*/
198         periodo(&diainicio, &diafim, &mesinicio, &mesfim);                  //**< Invoca a função
      periodo*/
199         contador_calorias(&dieta, diainicio, mesinicio, diafim, mesfim);    //**< Invoca a função */
200         listagem(&dieta, &plano, &dados, diainicio, mesinicio, diafim, mesfim); //**< Invoca a função
      contador_calorias*/
201         listar_plano_csv(&plano, diainicio, mesinicio, diafim, mesfim);     //**< Invoca a função
      listar_plano_csv */
202         calcmedia_csv(&dados, &dieta, diainicio, diafim, mesinicio, mesfim); //**< Invoca a função
      calcmedia_csv*/
203         tabela_csv(&dados, &dieta, &plano);                                 //**< Invoca a função
      tabela_csv*/
204
205         break;
206
207     case 2: //**< Caso o argumento seja -tsv */
208
209         ficheiro_tsv_1(argv[2], &dados);                                    //**< Invoca a função
      ficheiro_tsv_1*/
210         ficheiro_tsv_2(argv[3], &dieta);                                    //**< Invoca a função
      ficheiro_tsv_2 */
211         ficheiro_tsv_3(argv[4], &plano);                                    //**< Invoca a função
      ficheiro_tsv_3 */
212         periodo(&diainicio, &diafim, &mesinicio, &mesfim);                  //**< Invoca a função
      periodo*/
213         contador_calorias(&dieta, diainicio, mesinicio, diafim, mesfim);    //**< Invoca a função
      contador_calorias*/
214         listagem(&dieta, &plano, &dados, diainicio, mesinicio, diafim, mesfim); //**< Invoca a função */
215         listar_plano_tsv(&plano, diainicio, mesinicio, diafim, mesfim);     //**< Invoca a função
      listagem*/
216         calcmedia_tsv(&dados, &dieta, diainicio, diafim, mesinicio, mesfim); //**< Invoca a função
      calcmedia_tsv*/
217         tabela_tsv(&dados, &dieta, &plano);                                 //**< Invoca a função
      tabela_tsv*/
218
219         break;
220
221     case 3: //**< Caso o argumento seja -ajuda */
222
223         printf("\nEste programa utiliza os seguintes formatos (CSV ou TSV). Existindo tambem uma versão
      binária de cada ficheiro \n");
224         printf("\nModo de utilização:\n");
225         printf("\tPara utilizar com arquivos CSV, execute: %s -csv dados.csv dietas.csv planos.csv\n",
      argv[0]);
226         printf("\tPara utilizar com arquivos TSV, execute: %s -tsv dados.tsv dietas.tsv planos.tsv\n",
      argv[0]);
227         printf("\tPara utilizar com arquivos (CSV|bin), execute: %s -csv dados.csv dietas_csv.bin
      planos.csv\n", argv[0]);
228         printf("\tPara utilizar com arquivos (TSV|bin), execute: %s -tsv dados.tsv dietas_tsv.bin
      planos_tsv.bin\n", argv[0]);
229         break;
230
231         return 0;
232     }
233 }
```

5.3.2.4 periodo()

```
void periodo (
              int * diainicio,
              int * mesinicio,
              int * diafim,
              int * mesfim )
```

Define um período especificado pelos usuários.

Esta função solicita ao usuário que insira datas de início e fim (no formato dd-mm) e verifica se as datas estão dentro de intervalos válidos (dia: 1-31, mês: 1-12). A função continua a solicitar entradas até que datas válidas sejam fornecidas.

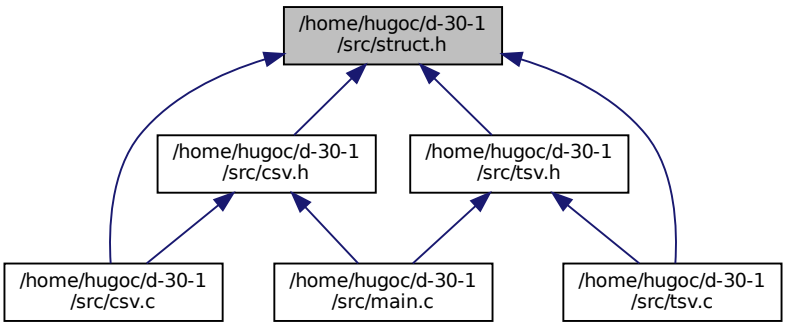
Parameters

diainicio	Armazena o dia de início defenido.
mesinicio	Armazena o mes de início defenido.
diafim	Armazena o dia de fim defenido.
mesfim	Armazena o mes de fim defenido.

```
30 {
31     printf("\nDefinir periodo (dd-mm): \n");
32     while (*diainicio > 31 || *diainicio < 1 || *mesinicio > 12 || *mesinicio < 1) //**< Condições
       necessárias  */
33     {
34         printf("\n\tData de inicio: ");
35         scanf("%d-%d", diainicio, mesinicio); //**< Leitura das datas de inicio  */
36     }
37
38     while (*diafim > 31 || *diafim < 1 || *mesfim > 12 || *mesfim < 1) //**< Condições necessárias  */
39     {
40         printf("\tData de fim: ");
41         scanf("%d-%d", diafim, mesfim); //**< Leitura das datas de fim  */
42     }
43 }
```

5.4 /home/hugoc/d-30-1/src/struct.h File Reference

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



Data Structures

- struct Dados

  Esta estrutura contém arrays para armazenar o número do cliente, o nome do cliente e o número de telefone associado a cada cliente.

- struct Dieta

  Esta estrutura contém arrays para armazenar o número do cliente, tipos de refeição, tipos de alimento, calorias, e a data de cada refeição.

- struct Plano

  Esta estrutura contém arrays para armazenar o número do cliente, data tipos de refeição, calorias minimas e maximas.

Macros

- #define max 100

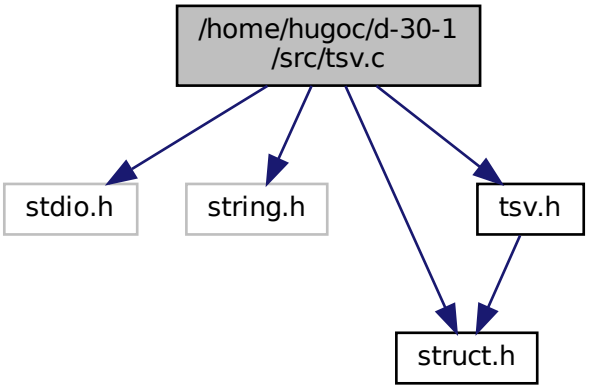
  Define o valor máximo como 100.
- #define min 10

  Define o valor mínimo como 10.

5.5 /home/hugoc/d-30-1/src/tsv.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "struct.h"
#include "tsv.h"
```

Include dependency graph for tsv.c:



Functions

- void ficheiro_tsv_1 (char *arquivo, Dados *dados)

  Lê dados de um arquivo TSV e bin e armazena os na struct Dados.
- void ficheiro_tsv_2 (char *arquivo, Dieta *dieta)

  Lê dados de um arquivo TSV (tab-separated values) e .bin e armazena os na estrutura Dieta.
- void ficheiro_tsv_3 (char *arquivo, Plano *plano)

  Lê dados de um arquivo TSV (tab-separated values) e .bin e armazena os na estrutura Plano.
- void listar_plano_tsv (Plano *plano, int diainicio, int diafim, int mesinicio, int mesfim)

  Está função é responsável por listar um cliente existente no plano, a escolha do utilizador para ficheiros TSV.
- void calcmedia_tsv (Dados *dados, Dieta *dieta, int diainicio, int mesinicio, int diafim, int mesfim)

  Calcula a média de calorias consumidas por cliente num período de tempo para ficheiros TSV.
- void tabela_tsv (Dados *dados, Dieta *dieta, Plano *plano)

  Tabela com todas as informações do plano nutricional para ficheiros TSV.

5.5.1 Function Documentation

5.5.1.1 calcmedia_tsv()

```
void calcmedia_tsv (
            Dados * dados,
            Dieta * dieta,
            int diainicio,
            int mesinicio,
            int diafim,
            int mesfim )
```

Calcula a média de calorias consumidas por cliente num período de tempo para ficheiros TSV.

Inicialmente, a mesma verifica se as datas estão dentro de intervalos válidos, restringindo assim alguns dos clientes, em seguida verifica o numero de cliente e a refeição, onde realiza a soma das calorias e a contagem de vezes que o cliente realizou a refeição. Por fim caso as calorias sejam diferentes de 0, realiza a média das calorias consumidas por cliente.

Parameters

plano	Invocação da struct plano na função.
diainicio	Invocação do dia de início defenido.
mesinicio	Invocação do mes de início defenido.
diafim	Invocação do dia de fim defenido.
mesfim	Invocação do mes de fim defenido.

```
309 {
310     int i = 0;                                                  //**< Responsável por percorrer o array*/
311     float c1_pequeno = 0, c1_almoco = 0, c1_jantar = 0;         //**< Armazena as contagens */
312     float cal_pequeno1 = 0, cal_almoco1 = 0, cal_jantar1 = 0; //**< Armazena as calorias */
313
314     float c2_pequeno = 0, c2_almoco = 0, c2_jantar = 0;         //**< Armazena as contagens */
315     float cal_pequeno2 = 0, cal_almoco2 = 0, cal_jantar2 = 0; //**< Armazena as calorias */
316
317     float c3_pequeno = 0, c3_almoco = 0, c3_jantar = 0;         //**< Armazena as contagens */
318     float cal_pequeno3 = 0, cal_almoco3 = 0, cal_jantar3 = 0; //**< Armazena as calorias */
319
320     printf("\nMédia de todos os clientes:\n\n");
321
322     for (i = 0; i < 6; i++)
323     {
324         if ((mesinicio < dieta->mes[i] && dieta->mes[i] < mesfim) || //**< Válida o período  */
325             (mesinicio == dieta->mes[i] && diainicio <= dieta->dia[i] && dieta->dia[i] <= diafim) ||
326             (mesfim == dieta->mes[i] && diainicio <= dieta->dia[i] && dieta->dia[i] <= diafim))
327         {
328             if (dieta->n_cliente[i] == 1) //**< Verifica o numero de cliente */
329
330             {
331                 if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição  */
332                 {
333                     cal_pequeno1 = cal_pequeno1 + dieta->calorias[i]; //**< Acumula as calorias  */
334                     c1_pequeno++;                                     //**< Contador  */
335                 }
336                 if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição  */
337                 {
338                     cal_almoco1 = cal_almoco1 + dieta->calorias[i]; //**< Acumula as calorias  */
339                     c1_almoco++;                                    //**< Contador  */
340                 }
341                 if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição  */
342                 {
343                     cal_jantar1 = cal_jantar1 + dieta->calorias[i]; //**< Acumula as calorias  */
344                     c1_jantar++;                                    //**< Contador  */
345                 }
346             }
```

```
347
348             if (dieta->n_cliente[i] == 2) //**< Verifica o numero de cliente */
349             {
350                 if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição  */
351                 {
352                     cal_pequeno2 = cal_pequeno2 + dieta->calorias[i]; //**< Acumula as calorias  */
353                     c2_pequeno++;                                     //**< Contador  */
354                 }
355                 if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição  */
356                 {
357                     cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Acumula as calorias  */
358                     c2_almoco++;                                    //**< Contador  */
359                 }
360                 if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição  */
361                 {
362                     cal_jantar2 = cal_jantar2 + dieta->calorias[i]; //**< Acumula as calorias  */
363                     c2_jantar++;                                    //**< Contador  */
364                 }
365             }
366
367             if (dieta->n_cliente[i] == 3) //**< Verifica o numero de cliente */
368             {
369                 if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição  */
370                 {
371                     cal_pequeno3 = cal_pequeno3 + dieta->calorias[i]; //**< Acumula as calorias  */
372                     c3_pequeno++;                                     //**< Contador  */
373                 }
374                 if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição  */
375                 {
376                     cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Acumula as calorias  */
377                     c3_almoco++;                                    //**< Contador  */
378                 }
379                 if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição  */
380                 {
381                     cal_jantar3 = cal_jantar3 + dieta->calorias[i]; //**< Acumula as calorias  */
382                     c3_jantar++;                                    //**< Contador  */
383                 }
384             }
385         }
386     }
387
388     // Cliente 1
389     if (cal_pequeno1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
390     {
391         float media_cal_pequeno1 = cal_pequeno1 / c1_pequeno;
              //**< Realiza a media   */
392         printf("\tMédia das calorias consumidas pelo cliente 1 ao pequeno almoço: %.2f\n",
    media_cal_pequeno1); //**< Apresenta a media  */
393     }
394
395     if (cal_almoco1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
396     {
397         float media_cal_almoco1 = cal_almoco1 / c1_almoco;
    //**< Realiza a media   */
398         printf("\tMédia das calorias consumidas pelo cliente 1 ao almoço: %.2f\n", media_cal_almoco1);
    //**< Apresenta a media  */
399     }
400
401     if (cal_jantar1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
402     {
403         float media_cal_jantar1 = cal_jantar1 / c1_jantar;
    //**< Realiza a media   */
404         printf("\tMédia das calorias consumidas pelo cliente 1 ao jantar: %.2f\n", media_cal_jantar1);
    //**< Apresenta a media  */
405     }
406
407     // Cliente 2
408     if (cal_pequeno2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
409     {
410         float media_cal_pequeno2 = cal_pequeno2 / c2_pequeno;
              //**< Realiza a media   */
411         printf("\tMédia das calorias consumidas pelo cliente 2 ao pequeno almoço: %.2f\n",
    media_cal_pequeno2); //**< Apresenta a media  */
412     }
413
414     if (cal_almoco2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
415     {
416         float media_cal_almoco2 = cal_almoco2 / c2_almoco;
    //**< Realiza a media   */
417         printf("\tMédia das calorias consumidas pelo cliente 2 ao almoço: %.2f\n", media_cal_almoco2);
    //**< Apresenta a media  */
418     }
419
420     if (cal_jantar2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
421     {
422         float media_cal_jantar2 = cal_jantar2 / c2_jantar;
    //**< Realiza a media   */
```

```
423         printf("\tMédia das calorias consumidas pelo cliente 2 ao jantar: %.2f\n", media_cal_jantar2);
      //**< Apresenta a media  */
424     }
425
426     // Cliente 3
427     if (cal_pequeno3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
428     {
429         float media_cal_pequeno3 = cal_pequeno3 / c3_pequeno;
              //**< Realiza a media   */
430         printf("\tMédia das calorias consumidas pelo cliente 3 ao pequeno almoço: %.2f\n",
      media_cal_pequeno3); //**< Apresenta a media  */
431     }
432
433     if (cal_almoco3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
434     {
435         float media_cal_almoco3 = cal_almoco3 / c3_almoco;
      //**< Realiza a media   */
436         printf("\tMédia das calorias consumidas pelo cliente 3 ao almoço: %.2f\n", media_cal_almoco3);
      //**< Apresenta a media  */
437     }
438
439     if (cal_jantar3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
440     {
441         float media_cal_jantar3 = cal_jantar3 / c3_jantar;
      //**< Realiza a media   */
442         printf("\tMédia das calorias consumidas pelo cliente 3 ao jantar: %.2f\n", media_cal_jantar3);
      //**< Apresenta a media  */
443     }
444 }
```

5.5.1.2 ficheiro_tsv_1()

```
void ficheiro_tsv_1 (
            char * arquivo,
            Dados * dados )
```

Lê dados de um arquivo TSV e bin e armazena os na struct Dados.

Parameters

arquivo	Nome do arquivo TSV a ser lido na função.
dados	Invocação da struct dados na função.

Criação de um ciclo while para ler os dados do arquivo TSV e armazená-los na estrutura Dados.

Parameters

ficheiro	Nome do arquivo TSV a ser lido na função.
dados	Invocação da struct dados na função.
i	Variável para percorrer o array.

For: vai realizar a leitura dos dados do arquivo binário e armazená-los na struct Dados.

For: Apresenta os dados armazenados na struct Dados.

```
19 {
20     if (strcmp(arquivo, "dados.tsv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
       */
21     {
22         FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
       ficheiro. */
23         int i = 0;
24
25         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
```

```
26          {
27              printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
28              return;                                               //**< Retorno indicando falha na
        abertura do arquivo.*/
29          }
30
31          printf("\nDados clientes:\n\n");
32
41          while (fscanf(ficheiro, "%d\t%s\t%d\n", &dados->n_cliente[i], dados->nome_cliente[i],
        &dados->num_telefone[i]) == 3)
42          {
43              printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
        dados->num_telefone[i]);
44              i++;
45          }
46
47          fclose(ficheiro); //**< Fecha o arquivo. */
48      }
49      else if (strcmp(arquivo, "dados_tsv.bin") == 0) //**< Compara as strings, caso sejam iguais entra
        neste loop */
50      {
51          FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
        nome ficheiro. */
52
53          int i = 0;
54
55          if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
56          {
57              printf("Erro: Não foi possível abrir o arquivo\n"); //**< Apresenta uma mensagem de erro*/
58              return;                                             //**< Retorno indicando falha na abertura
        do arquivo.*/
59          }
60          printf("\nDados clientes:\n\n");
61
67          for (int i = 0; i < 3; i++)
68          {
69              fread(&dados->n_cliente[i], sizeof(int), 1, ficheiro);      //**< Modo leitura relativamente
        a dados binários interios */
70              fread(dados->nome_cliente[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente
        a dados binários do tipo string */
71              fread(&dados->num_telefone[i], sizeof(int), 1, ficheiro);   //**< Modo leitura relativamente
        a dados binários interios */
72          }
76          for (int i = 0; i < 3; i++)
77          {
78              printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
        dados->num_telefone[i]);
79          }
80
81          fclose(ficheiro);
82      }
83      else //**< Outros casos  */
84      {
85          printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro  */
86          return;
87      }
88  }
```

5.5.1.3 ficheiro_tsv_2()

```
void ficheiro_tsv_2 (
            char * arquivo,
            Dieta * dieta )
```

Lê dados de um arquivo TSV (tab-separated values) e .bin e armazena os na estrutura Dieta.

Parameters

arquivo	Nome do arquivo (TSV|bin) a ser lido.
dieta	Ponteiro para a estrutura Dieta onde os dados serão armazenados.

Extrai dados da linha formatada do arquivo TSV e os armazena na estrutura Dieta.

Parameters

linha	Nome da linha do arquivo CSV a ser lido na função.
dieta	Invocação da struct dieta na função.
i	Variável para percorrer o array.

For: vai realizar a leitura dos dados do arquivo binário e armazená-los na struct Dados.

For: Apresenta os dados armazenados na struct Dados.

```
97 {
98      if (strcmp(arquivo, "dietas.tsv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
        */
99      {
100         FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
        ficheiro. */
101         char linha[max];                      //**< Tamanho para armazenar cada linha do arquivo. */
102
103         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
104         {
105             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
106             return;                                               //**< Retorno indicando falha na
        abertura do arquivo.*/
107         }
108
109         printf("\nDieta realizada pelo cliente:\n\n");
110
111         for (int i = 0; i < max; i++)
112         {
113             if (fgets(linha, sizeof(linha), ficheiro) == NULL) //**< Verifica se as linhas do arquivo
        CSV estão vazias. */
114             {
115                 break; //**< Sai do loop se não houver mais linhas para ler. */
116             }
117
126             sscanf(linha, "%d\t%d-%d-%d\t%99[^\t]\t%[^\t]\t%d ", &dieta->n_cliente[i], &dieta->dia[i],
        &dieta->mes[i], &dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], &dieta->calorias[i]);
127             printf("\t%04d %02d-%02d-%d %s %s %d \n", dieta->n_cliente[i], dieta->dia[i], dieta->mes[i],
        dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], dieta->calorias[i]);
128         }
129
130         fclose(ficheiro); //**< Fecha o arquivo após a leitura dos dados. */
131     }
132     else if (strcmp(arquivo, "dietas_tsv.bin") == 0) //**< Compara as strings, caso sejam iguais entra
        neste loop */
133     {
134         FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
        nome ficheiro. */
135
136         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
137         {
138             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
139             return;                                               //**< Retorno indicando falha na
        abertura do arquivo.*/
140         }
141
142         printf("\nDieta realizada pelo cliente:\n\n");
143
148         for (int i = 0; i < 4; i++)
149         {
150             fread(&dieta->n_cliente[i], sizeof(int), 1, ficheiro);  //**< Modo leitura relativamente a
        dados binários interios */
151             fread(&dieta->dia[i], sizeof(int), 1, ficheiro);        //**< Modo leitura relativamente a
        dados binários interios */
152             fread(&dieta->mes[i], sizeof(int), 1, ficheiro);        //**< Modo leitura relativamente a
        dados binários interios */
153             fread(&dieta->ano[i], sizeof(int), 1, ficheiro);        //**< Modo leitura relativamente a
        dados binários interios */
154             fread(dieta->refeicao[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a
        dados binários do tipo string */
155             fread(dieta->alimento[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a
        dados binários do tipo string */
156             fread(&dieta->calorias[i], sizeof(int), 1, ficheiro);   //**< Modo leitura relativamente a
        dados binários interios */
157         }
161         for (int i = 0; i < 4; i++)
162         {
163             printf("\t%04d %02d-%02d-%d %s %s %d \n", dieta->n_cliente[i], dieta->dia[i], dieta->mes[i],
        dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], dieta->calorias[i]);
164         }
165
166         fclose(ficheiro);
```

```
167     }
168     else //**< Outros casos  */
169     {
170         printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro  */
171         return;
172     }
173 }
```

5.5.1.4 ficheiro_tsv_3()

```
void ficheiro_tsv_3 (
            char * arquivo,
            Plano * plano )
```

Lê dados de um arquivo TSV (tab-separated values) e .bin e armazena os na estrutura Plano.

Parameters

arquivo	Nome do arquivo (TSV|bin) a ser lido.
dieta	Ponteiro para a estrutura Plano onde os dados serão armazenados.

Extrai dados da linha formatada do arquivo TSV e os armazena na estrutura Dieta.

Parameters

linha	Nome da linha do arquivo CSV a ser lido na função.
dieta	Invocação da struct dieta na função.
i	Variável para percorrer o array.

For: vai realizar a leitura dos dados do arquivo binário e armazená-los na struct Dados.

For: Apresenta os dados armazenados na struct Dados.

```
181 {
182     if (strcmp(arquivo, "planos.tsv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
      */
183     {
184         FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
      ficheiro. */
185         char linha[100];                      //**< Tamanho para armazenar cada linha do arquivo. */
186 
187         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
188         {
189             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
190             return;                                               //**< Retorno indicando falha na
      abertura do arquivo.*/
191         }
192 
193         printf("\nPlano nutricional:\n\n");
194 
195         for (int i = 0; i < 100; i++)
196         {
197             if (fgets(linha, sizeof(linha), ficheiro) == NULL) //**< Verifica se as linhas do arquivo
      CSV estão vazias. */
198             {
199                 break; //**< Sai do loop se não houver mais linhas para ler. */
200             }
201 
210             sscanf(linha, "%d \t %d-%d-%d \t %99[^\t] \t %d Cal, %d Cal", &plano->n_cliente[i],
      &plano->dia[i], &plano->mes[i], &plano->ano[i], plano->refeicao[i], &plano->calorias_min[i],
      &plano->calorias_max[i]);
211             printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal\n", plano->n_cliente[i], plano->dia[i],
      plano->mes[i], plano->ano[i], plano->refeicao[i], plano->calorias_min[i], plano->calorias_max[i]);
212         }
```

```
213         fclose(ficheiro); //**< Fecha o arquivo após a leitura dos dados. */
214     }
215     else if (strcmp(arquivo, "planos_tsv.bin") == 0) //**< Verifica se as linhas do arquivo CSV estão
    vazias. */
216     {
217         FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
    nome ficheiro. */
218 
219         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
220         {
221             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
222             return;                                               //**< Retorno indicando falha na
    abertura do arquivo.*/
223         }
224 
225         printf("\nPlano nutricional:\n\n");
229         for (int i = 0; i < 3; i++)
230         {
231             fread(&plano->n_cliente[i], sizeof(int), 1, ficheiro);    //**< Modo leitura relativamente a
    dados binários interios */
232             fread(&plano->dia[i], sizeof(int), 1, ficheiro);          //**< Modo leitura relativamente a
    dados binários interios */
233             fread(&plano->mes[i], sizeof(int), 1, ficheiro);          //**< Modo leitura relativamente a
    dados binários interios */
234             fread(&plano->ano[i], sizeof(int), 1, ficheiro);          //**< Modo leitura relativamente a
    dados binários interios */
235             fread(plano->refeicao[i], sizeof(char), max, ficheiro);   //**< Modo leitura relativamente a
    dados binários do tipo string */
236             fread(&plano->calorias_min[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a
    dados binários interios */
237             fread(&plano->calorias_max[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a
    dados binários interios */
238         }
242         for (int i = 0; i < 3; i++)
243         {
244             printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal \n", plano->n_cliente[i], plano->dia[i],
    plano->mes[i], plano->ano[i], plano->refeicao[i], plano->calorias_min[i], plano->calorias_max[i]);
245         }
246 
247         fclose(ficheiro); //**< Fecha o arquivo. */
248     }
249     else //**< Outros casos  */
250     {
251         printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro  */
252         return;
253     }
254 }
```

5.5.1.5 listar_plano_tsv()

```
void listar_plano_tsv (
              Plano * plano,
              int diainicio,
              int diafim,
              int mesinicio,
              int mesfim )
```

Está função é responsável por listar um cliente existente no plano, a escolha do utilizador para ficheiros TSV.

Realiza a leitura do número do cliente e da refeição que o utilizador pretende visualizar, verifica o periodo em seguida compra cliente e refeição com os dados do plano, caso sejam verdadeiros apresenta os dados do cliente caso sejam falsa aprenta uma mensagem de erro.

Parameters

plano	Invocação da struct plano na função.
diainicio	Invocação do dia de início defenido.
mesinicio	Invocação do mes de início defenido.
diafim	Invocação do dia de fim defenido.
mesfim	Invocação do mes de fim defenido.

```
270 {
271     int cliente;       //**< Armazena nº cliente*/
272     char refeicao[20]; //**< Armazena o nome do cliente*/
273 
274     printf("\nDigite o cliente e a refeição que pretende visualizar (numero refeição):");
275     scanf("%d\t %[^\n]", &cliente, refeicao); //**< Realiza a leitura */
276 
277     printf("\n\tPlano nutricional do cliente %04d, refeição:%s\n\n", cliente, refeicao);
278 
279     for (int i = 0; i < 100; i++)
280     {
281         if ((mesinicio < plano->mes[i] && plano->mes[i] < mesfim) || //**< Verifica o período se for
    válido entra no loop */
282             (mesinicio == plano->mes[i] && diainicio <= plano->dia[i] && plano->dia[i] <= diafim) ||
283             (mesfim == plano->mes[i] && diainicio <= plano->dia[i] && plano->dia[i] <= diafim))
284         {
285             if (cliente == plano->n_cliente[i]) //**< Verifica se o cliente existe */
286             {
287                 if (strcmp(refeicao, plano->refeicao[i]) == 0) //**< Verifica se a refeição existe */
288                 {
289                     printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal\n", plano->n_cliente[i],
    plano->dia[i], plano->mes[i], plano->ano[i], plano->refeicao[i], plano->calorias_min[i],
    plano->calorias_max[i]); //**< Apresenta o plano do cliente selecionado */
290                 }
291             }
292         }
293     }
294 }
```

5.5.1.6 tabela_tsv()

```
void tabela_tsv (
            Dados * dados,
            Dieta * dieta,
            Plano * plano )
```

Tabela com todas as informações do plano nutricional para ficheiros TSV.

Inicialmente, a mesma verifica o numero de cliente e a refeição, dentro dessas condições soma as calorias e por cliente e refeição. Por fim verifica o numero de cliente e a refeição, e apresenta as calorias acumuladas por cliente.

Parameters

dados	Invocação da struct dados na função.
dieta	Invocação da struct dieta na função.
plano	Invocação da struct plano na função.

```
456 {
457     float cal_pequeno1 = 0, cal_almoco1 = 0, cal_jantar1 = 0; //**< Armazena as calorias cliente 1*/
458     float cal_pequeno2 = 0, cal_almoco2 = 0, cal_jantar2 = 0; //**< Armazena as calorias cliente 2*/
459     float cal_pequeno3 = 0, cal_almoco3 = 0, cal_jantar3 = 0; //**< Armazena as calorias cliente 3*/
460 
461     for (int i = 0; i < 6; i++)
462     {
463 
464         if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */
465         {
466             if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */
467             {
468                 cal_pequeno1 = cal_pequeno1 + dieta->calorias[i]; //**< Soma as calorias */
469             }
470             if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */
471             {
472                 cal_almoco1 = cal_almoco1 + dieta->calorias[i]; //**< Soma as calorias */
473             }
474             if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */
475             {
476                 cal_jantar1 = cal_jantar1 + dieta->calorias[i]; //**< Soma as calorias */
477             }
478         }
```

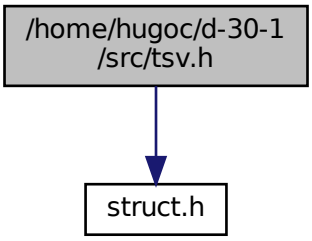
```
479
480         if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */
481         {
482             if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */
483             {
484                 cal_pequeno2 = cal_pequeno2 + dieta->calorias[i]; //**< Soma as calorias */
485             }
486             if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */
487             {
488                 cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Soma as calorias */
489             }
490             if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */
491             {
492                 cal_jantar2 = cal_jantar2 + dieta->calorias[i];
493             }
494         }
495
496         if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */
497         {
498             if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */
499             {
500                 cal_pequeno3 = cal_pequeno3 + dieta->calorias[i]; //**< Soma as calorias */
501             }
502             if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */
503             {
504                 cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Soma as calorias */
505             }
506             if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */
507             {
508                 cal_jantar3 = cal_jantar3 + dieta->calorias[i]; //**< Soma as calorias */
509             }
510         }
511     }
512
513     printf("\nTabela:\n\n");
514
    printf("\t-------------------------------------------------------------------------------------------------\n");
515     printf("\t|   NP   | Paciente |   Tipo Refeição  |   Inicio   |    Fim     | Mínimo | Máximo  |
    Consumo |\n");
516
    printf("\t|--------|----------|------------------|------------|------------|--------|---------|---------|\n");
517
518     for (int i = 0; i < 3; i++)
519     {
520         for (int j = 0; j < 3; j++)
521         {
522             if (dados->n_cliente[i] == plano->n_cliente[j]) //**< Responsável por verificar os clientes
    que contém planos, para realizar a aprsentação do nome correto  */
523             {
524
525                 if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */
526                 {
527                     if (strcmp(plano->refeicao[j], "jantar") == 0) //**< Verifica a refeição */
528                     {
529
530                         printf("\t|  %04d  |   %s  | %-16s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f   |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_jantar1); //**< Apresenta linha da tabela */
531                     }
532                     else if (strcmp(plano->refeicao[j], "almoço") == 0) //**< Verifica a refeição */
533                     {
534                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_almoco1); //**< Apresenta linha da tabela */
535                     }
536                     else if (strcmp(plano->refeicao[j], "pequeno almoço") == 0) //**< Verifica a
    refeição */
537                     {
538                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_pequeno1); //**< Apresenta linha da tabela */
539                     }
540                     else //**< Outros casos */
541                     {
542                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   | -----\n ", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j]); //**<
    Apresenta linha da tabela */
543                     }
544                 }
545                 if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */
546                 {
547                     if (strcmp(plano->refeicao[j], "jantar") == 0) //**< Verifica a refeição */
548                     {
```

```
549
550                         printf("\t|  %04d  |   %s  | %-16s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f   |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_jantar2); //**< Apresenta linha da tabela */
551                     }
552                     else if (strcmp(plano->refeicao[j], "almoço") == 0) //**< Verifica a refeição */
553                     {
554                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_almoco2); //**< Apresenta linha da tabela */
555                     }
556                     else if (strcmp(plano->refeicao[j], "pequeno almoço") == 0) //**< Verifica a
    refeição */
557                     {
558                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_pequeno2); //**< Apresenta linha da tabela */
559                     }
560                     else //**< Outros casos */
561                     {
562                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   | -----\n ", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j]); //**<
    Apresenta linha da tabela */
563                     }
564                 }
565
566                 if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */
567                 {
568                     if (strcmp(plano->refeicao[j], "jantar") == 0) //**< Verifica a refeição */
569                     {
570
571                         printf("\t|  %04d  |   %s  | %-16s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f   |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_jantar3); //**< Apresenta linha da tabela */
572                     }
573                     else if (strcmp(plano->refeicao[j], "almoço") == 0) //**< Verifica a refeição */
574                     {
575                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_almoco3); //**< Apresenta linha da tabela */
576                     }
577                     else if (strcmp(plano->refeicao[j], "pequeno almoço") == 0) //**< Verifica a
    refeição */
578                     {
579                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_pequeno3); //**< Apresenta linha da tabela */
580                     }
581                     else //**< Outros casos */
582                     {
583                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   | -----\n ", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j]); //**<
    Apresenta linha da tabela */
584                     }
585                 }
586             }
587         }
588     }
589
    printf("\t-------------------------------------------------------------------------------------------------\n");
590 }
```

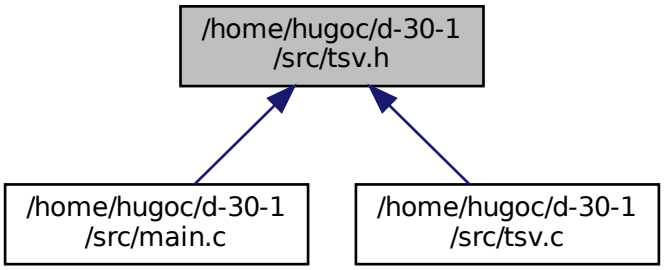
5.6 /home/hugoc/d-30-1/src/tsv.h File Reference

```
#include "struct.h"
```

Include dependency graph for tsv.h:



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



Functions

- void ficheiro_tsv_1 (char ∗arquivo, Dados ∗dados)

  Lê dados de um arquivo TSV e bin e armazena os na struct Dados.
- void ficheiro_tsv_2 (char ∗arquivo, Dieta ∗dieta)

  Lê dados de um arquivo TSV (tab-separated values) e .bin e armazena os na estrutura Dieta.
- void ficheiro_tsv_3 (char ∗arquivo, Plano ∗plano)

  Lê dados de um arquivo TSV (tab-separated values) e .bin e armazena os na estrutura Plano.
- void listar_plano_tsv (Plano ∗plano, int diainicio, int diafim, int mesinicio, int mesfim)

  Está função é responsável por listar um cliente existente no plano, a escolha do utilizador para ficheiros TSV.
- void calcmedia_tsv (Dados ∗dados, Dieta ∗dieta, int diainicio, int mesinicio, int diafim, int mesfim)

  Calcula a média de calorias consumidas por cliente num período de tempo para ficheiros TSV.
- void tabela_tsv (Dados ∗dados, Dieta ∗dieta, Plano ∗plano)

  Tabela com todas as informações do plano nutricional para ficheiros TSV.

5.6.1 Function Documentation

5.6.1.1 calcmedia_tsv()

```
void calcmedia_tsv (
            Dados * dados,
            Dieta * dieta,
            int diainicio,
            int mesinicio,
            int diafim,
            int mesfim )
```

Calcula a média de calorias consumidas por cliente num período de tempo para ficheiros TSV.

Inicialmente, a mesma verifica se as datas estão dentro de intervalos válidos, restringindo assim alguns dos clientes, em seguida verifica o numero de cliente e a refeição, onde realiza a soma das calorias e a contagem de vezes que o cliente realizou a refeição. Por fim caso as calorias sejam diferentes de 0, realiza a média das calorias consumidas por cliente.

Parameters

plano	Invocação da struct plano na função.
diainicio	Invocação do dia de início defenido.
mesinicio	Invocação do mes de início defenido.
diafim	Invocação do dia de fim defenido.
mesfim	Invocação do mes de fim defenido.

```
309 {
310     int i = 0;                                                  //**< Responsável por percorrer o array*/
311     float c1_pequeno = 0, c1_almoco = 0, c1_jantar = 0;         //**< Armazena as contagens */
312     float cal_pequeno1 = 0, cal_almoco1 = 0, cal_jantar1 = 0; //**< Armazena as calorias */
313
314     float c2_pequeno = 0, c2_almoco = 0, c2_jantar = 0;         //**< Armazena as contagens */
315     float cal_pequeno2 = 0, cal_almoco2 = 0, cal_jantar2 = 0; //**< Armazena as calorias */
316
317     float c3_pequeno = 0, c3_almoco = 0, c3_jantar = 0;         //**< Armazena as contagens */
318     float cal_pequeno3 = 0, cal_almoco3 = 0, cal_jantar3 = 0; //**< Armazena as calorias */
319
320     printf("\nMédia de todos os clientes:\n\n");
321
322     for (i = 0; i < 6; i++)
323     {
324         if ((mesinicio < dieta->mes[i] && dieta->mes[i] < mesfim) || //**< Válida o período  */
325             (mesinicio == dieta->mes[i] && diainicio <= dieta->dia[i] && dieta->dia[i] <= diafim) ||
326             (mesfim == dieta->mes[i] && diainicio <= dieta->dia[i] && dieta->dia[i] <= diafim))
327         {
328             if (dieta->n_cliente[i] == 1) //**< Verifica o numero de cliente */
329
330             {
331                 if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição  */
332                 {
333                     cal_pequeno1 = cal_pequeno1 + dieta->calorias[i]; //**< Acumula as calorias  */
334                     c1_pequeno++;                                     //**< Contador  */
335                 }
336                 if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição  */
337                 {
338                     cal_almoco1 = cal_almoco1 + dieta->calorias[i]; //**< Acumula as calorias  */
339                     c1_almoco++;                                    //**< Contador  */
340                 }
341                 if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição  */
342                 {
343                     cal_jantar1 = cal_jantar1 + dieta->calorias[i]; //**< Acumula as calorias  */
344                     c1_jantar++;                                    //**< Contador  */
345                 }
346             }
347
348             if (dieta->n_cliente[i] == 2) //**< Verifica o numero de cliente */
349             {
350                 if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição  */
351                 {
352                     cal_pequeno2 = cal_pequeno2 + dieta->calorias[i]; //**< Acumula as calorias  */
353                     c2_pequeno++;                                     //**< Contador  */
354                 }
355                 if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição  */
```

```
356                 {
357                     cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Acumula as calorias  */
358                     c2_almoco++;                                    //**< Contador  */
359                 }
360                 if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição  */
361                 {
362                     cal_jantar2 = cal_jantar2 + dieta->calorias[i]; //**< Acumula as calorias  */
363                     c2_jantar++;                                    //**< Contador  */
364                 }
365             }
366 
367             if (dieta->n_cliente[i] == 3) //**< Verifica o numero de cliente */
368             {
369                 if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição  */
370                 {
371                     cal_pequeno3 = cal_pequeno3 + dieta->calorias[i]; //**< Acumula as calorias  */
372                     c3_pequeno++;                                      //**< Contador  */
373                 }
374                 if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição  */
375                 {
376                     cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Acumula as calorias  */
377                     c3_almoco++;                                    //**< Contador  */
378                 }
379                 if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição  */
380                 {
381                     cal_jantar3 = cal_jantar3 + dieta->calorias[i]; //**< Acumula as calorias  */
382                     c3_jantar++;                                    //**< Contador  */
383                 }
384             }
385         }
386     }
387 
388     // Cliente 1
389     if (cal_pequeno1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
390     {
391         float media_cal_pequeno1 = cal_pequeno1 / c1_pequeno;
              //**< Realiza a media   */
392         printf("\tMédia das calorias consumidas pelo cliente 1 ao pequeno almoço: %.2f\n",
    media_cal_pequeno1); //**< Apresenta a media  */
393     }
394 
395     if (cal_almoco1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
396     {
397         float media_cal_almoco1 = cal_almoco1 / c1_almoco;
    //**< Realiza a media   */
398         printf("\tMédia das calorias consumidas pelo cliente 1 ao almoço: %.2f\n", media_cal_almoco1);
    //**< Apresenta a media  */
399     }
400 
401     if (cal_jantar1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
402     {
403         float media_cal_jantar1 = cal_jantar1 / c1_jantar;
    //**< Realiza a media   */
404         printf("\tMédia das calorias consumidas pelo cliente 1 ao jantar: %.2f\n", media_cal_jantar1);
    //**< Apresenta a media  */
405     }
406 
407     // Cliente 2
408     if (cal_pequeno2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
409     {
410         float media_cal_pequeno2 = cal_pequeno2 / c2_pequeno;
              //**< Realiza a media   */
411         printf("\tMédia das calorias consumidas pelo cliente 2 ao pequeno almoço: %.2f\n",
    media_cal_pequeno2); //**< Apresenta a media  */
412     }
413 
414     if (cal_almoco2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
415     {
416         float media_cal_almoco2 = cal_almoco2 / c2_almoco;
    //**< Realiza a media   */
417         printf("\tMédia das calorias consumidas pelo cliente 2 ao almoço: %.2f\n", media_cal_almoco2);
    //**< Apresenta a media  */
418     }
419 
420     if (cal_jantar2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
421     {
422         float media_cal_jantar2 = cal_jantar2 / c2_jantar;
    //**< Realiza a media   */
423         printf("\tMédia das calorias consumidas pelo cliente 2 ao jantar: %.2f\n", media_cal_jantar2);
    //**< Apresenta a media  */
424     }
425 
426     // Cliente 3
427     if (cal_pequeno3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
428     {
429         float media_cal_pequeno3 = cal_pequeno3 / c3_pequeno;
              //**< Realiza a media   */
```

```
430         printf("\tMédia das calorias consumidas pelo cliente 3 ao pequeno almoço: %.2f\n",
      media_cal_pequeno3); //**< Apresenta a media  */
431     }
432
433     if (cal_almoco3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
434     {
435         float media_cal_almoco3 = cal_almoco3 / c3_almoco;
      //**< Realiza a media   */
436         printf("\tMédia das calorias consumidas pelo cliente 3 ao almoço: %.2f\n", media_cal_almoco3);
      //**< Apresenta a media  */
437     }
438
439     if (cal_jantar3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop  */
440     {
441         float media_cal_jantar3 = cal_jantar3 / c3_jantar;
      //**< Realiza a media   */
442         printf("\tMédia das calorias consumidas pelo cliente 3 ao jantar: %.2f\n", media_cal_jantar3);
      //**< Apresenta a media  */
443     }
444 }
```

5.6.1.2 ficheiro_tsv_1()

```
void ficheiro_tsv_1 (
            char * arquivo,
            Dados * dados )
```

Lê dados de um arquivo TSV e bin e armazena os na struct Dados.

Parameters

arquivo	Nome do arquivo TSV a ser lido na função.
dados	Invocação da struct dados na função.

Criação de um ciclo while para ler os dados do arquivo TSV e armazená-los na estrutura Dados.

Parameters

ficheiro	Nome do arquivo TSV a ser lido na função.
dados	Invocação da struct dados na função.
i	Variável para percorrer o array.

For: vai realizar a leitura dos dados do arquivo binário e armazená-los na struct Dados.

For: Apresenta os dados armazenados na struct Dados.

```
19 {
20     if (strcmp(arquivo, "dados.tsv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
      */
21     {
22         FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
      ficheiro. */
23         int i = 0;
24
25         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
26         {
27             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
28             return;                                              //**< Retorno indicando falha na
      abertura do arquivo.*/
29         }
30
31         printf("\nDados clientes:\n\n");
32
41         while (fscanf(ficheiro, "%d\t%s\t%d\n", &dados->n_cliente[i], dados->nome_cliente[i],
      &dados->num_telefone[i]) == 3)
```

```
42          {
43              printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
    dados->num_telefone[i]);
44              i++;
45          }
46
47          fclose(ficheiro); //**< Fecha o arquivo. */
48      }
49      else if (strcmp(arquivo, "dados_tsv.bin") == 0) //**< Compara as strings, caso sejam iguais entra
    neste loop */
50      {
51          FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
    nome ficheiro. */
52
53          int i = 0;
54
55          if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
56          {
57              printf("Erro: Não foi possível abrir o arquivo\n"); //**< Apresenta uma mensagem de erro*/
58              return;                                             //**< Retorno indicando falha na abertura
    do arquivo.*/
59          }
60          printf("\nDados clientes:\n\n");
61
67          for (int i = 0; i < 3; i++)
68          {
69              fread(&dados->n_cliente[i], sizeof(int), 1, ficheiro);      //**< Modo leitura relativamente
    a dados binários interios */
70              fread(dados->nome_cliente[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente
    a dados binários do tipo string */
71              fread(&dados->num_telefone[i], sizeof(int), 1, ficheiro);   //**< Modo leitura relativamente
    a dados binários interios */
72          }
76          for (int i = 0; i < 3; i++)
77          {
78              printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
    dados->num_telefone[i]);
79          }
80
81          fclose(ficheiro);
82      }
83      else //**< Outros casos  */
84      {
85          printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro  */
86          return;
87      }
88  }
```

5.6.1.3 ficheiro_tsv_2()

```
void ficheiro_tsv_2 (
              char * arquivo,
              Dieta * dieta )
```

Lê dados de um arquivo TSV (tab-separated values) e .bin e armazena os na estrutura Dieta.

Parameters

arquivo	Nome do arquivo (TSV|bin) a ser lido.
dieta	Ponteiro para a estrutura Dieta onde os dados serão armazenados.

Extrai dados da linha formatada do arquivo TSV e os armazena na estrutura Dieta.

Parameters

linha	Nome da linha do arquivo CSV a ser lido na função.
dieta	Invocação da struct dieta na função.
i	Variável para percorrer o array.

For: vai realizar a leitura dos dados do arquivo binário e armazená-los na struct Dados.

For: Apresenta os dados armazenados na struct Dados.

```
97 {
98     if (strcmp(arquivo, "dietas.tsv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
       */
99     {
100         FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
       ficheiro. */
101         char linha[max];                      //**< Tamanho para armazenar cada linha do arquivo. */
102 
103         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
104         {
105             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
106             return;                                               //**< Retorno indicando falha na
       abertura do arquivo.*/
107         }
108 
109         printf("\nDieta realizada pelo cliente:\n\n");
110 
111         for (int i = 0; i < max; i++)
112         {
113             if (fgets(linha, sizeof(linha), ficheiro) == NULL) //**< Verifica se as linhas do arquivo
       CSV estão vazias. */
114             {
115                 break; //**< Sai do loop se não houver mais linhas para ler. */
116             }
117 
126             sscanf(linha, "%d\t%d-%d-%d\t%99[^\t]\t%[^\t]\t%d ", &dieta->n_cliente[i], &dieta->dia[i],
       &dieta->mes[i], &dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], &dieta->calorias[i]);
127             printf("\t%04d %02d-%02d-%d %s %s %d \n", dieta->n_cliente[i], dieta->dia[i], dieta->mes[i],
       dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], dieta->calorias[i]);
128         }
129 
130         fclose(ficheiro); //**< Fecha o arquivo após a leitura dos dados. */
131     }
132     else if (strcmp(arquivo, "dietas_tsv.bin") == 0) //**< Compara as strings, caso sejam iguais entra
       neste loop */
133     {
134         FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
       nome ficheiro. */
135 
136         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
137         {
138             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
139             return;                                               //**< Retorno indicando falha na
       abertura do arquivo.*/
140         }
141 
142         printf("\nDieta realizada pelo cliente:\n\n");
143 
148         for (int i = 0; i < 4; i++)
149         {
150             fread(&dieta->n_cliente[i], sizeof(int), 1, ficheiro);  //**< Modo leitura relativamente a
       dados binários interios */
151             fread(&dieta->dia[i], sizeof(int), 1, ficheiro);        //**< Modo leitura relativamente a
       dados binários interios */
152             fread(&dieta->mes[i], sizeof(int), 1, ficheiro);        //**< Modo leitura relativamente a
       dados binários interios */
153             fread(&dieta->ano[i], sizeof(int), 1, ficheiro);        //**< Modo leitura relativamente a
       dados binários interios */
154             fread(dieta->refeicao[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a
       dados binários do tipo string */
155             fread(dieta->alimento[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a
       dados binários do tipo string */
156             fread(&dieta->calorias[i], sizeof(int), 1, ficheiro);   //**< Modo leitura relativamente a
       dados binários interios */
157         }
161         for (int i = 0; i < 4; i++)
162         {
163             printf("\t%04d %02d-%02d-%d %s %s %d \n", dieta->n_cliente[i], dieta->dia[i], dieta->mes[i],
       dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], dieta->calorias[i]);
164         }
165 
166         fclose(ficheiro);
167     }
168     else //**< Outros casos  */
169     {
170         printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro  */
171         return;
172     }
173 }
```

5.6.1.4 ficheiro_tsv_3()

```
void ficheiro_tsv_3 (
            char * arquivo,
            Plano * plano )
```

Lê dados de um arquivo TSV (tab-separated values) e .bin e armazena os na estrutura Plano.

Parameters

arquivo	Nome do arquivo (TSV|bin) a ser lido.
dieta	Ponteiro para a estrutura Plano onde os dados serão armazenados.

Extrai dados da linha formatada do arquivo TSV e os armazena na estrutura Dieta.

Parameters

linha	Nome da linha do arquivo CSV a ser lido na função.
dieta	Invocação da struct dieta na função.
i	Variável para percorrer o array.

For: vai realizar a leitura dos dados do arquivo binário e armazená-los na struct Dados.

For: Apresenta os dados armazenados na struct Dados.

```
181 {
182     if (strcmp(arquivo, "planos.tsv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
    */
183     {
184         FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
    ficheiro. */
185         char linha[100];                      //**< Tamanho para armazenar cada linha do arquivo. */
186 
187         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
188         {
189             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
190             return;                                               //**< Retorno indicando falha na
    abertura do arquivo.*/
191         }
192 
193         printf("\nPlano nutricional:\n\n");
194 
195         for (int i = 0; i < 100; i++)
196         {
197             if (fgets(linha, sizeof(linha), ficheiro) == NULL) //**< Verifica se as linhas do arquivo
    CSV estão vazias. */
198             {
199                 break; //**< Sai do loop se não houver mais linhas para ler. */
200             }
201 
210             sscanf(linha, "%d \t %d-%d-%d \t %99[^\t] \t %d Cal, %d Cal", &plano->n_cliente[i],
    &plano->dia[i], &plano->mes[i], &plano->ano[i], plano->refeicao[i], &plano->calorias_min[i],
    &plano->calorias_max[i]);
211             printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal\n", plano->n_cliente[i], plano->dia[i],
    plano->mes[i], plano->ano[i], plano->refeicao[i], plano->calorias_min[i], plano->calorias_max[i]);
212         }
213         fclose(ficheiro); //**< Fecha o arquivo após a leitura dos dados. */
214     }
215     else if (strcmp(arquivo, "planos_tsv.bin") == 0) //**< Verifica se as linhas do arquivo CSV estão
    vazias. */
216     {
217         FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
    nome ficheiro. */
218 
219         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
220         {
221             printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
222             return;                                               //**< Retorno indicando falha na
    abertura do arquivo.*/
223         }
```

```
224
225         printf("\nPlano nutricional:\n\n");
229         for (int i = 0; i < 3; i++)
230         {
231             fread(&plano->n_cliente[i], sizeof(int), 1, ficheiro);    //**< Modo leitura relativamente a
    dados binários interios */
232             fread(&plano->dia[i], sizeof(int), 1, ficheiro);          //**< Modo leitura relativamente a
    dados binários interios */
233             fread(&plano->mes[i], sizeof(int), 1, ficheiro);          //**< Modo leitura relativamente a
    dados binários interios */
234             fread(&plano->ano[i], sizeof(int), 1, ficheiro);          //**< Modo leitura relativamente a
    dados binários interios */
235             fread(plano->refeicao[i], sizeof(char), max, ficheiro);   //**< Modo leitura relativamente a
    dados binários do tipo string */
236             fread(&plano->calorias_min[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a
    dados binários interios */
237             fread(&plano->calorias_max[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a
    dados binários interios */
238         }
242         for (int i = 0; i < 3; i++)
243         {
244             printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal \n", plano->n_cliente[i], plano->dia[i],
    plano->mes[i], plano->ano[i], plano->refeicao[i], plano->calorias_min[i], plano->calorias_max[i]);
245         }
246
247         fclose(ficheiro); //**< Fecha o arquivo. */
248     }
249     else //**< Outros casos  */
250     {
251         printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro  */
252         return;
253     }
254 }
```

5.6.1.5 listar_plano_tsv()

```
void listar_plano_tsv (
              Plano * plano,
              int diainicio,
              int diafim,
              int mesinicio,
              int mesfim )
```

Está função é responsável por listar um cliente existente no plano, a escolha do utilizador para ficheiros TSV.

Realiza a leitura do número do cliente e da refeição que o utilizador pretende visualizar, verifica o periodo em seguida compra cliente e refeição com os dados do plano, caso sejam verdadeiros apresenta os dados do cliente caso sejam falsa aprenta uma mensagem de erro.

Parameters

plano	Invocação da struct plano na função.
diainicio	Invocação do dia de início defenido.
mesinicio	Invocação do mes de início defenido.
diafim	Invocação do dia de fim defenido.
mesfim	Invocação do mes de fim defenido.

```
270 {
271     int cliente;       //**< Armazena nº cliente*/
272     char refeicao[20]; //**< Armazena o nome do cliente*/
273
274     printf("\nDigite o cliente e a refeição que pretende visualizar (numero refeição):");
275     scanf("%d\t %[^\n]", &cliente, refeicao); //**< Realiza a leitura */
276
277     printf("\n\tPlano nutricional do cliente %04d, refeição:%s\n\n", cliente, refeicao);
278
279     for (int i = 0; i < 100; i++)
```

```
280     {
281         if ((mesinicio < plano->mes[i] && plano->mes[i] < mesfim) || //**< Verifica o período se for
    válido entra no loop */
282             (mesinicio == plano->mes[i] && diainicio <= plano->dia[i] && plano->dia[i] <= diafim) ||
283             (mesfim == plano->mes[i] && diainicio <= plano->dia[i] && plano->dia[i] <= diafim))
284         {
285             if (cliente == plano->n_cliente[i]) //**< Verifica se o cliente existe */
286             {
287                 if (strcmp(refeicao, plano->refeicao[i]) == 0) //**< Verifica se a refeição existe */
288                 {
289                     printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal\n", plano->n_cliente[i],
    plano->dia[i], plano->mes[i], plano->ano[i], plano->refeicao[i], plano->calorias_min[i],
    plano->calorias_max[i]); //**< Apresenta o plano do cliente selecionado */
290                 }
291             }
292         }
293     }
294 }
```

5.6.1.6 tabela_tsv()

```
void tabela_tsv (
            Dados * dados,
            Dieta * dieta,
            Plano * plano )
```

Tabela com todas as informações do plano nutricional para ficheiros TSV.

Inicialmente, a mesma verifica o numero de cliente e a refeição, dentro dessas condições soma as calorias e por cliente e refeição. Por fim verifica o numero de cliente e a refeição, e apresenta as calorias acumuladas por cliente.

Parameters

dados	Invocação da struct dados na função.
dieta	Invocação da struct dieta na função.
plano	Invocação da struct plano na função.

```
456 {
457     float cal_pequeno1 = 0, cal_almoco1 = 0, cal_jantar1 = 0; //**< Armazena as calorias cliente 1*/
458     float cal_pequeno2 = 0, cal_almoco2 = 0, cal_jantar2 = 0; //**< Armazena as calorias cliente 2*/
459     float cal_pequeno3 = 0, cal_almoco3 = 0, cal_jantar3 = 0; //**< Armazena as calorias cliente 3*/
460
461     for (int i = 0; i < 6; i++)
462     {
463
464         if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */
465         {
466             if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */
467             {
468                 cal_pequeno1 = cal_pequeno1 + dieta->calorias[i]; //**< Soma as calorias */
469             }
470             if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */
471             {
472                 cal_almoco1 = cal_almoco1 + dieta->calorias[i]; //**< Soma as calorias */
473             }
474             if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */
475             {
476                 cal_jantar1 = cal_jantar1 + dieta->calorias[i]; //**< Soma as calorias */
477             }
478         }
479
480         if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */
481         {
482             if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */
483             {
484                 cal_pequeno2 = cal_pequeno2 + dieta->calorias[i]; //**< Soma as calorias */
485             }
486             if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */
487             {
488                 cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Soma as calorias */
```

```
489             }
490             if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */
491             {
492                 cal_jantar2 = cal_jantar2 + dieta->calorias[i];
493             }
494         }
495 
496         if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */
497         {
498             if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */
499             {
500                 cal_pequeno3 = cal_pequeno3 + dieta->calorias[i]; //**< Soma as calorias */
501             }
502             if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */
503             {
504                 cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Soma as calorias */
505             }
506             if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */
507             {
508                 cal_jantar3 = cal_jantar3 + dieta->calorias[i]; //**< Soma as calorias */
509             }
510         }
511     }
512 
513     printf("\nTabela:\n\n");
514 
    printf("\t-------------------------------------------------------------------------------------------------\n");
515     printf("\t|   NP   | Paciente |   Tipo Refeição  |   Inicio   |    Fim     | Mínimo | Máximo  |
    Consumo |\n");
516 
    printf("\t|--------|----------|------------------|------------|------------|--------|---------|---------|\n");
517 
518     for (int i = 0; i < 3; i++)
519     {
520         for (int j = 0; j < 3; j++)
521         {
522             if (dados->n_cliente[i] == plano->n_cliente[j]) //**< Responsável por verificar os clientes
    que contém planos, para realizar a aprsentação do nome correto  */
523             {
524 
525                 if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */
526                 {
527                     if (strcmp(plano->refeicao[j], "jantar") == 0) //**< Verifica a refeição */
528                     {
529 
530                         printf("\t|  %04d  |   %s  | %-16s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f   |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_jantar1); //**< Apresenta linha da tabela */
531                     }
532                     else if (strcmp(plano->refeicao[j], "almoço") == 0) //**< Verifica a refeição */
533                     {
534                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_almoco1); //**< Apresenta linha da tabela */
535                     }
536                     else if (strcmp(plano->refeicao[j], "pequeno almoço") == 0) //**< Verifica a
    refeição */
537                     {
538                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_pequeno1); //**< Apresenta linha da tabela */
539                     }
540                     else //**< Outros casos */
541                     {
542                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   | -----\n ", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j]); //**<
    Apresenta linha da tabela */
543                     }
544                 }
545                 if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */
546                 {
547                     if (strcmp(plano->refeicao[j], "jantar") == 0) //**< Verifica a refeição */
548                     {
549 
550                         printf("\t|  %04d  |   %s  | %-16s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f   |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
    cal_jantar2); //**< Apresenta linha da tabela */
551                     }
552                     else if (strcmp(plano->refeicao[j], "almoço") == 0) //**< Verifica a refeição */
553                     {
554                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
    %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
```

```
        dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
        cal_almoco2); //**< Apresenta linha da tabela */
555                     }
556                     else if (strcmp(plano->refeicao[j], "pequeno almoço") == 0) //**< Verifica a
        refeição */
557                     {
558                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
        %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
        dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
        cal_pequeno2); //**< Apresenta linha da tabela */
559                     }
560                     else //**< Outros casos */
561                     {
562                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
        %d   | -----\n ", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
        dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j]); //**<
        Apresenta linha da tabela */
563                     }
564                 }
565
566                 if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */
567                 {
568                     if (strcmp(plano->refeicao[j], "jantar") == 0) //**< Verifica a refeição */
569                     {
570
571                         printf("\t|  %04d  |   %s  | %-16s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
        %d   |  %0.2f   |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
        dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
        cal_jantar3); //**< Apresenta linha da tabela */
572                     }
573                     else if (strcmp(plano->refeicao[j], "almoço") == 0) //**< Verifica a refeição */
574                     {
575                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
        %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
        dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
        cal_almoco3); //**< Apresenta linha da tabela */
576                     }
577                     else if (strcmp(plano->refeicao[j], "pequeno almoço") == 0) //**< Verifica a
        refeição */
578                     {
579                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
        %d   |  %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
        dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
        cal_pequeno3); //**< Apresenta linha da tabela */
580                     }
581                     else //**< Outros casos */
582                     {
583                         printf("\t|  %04d  |   %s  | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |  %d   |
        %d   | -----\n ", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
        dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j]); //**<
        Apresenta linha da tabela */
584                     }
585                 }
586             }
587         }
588     }
589
        printf("\t-------------------------------------------------------------------------------------------\n");
590 }
```

Index