Grupo 30

Generated by Doxygen 1.8.17

1 Trabalho Prático	1
1.1 Grupo 30	1
1.2 Organização	1
1.3 Relatórios e Código	1
1.4 Descrição do Projeto	1
1.5 Estratégia de Desenvolvimento	2
1.6 Decisões Coletivas	2
1.7 Compilação e Execução:	2
1.8 Comando com Sintaxe de Utilização:	2
1.9 Conclusão	3
2 Data Structure Index	5
2.1 Data Structures	5
3 File Index	7
3.1 File List	7
4 Data Structure Documentation	9
4.1 Dados Struct Reference	9
4.1.1 Detailed Description	9
4.1.2 Field Documentation	9
4.1.2.1 n_cliente	9
4.1.2.2 nome_cliente	9
4.1.2.3 num_telefone	10
4.2 Dieta Struct Reference	10
4.2.1 Detailed Description	10
4.2.2 Field Documentation	10
4.2.2.1 alimento	10
4.2.2.2 ano	10
4.2.2.3 calorias	11
4.2.2.4 dia	11
4.2.2.5 mes	11
4.2.2.6 n_cliente	11
4.2.2.7 refeicao	11
4.3 Plano Struct Reference	11
4.3.1 Detailed Description	12
4.3.2 Field Documentation	12
4.3.2.1 ano	12
4.3.2.2 calorias_max	12
4.3.2.3 calorias_min	12
4.3.2.4 dia	12
4.3.2.5 mes	13
4.3.2.6 n_cliente	13

4.3.2.7 refeicao	. 13
5 File Documentation	15
5.1 /home/hugoc/d-30-1/src/csv.c File Reference	. 15
5.1.1 Function Documentation	. 16
5.1.1.1 calcmedia_csv()	. 16
5.1.1.2 ficheiro_csv_1()	. 18
5.1.1.3 ficheiro_csv_2()	. 19
5.1.1.4 ficheiro_csv_3()	. 21
5.1.1.5 listar_plano_csv()	. 22
5.1.1.6 tabela_csv()	. 23
5.2 /home/hugoc/d-30-1/src/csv.h File Reference	. 26
5.2.1 Function Documentation	. 27
5.2.1.1 calcmedia_csv()	. 27
5.2.1.2 ficheiro_csv_1()	. 29
5.2.1.3 ficheiro_csv_2()	. 30
5.2.1.4 ficheiro_csv_3()	. 32
5.2.1.5 listar_plano_csv()	. 33
5.2.1.6 tabela_csv()	. 34
5.3 /home/hugoc/d-30-1/src/main.c File Reference	. 37
5.3.1 Detailed Description	. 37
5.3.2 Function Documentation	. 38
5.3.2.1 contador_calorias()	. 38
5.3.2.2 listagem()	. 39
5.3.2.3 main()	. 40
5.3.2.4 periodo()	. 41
5.4 /home/hugoc/d-30-1/src/struct.h File Reference	. 42
5.5 /home/hugoc/d-30-1/src/tsv.c File Reference	. 43
5.5.1 Function Documentation	. 44
5.5.1.1 calcmedia_tsv()	. 44
5.5.1.2 ficheiro_tsv_1()	. 46
5.5.1.3 ficheiro_tsv_2()	. 47
5.5.1.4 ficheiro_tsv_3()	. 49
5.5.1.5 listar_plano_tsv()	. 50
5.5.1.6 tabela_tsv()	. 51
5.6 /home/hugoc/d-30-1/src/tsv.h File Reference	. 53
5.6.1 Function Documentation	. 54
5.6.1.1 calcmedia_tsv()	. 55
5.6.1.2 ficheiro_tsv_1()	. 57
5.6.1.3 ficheiro_tsv_2()	. 58
5.6.1.4 ficheiro_tsv_3()	. 60
5.6.1.5 listar_plano_tsv()	. 61

		iii
	5.6.1.6 tabela_tsv()	62
Index		65

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

2 Trabalho Prático

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.

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 3

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.

4 Trabalho Prático

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Dados		
	Esta estrutura contém arrays para armazenar o número do cliente, o nome do cliente e o número de telefone associado a cada cliente	9
Dieta		
	Esta estrutura contém arrays para armazenar o número do cliente, tipos de refeição, tipos de alimento, calorias, e a data de cada refeição	10
Plano		
	Esta estrutura contém arrays para armazenar o número do cliente, data tipos de refeição, calorias minimas e maximas	11

6 Data Structure Index

Chapter 3

File Index

3.1 File List

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

home/hugoc/d-30-1/src/csv.c	. 15
home/hugoc/d-30-1/src/csv.h	. 26
home/hugoc/d-30-1/src/main.c	. 37
home/hugoc/d-30-1/src/struct.h	. 42
home/hugoc/d-30-1/src/tsv.c	. 43
home/hugoc/d-30-1/src/tsv.h	. 53

8 File Index

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.

4.3 Plano Struct Reference 11

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

```
\verb|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 Plano Struct Reference

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

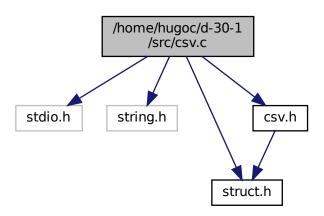
Chapter 5

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()

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

```
340
                         c1_pequeno++;
                                                                              //**< Contador */
341
                     if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */</pre>
342
343
                         cal_almoco1 = cal_almoco1 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
344
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 */</pre>
350
                         c1_jantar++;
                                                                            //**< Contador */
351
352
                }
353
354
                 if (dieta->n_cliente[i] == 2) //**< Verifica o numero de cliente */</pre>
355
                     if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição */
356
357
358
                         cal_pequeno2 = cal_pequeno2 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
359
                         c2_pequeno++;
                                                                              //**< Contador */
360
361
                     if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
362
                         cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
363
364
                                                                            //**< Contador */
                         c2_almoco++;
365
366
                     if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */
367
                         cal_jantar2 = cal_jantar2 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
368
369
                         c2_jantar++;
                                                                            //**< Contador */
370
                     }
371
                }
372
373
                 if (dieta->n_cliente[i] == 3) //**< Verifica o numero de cliente */</pre>
374
                     if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição */</pre>
375
376
                     {
377
                         cal_pequeno3 = cal_pequeno3 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
378
                                                                              //**< Contador */
                         c3_pequeno++;
379
380
                     if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
381
                         cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
382
383
                                                                            //**< Contador */
                         c3_almoco++;
384
385
                     if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */
386
                         {\tt cal\_jantar3 = cal\_jantar3 + dieta->calorias[i]; //**< Acumula as calorias */}
387
388
                         c3_jantar++;
                                                                            //**< Contador */
389
390
                }
391
            }
392
        }
393
        // Cliente 1
394
        if (cal_pequeno1 != 0) //**< Se as calorias forem differentes de 0 entra no loop */
395
396
397
            float media_cal_pequeno1 = cal_pequeno1 / c1_pequeno;
            //**< Realiza a media */ printf("\tMédia das calorias consumidas pelo cliente 1 ao pequeno almoço: .2f\n",
398
       media\_cal\_pequeno1); //** < Apresenta a media */
399
400
401
        if (cal_almocol != 0) //**< Se as calorias forem differentes de 0 entra no loop */
402
403
            float media_cal_almoco1 = cal_almoco1 / c1_almoco;
       //**< Realiza a media
            printf("\tMédia das calorias consumidas pelo cliente 1 ao almoço: %.2f\n", media_cal_almocol);
404
       //**< Apresenta a media */
405
406
407
        if (cal_jantar1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop */
408
            float media_cal_jantar1 = cal_jantar1 / c1_jantar;
409
       //**< Realiza a media */
printf("\tMédia das calorias consumidas pelo cliente 1 ao jantar: %.2f\n", media_cal_jantar1);
410
       //**< Apresenta a media */
411
412
        // Cliente 2
413
        if (cal_pequeno2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop \ */
414
415
416
            float media_cal_pequeno2 = cal_pequeno2 / c2_pequeno;
               //**< Realiza a media
            printf("\t^média das calorias consumidas pelo cliente 2 ao pequeno almoço: .2f\n^m,
417
       media_cal_pequeno2); //**< Apresenta a media</pre>
418
```

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

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.

```
18
       if (strcmp(arquivo, "dados.csv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
19
20
21
           int i = 0;
          FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, com o nome ficheiro.
22
23
           if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
2.4
2.5
              printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
26
                                                                    //**< Retorno indicando falha na
               return;
      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++;
45
          fclose(ficheiro); //**< Fecha o arquivo. */</pre>
46
47
48
      else if (strcmp(arquivo, "dados_csv.bin") == 0) //**< Compara as strings, caso sejam iguais entra
49
      neste loop */
50
52
53
         FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
      nome ficheiro. */
54
           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
                                                                    //**< Retorno indicando falha na
      abertura do arquivo.*/
59
          printf("\nDados clientes:\n\n");
60
66
           for (int i = 0; i < 3; i++)
67
              68
      a ficheiros binários */
69
              fread(dados->nome_cliente[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente</pre>
      a dados binários do tipo string */
70
              fread(&dados->num_telefone[i], sizeof(int), 1, ficheiro);  //**< Modo leitura relativamente</pre>
       a dados binários interios */
71
76
           for (int i = 0; i < 3; i++)
78
              printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
       dados->num_telefone[i]);
79
80
          fclose(ficheiro); //**< Fecha o arquivo. */</pre>
81
82
      }
      else //**< Outros casos */
8.5
          printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro */</pre>
86
87
88
```

5.1.1.3 ficheiro_csv_2()

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

Parameters

	arquivo	Nome do arquivo CSV a ser lido na função.
ſ	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.

```
98
           (strcmp(arquivo, "dietas.csv") == 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
101
       ficheiro. */
            char linha[max];
102
                                                     //**< Tamanho para armazenar cada linha do arquivo. */
103
104
             if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
105
                 printf("Erro: N\Bar{a}o foi possível abrir \$s\n", arquivo); //**< Apresenta uma mensagem de erro*/
106
                                                                           //**< Retorno indicando falha na
107
                 return;
       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. 
 \star/
115
                {
                     break; //**< Sai do loop se não houver mais linhas para ler. */</pre>
116
117
                 }
118
                 sscanf(linha, "%d;%d-%d-%d;%[^;];%[^;];%d ", &dieta->n_cliente[i], &dieta->dia[i],
126
       &dieta->mes[i], &dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], &dieta->calorias[i]);
printf("\t%04d %02d-%02d-%d %s %s %d \n", dieta->n_cliente[i], dieta->dia[i], dieta->mes[i],
127
       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</pre>
       neste loop */
134
135
            FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
136
137
             if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
138
                 printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
139
                                                                           //**< Retorno indicando falha na
140
                 return:
       abertura do arquivo.*/
141
           }
142
            printf("\nDieta realizada pelo cliente:\n\n");
143
144
149
             for (int i = 0; i < 4; i++)
150
                 fread(&dieta->n_cliente[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a</pre>
151
       dados binários interios */
                fread(&dieta->dia[i], sizeof(int), 1, ficheiro);
                                                                            //**< Modo leitura relativamente a
152
       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 */
                fread(dieta->refeicao[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a</pre>
155
       dados binários do tipo string \star/
                fread(dieta->alimento[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a</pre>
156
       dados binários do tipo string */
                fread(&dieta->calorias[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a</pre>
157
       dados binários interios */
158
            for (int i = 0; i < 4; i++)
163
164
                printf("\t%04d %02d-%02d-%d %s %s %d \n", dieta->n_cliente[i], dieta->dia[i], dieta->mes[i],
165
       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 */</pre>
174
175
176 }
```

5.1.1.4 ficheiro csv 3()

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
            FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
188
       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
                printf("Erro: N\~{a}o foi poss\'{i}vel abrir \$s\n", arquivo); //**< Apresenta uma mensagem de erro*/
193
194
                                                                        //**< Retorno indicando falha na
                return:
       abertura do arquivo.*/
195
```

```
196
197
           printf("\nPlano nutricional:\n\n");
198
199
           for (int i = 0; i < 100; i++)
200
               if (fgets(linha, sizeof(linha), ficheiro) == NULL) //**< Verifica se as linhas do arquivo
201
      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. */</pre>
2.04
205
               sscanf(linha, "%d;%d-%d;%[^;];%d Cal, %d Cal", &plano->n_cliente[i], &plano->dia[i],
214
       &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
           fclose(ficheiro); //**< Fecha o arquivo após a leitura dos dados. */
217
218
219
       else if (strcmp(arquivo, "planos_csv.bin") == 0) //**< Compara as strings, caso sejam iguais entra</pre>
220
          FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
221
      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
                eturn;
                                                                   //**< 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
      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 */
                                                                      //**< Modo leitura relativamente a
240
               fread(&plano->ano[i], sizeof(int), 1, ficheiro);
      dados binários interios */
               fread(plano->refeicao[i], sizeof(char), max, ficheiro);  //**< Modo leitura relativamente a</pre>
241
       dados binários do tipo string */
242
               fread(&plano->calorias_min[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a</pre>
      dados binários interios */
               fread(&plano->calorias_max[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a</pre>
243
      dados binários interios */
244
248
           for (int i = 0; i < 3; i++)
249
      250
251
252
           fclose(ficheiro); //**< Fecha o arquivo. */</pre>
253
254
255
       else //**< Outros casos */
256
257
           printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro */</pre>
258
           return;
259
260 }
```

5.1.1.5 listar plano csv()

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 {
        int cliente;
                         //**< Armazena n° cliente*/
278
        char refeicao[20]; //**< Armazena o nome do cliente*/</pre>
279
        280
2.81
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
             \textbf{if ((mesinicio < plano->mes[i] \&\& plano->mes[i] < mesfim) } \\ || //**< Verifica o período se for 
287
       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))</pre>
290
                if (cliente == plano->n_cliente[i]) //**< Verifica se o cliente existe */</pre>
291
292
                {
                     if (strcmp(refeicao, plano->refeicao[i]) == 0) //**< Verifica se a refeição existe */</pre>
293
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()

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*/
```

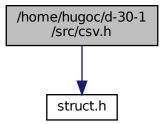
```
466
        float cal_pequeno3 = 0, cal_almoco3 = 0, cal_jantar3 = 0; //**< Armazena as calorias cliente 3*/
467
468
        for (int i = 0; i < 6; i++)
469
470
471
            if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */</pre>
472
473
                 if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição */
474
475
                     cal pequeno1 = cal pequeno1 + dieta->calorias[i]; //**< Soma as calorias */
476
477
                 if (strcmp(dieta->refeiçao[i], "almoco") == 0) //**< Verifica a refeição */
478
479
                    cal_almocol = cal_almocol + dieta->calorias[i]; //**< Soma as calorias */</pre>
480
481
                 if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */</pre>
482
                    cal jantar1 = cal jantar1 + dieta->calorias[i]; //**< Soma as calorias */</pre>
483
484
485
            }
486
487
            if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */</pre>
488
                 if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */
489
490
491
                     cal_pequeno2 = cal_pequeno2 + dieta->calorias[i]; //**< Soma as calorias */</pre>
492
493
                 if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
494
495
                     cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Soma as calorias */</pre>
496
497
                 if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */
498
499
                     cal_jantar2 = cal_jantar2 + dieta->calorias[i]; //**< Soma as calorias */</pre>
500
501
            }
502
503
            if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */</pre>
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 */</pre>
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
                if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */</pre>
513
514
515
                     cal jantar3 = cal jantar3 + dieta->calorias[i]: //**< Soma as calorias */
516
517
518
519
        printf("\nTabela:\n\n");
520
521
                                                                                                                   --\n");
       printf("\t---
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</pre>
       que contém planos, para realizar a aprsentação do nome correto \ \star /
530
531
532
                     if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */</pre>
533
534
                         if (strcmp(plano->refeicao[j], " jantar") == 0) //**< Verifica a refeição */</pre>
535
           printf("\t| %04d | %s | %-16s | %02d-%02d-2023 | %02d-%02d-2023 | %d | %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
536
       dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
       cal_jantar1); //**< Apresenta linha da tabela */</pre>
537
538
                         else if (strcmp(plano->refeicao[i], " almoco") == 0) //**< Verifica a refeição */</pre>
539
                             printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d
540
       %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_almocol); //**< Apresenta linha da tabela */</pre>
541
                         else if (strcmp(plano->refeicao[j], " pequeno almoço") == 0) //**< Verifica a</pre>
542
```

```
refeição */
543
                                                                                printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d
544
                    \label{eq:continuous} $0.2f \mid n", plano->n\_cliente[j], dados->nome\_cliente[i], plano->refeicao[j], dieta->dia[i], plano->refeicao[j], dieta->refeicao[j], dieta->dia[i], plano->refeicao[j], dieta->dia[i], plano->refeicao[j], dieta->dia[i], plano->refeicao[j], dieta->dia[i], plano->refeicao[j], dieta->dia[i], plano->refeicao[j], dieta->refeicao[j], 
                    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
                    cal pequeno1); //**< Apresenta linha da tabela */
546
                                                                     else //**< Outros casos */
547
                    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],
548
                    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 */</pre>
552
553
                                                                     if (strcmp(plano->refeicao[j], " jantar") == 0) //**< Verifica a refeição */</pre>
554
555
                                                                                                                                               %s | %-16s | %02d-%02d-2023 | %02d-%02d-2023 | %d
556
                                                                               printf("\t| %04d |
                    %d | %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
                    \label{lem:dieta-mes} \verb| dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias\_min[j], plano->calorias\_max[j], plano-
                    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
                    \label{eq:cliente} $0.2f \mid \n", plano->n\_cliente[j], dados->nome\_cliente[i], plano->refeicao[j], dieta->dia[i], plano->refeicao[j], plano->refei
                    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
                    cal_almoco2); //**< Apresenta linha da tabela */</pre>
561
                                                                     else if (strcmp(plano->refeicao[j], " pequeno almoço") == 0) //**< Verifica a</pre>
                    refeição */
563
                                                                                printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d
564
                    %d | %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i], dieta->mes[i], plano->calorias_min[j], plano->calorias_max[j],
                    cal_pequeno2); //**< Apresenta linha da tabela */</pre>
565
566
                                                                    else //**< Outros casos */
567
                                                                    {
                    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->calorias_min[j], plano->calorias_max[j]); //**<
568
                    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
                   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],
577
                    cal_jantar3); //**< Apresenta linha da tabela */</pre>
578
                                                                     else if (strcmp(plano->refeicao[j], " almoço") == 0) //**< Verifica a refeição */
579
580
                    printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d | %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
581
                    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
                    cal_almoco3); //**< Apresenta linha da tabela */
582
                                                                     else if (strcmp(plano->refeicao[j], " pequeno almoço") == 0) //**< Verifica a</pre>
583
                    refeição */
584
585
                                                                               printf("\t| %04d |
                                                                                                                                                %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |
                    %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 */</pre>
586
                                                                    else //**< Outros casos */
587
588
                                                                                printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d
589
                    %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");
```

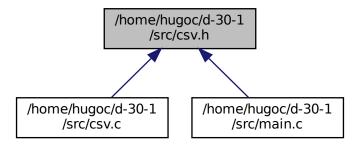
596 }

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()

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

```
342
                    if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
343
344
                        cal_almocol = cal_almocol + dieta->calorias[i]; //**< Acumula as calorias */</pre>
                        c1_almoco++;
                                                                         //**< Contador */
345
346
                    if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */
347
348
349
                        cal_jantar1 = cal_jantar1 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
350
                                                                         //**< Contador */
                        c1_jantar++;
351
352
                }
353
354
                if (dieta->n_cliente[i] == 2) //**< Verifica o numero de cliente */</pre>
355
356
                    if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição */
357
                        358
359
                                                                           //**< Contador */
                        c2_pequeno++;
360
361
                    if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
362
363
                        cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
364
                        c2_almoco++;
                                                                         //**< Contador */
365
                    if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */
366
367
368
                        cal_jantar2 = cal_jantar2 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
                                                                         //**< Contador */
369
370
371
                }
372
373
                if (dieta->n_cliente[i] == 3) //**< Verifica o numero de cliente */</pre>
374
375
                    if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição */</pre>
376
                        cal_pequeno3 = cal_pequeno3 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
377
378
                                                                           //**< Contador */
                        c3_pequeno++;
379
380
                    if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
381
382
                        cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
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 */</pre>
388
                                                                         //**< Contador */
                        c3_jantar++;
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
            printf("\tMédia das calorias consumidas pelo cliente 1 ao pequeno almoço: %.2f\n",
398
       media_cal_pequenol); //**< Apresenta a media */</pre>
399
400
401
        if (cal almocol != 0) //**< Se as calorias forem differentes de 0 entra no loop */
402
403
            float media_cal_almoco1 = cal_almoco1 / c1_almoco;
       //** < Realiza a media
            printf("\tMedia das calorias consumidas pelo cliente 1 ao almoço: \$.2f\n", media\_cal\_almocol);
404
       //**< Apresenta a media */
405
406
407
        if (cal_jantarl != 0) //**< Se as calorias forem differentes de 0 entra no loop */
408
409
            float media_cal_jantar1 = cal_jantar1 / c1_jantar;
       //**< Realiza a media */
printf("\tMédia das calorias consumidas pelo cliente 1 ao jantar: %.2f\n", media_cal_jantar1);
410
       //**< Apresenta a media */
411
412
413
        // Cliente 2
414
        if (cal_pequeno2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop */
415
       {
            float media_cal_pequeno2 = cal_pequeno2 / c2_pequeno;
416
              //**< Realiza a media
            printf("\t M\'edia das calorias consumidas pelo cliente 2 ao pequeno almoço: <math>.2f\n",
417
       media_cal_pequeno2); //**< Apresenta a media</pre>
418
419
        if (cal almoco2 != 0) //**< Se as calorias forem differentes de 0 entra no loop */
420
```

```
421
       {
            float media_cal_almoco2 = cal_almoco2 / c2_almoco;
       //**< Realiza a media */
printf("\tMédia das calorias consumidas pelo cliente 2 ao almoço: %.2f\n", media_cal_almoco2);
423
      //**< 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 */
printf("\tMédia das calorias consumidas pelo cliente 2 ao jantar: %.2f\n", media_cal_jantar2);
429
      //**< 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</pre>
437
438
439
        if (cal\_almoco3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop */
            float media_cal_almoco3 = cal_almoco3 / c3_almoco;
441
       //**< Realiza a media
           printf("\tMédia das calorias consumidas pelo cliente 3 ao almoço: %.2f\n", media_cal_almoco3);
442
      //**< Apresenta a media */
443
444
445
       if (cal_jantar3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop */</pre>
446
447
            float media_cal_jantar3 = cal_jantar3 / c3_jantar;
      //**< Realiza a media
           printf("\tMédia das calorias consumidas pelo cliente 3 ao jantar: %.2f\n", media_cal_jantar3);
448
      //**< Apresenta a media */
449
450 }
```

5.2.1.2 ficheiro_csv_1()

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.
       if (strcmp(arquivo, "dados.csv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
18
19
       {
20
21
           int i = 0;
22
           FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, com o nome ficheiro.
2.3
           if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
24
25
               printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
26
27
                                                                    //**< Retorno indicando falha na
               return;
       abertura do arquivo.*/
2.8
29
          printf("\nDados clientes:\n\n");
30
31
       40
41
               printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
42
       dados->num_telefone[i]);
43
              i++;
44
45
46
          fclose(ficheiro); //**< Fecha o arquivo. */
47
48
       else if (strcmp(arquivo, "dados_csv.bin") == 0) //**< Compara as strings, caso sejam iguais entra
49
       neste loop */
50
51
          int i = 0;
52
          FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
53
      nome ficheiro. */
           if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
55
56
57
               printf("Erro: N\~{a}o foi poss\'{ivel 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
              fread(&dados->n_cliente[i], sizeof(int), 1, ficheiro);
                                                                         //**< Modo leitura relativamente
68
       a ficheiros binários */
               fread(dados->nome_cliente[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente</pre>
69
       a dados binários do tipo string */
               fread(&dados->num_telefone[i], sizeof(int), 1, ficheiro);  //**< Modo leitura relativamente</pre>
70
       a dados binários interios \star/
71
76
           for (int i = 0; i < 3; i++)
78
               printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
       dados->num_telefone[i]);
79
80
          fclose(ficheiro); //**< Fecha o arquivo. */
81
      }
84
       else //**< Outros casos */
85
86
          printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro */</pre>
87
           return:
88
89 }
```

5.2.1.3 ficheiro_csv_2()

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.

```
98
           (\mathsf{strcmp}\,(\mathsf{arquivo},\,\,\mathsf{"dietas.csv"}) \,\,=\,\, 0) \,\,\,//\!*\,\star\!<\,\, \mathsf{Compara}\,\,\mathsf{as}\,\,\mathsf{strings},\,\,\mathsf{caso}\,\,\mathsf{sejam}\,\,\mathsf{iguais}\,\,\mathsf{entra}\,\,\mathsf{neste}\,\,\mathsf{loop}
99
100
             FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
101
       ficheiro. */
            char linha[max];
102
                                                      //**< Tamanho para armazenar cada linha do arquivo. */
103
104
             if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
105
                 printf("Erro: N\Bar{a}o foi possível abrir \$s\n", arquivo); //**< Apresenta uma mensagem de erro*/
106
                                                                             //**< Retorno indicando falha na
107
                  return;
       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. 
 \star/
115
                     break; //**< Sai do loop se não houver mais linhas para ler. */</pre>
116
117
                 }
118
                 sscanf(linha, "%d;%d-%d-%d;%[^;];%[^;];%d ", &dieta->n_cliente[i], &dieta->dia[i],
126
       &dieta->mes[i], &dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], &dieta->calorias[i]);
printf("\t%04d %02d-%02d-%d %s %s %d \n", dieta->n_cliente[i], dieta->dia[i], dieta->mes[i],
127
       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</pre>
       neste loop */
134
135
             FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
136
137
             if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
138
                 printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
139
                                                                             //**< Retorno indicando falha na
140
                 return:
       abertura do arquivo.*/
141
           }
142
             printf("\nDieta realizada pelo cliente:\n\n");
143
144
149
             for (int i = 0; i < 4; i++)
150
                 151
       dados binários interios */
                 fread(&dieta->dia[i], sizeof(int), 1, ficheiro);
                                                                              //**< Modo leitura relativamente a
152
       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 */
                                                    fread(dieta->refeicao[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a</pre>
155
                       dados binários do tipo string \star/
                                                   fread (dieta->alimento[i], \ size of (char), \ max, \ ficheiro); \ //**< \ Modo \ leitura \ relativamente \ a \ Additional (char) \ a \ Additional (
156
                       dados binários do tipo string */
                                                   fread(&dieta->calorias[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a</pre>
157
                       dados binários interios */
 158
                                        for (int i = 0; i < 4; i++)
 163
 164
                                                    printf("\t%04d %02d-%02d-%d %s %s %d \n", dieta->n_cliente[i], dieta->dia[i], dieta->mes[i],
 165
                       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
                         }
175
176 }
```

5.2.1.4 ficheiro csv 3()

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.

```
185 {
186
        if (strcmp(arquivo, "planos.csv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
187
           FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
188
       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
                printf("Erro: N\~{a}o foi poss\'{vel abrir $s\n", arquivo); //**< Apresenta uma mensagem de erro*/
193
                                                                        //**< Retorno indicando falha na
194
                return:
       abertura do arquivo.*/
195
            }
```

```
196
197
            printf("\nPlano nutricional:\n\n");
198
199
            for (int i = 0; i < 100; i++)
200
                if (fgets(linha, sizeof(linha), ficheiro) == NULL) //**< Verifica se as linhas do arquivo
201
       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. */</pre>
204
205
                sscanf(linha, "%d;%d-%d;%[^;];%d Cal, %d Cal", &plano->n_cliente[i], &plano->dia[i],
214
       &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</pre>
       neste loop */
220
           FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
2.2.1
       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
                 eturn;
                                                                      //**< 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
       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 */
                fread(plano->refeicao[i], sizeof(char), max, ficheiro);  //**< Modo leitura relativamente a</pre>
241
       dados binários do tipo string */
242
               fread(&plano->calorias_min[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a</pre>
       dados binários interios \star/
                fread(&plano->calorias_max[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a</pre>
243
       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 */</pre>
258
            return;
259
260 }
```

5.2.1.5 listar plano csv()

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 {
       int cliente;
                       //**< Armazena n° cliente*/
278
       char refeicao[20]; //**< Armazena o nome do cliente*/</pre>
279
       280
281
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
      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))</pre>
290
              if (cliente == plano->n_cliente[i]) //**< Verifica se o cliente existe */
291
292
              {
                  if (strcmp(refeicao, plano->refeicao[i]) == 0) //**< Verifica se a refeição existe */</pre>
293
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()

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.

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 */</pre>
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->refeiçao[i], "almoco") == 0) //**< Verifica a refeição */
478
479
                     cal_almocol = cal_almocol + dieta->calorias[i]; //**< Soma as calorias */</pre>
480
481
                 if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */</pre>
482
                     cal jantar1 = cal jantar1 + dieta->calorias[i]; //**< Soma as calorias */</pre>
483
484
485
            }
486
487
            if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */</pre>
488
                 if (strcmp(dieta->refeicao[i], " pequeno almoço") == 0) //**< Verifica a refeição */
489
490
491
                     cal_pequeno2 = cal_pequeno2 + dieta->calorias[i]; //**< Soma as calorias */</pre>
492
493
                 if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
494
495
                     cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Soma as calorias */</pre>
496
497
                 if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */
498
499
                     cal_jantar2 = cal_jantar2 + dieta->calorias[i]; //**< Soma as calorias */</pre>
500
501
            }
502
503
            if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */</pre>
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 */</pre>
508
                 if (strcmp(dieta->refeicao[i], " almoço") == 0) //**< Verifica a refeição */
509
510
511
                     cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Soma as calorias */
512
                if (strcmp(dieta->refeicao[i], " jantar") == 0) //**< Verifica a refeição */</pre>
513
514
515
                     cal jantar3 = cal jantar3 + dieta->calorias[i]: //**< Soma as calorias */</pre>
516
517
518
519
        printf("\nTabela:\n\n");
520
521
                                                                                                                    -\n");
       printf("\t--
522
        printf("\t| NP
                           | Paciente | Tipo Refeição | Inicio | Fim | Mínimo | Máximo
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</pre>
       que contém planos, para realizar a aprsentação do nome correto \ \star /
530
531
532
                     if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */</pre>
533
534
                         if (strcmp(plano->refeicao[j], " jantar") == 0) //**< Verifica a refeição */</pre>
535
           printf("\t| %04d | %s | %-16s | %02d-%02d-2023 | %02d-%02d-2023 | %d | %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
536
       dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
       cal_jantar1); //**< Apresenta linha da tabela */</pre>
537
538
                         else if (strcmp(plano->refeicao[i], " almoco") == 0) //**< Verifica a refeição */</pre>
539
                             printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d
540
       %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_almocol); //**< Apresenta linha da tabela */</pre>
541
                         else if (strcmp(plano->refeicao[j], " pequeno almoço") == 0) //**< Verifica a</pre>
542
```

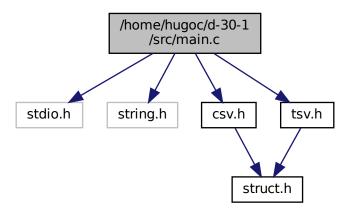
```
refeição */
543
                                                                                printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d
544
                    \label{eq:continuous} $$ 0.2f \mid n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i], plano->refeicao[j], dieta->refeicao[j], dieta->r
                    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
cal_pequenol); //**
Apresenta linha da tabela */
546
                                                                     else //**< Outros casos */
547
                    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],
548
                    Apresenta linha da tabela */
549
550
551
                                                          if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */</pre>
552
553
                                                                     if (strcmp(plano->refeicao[j], " jantar") == 0) //**< Verifica a refeição */</pre>
554
555
                                                                                                                                               %s | %-16s | %02d-%02d-2023 | %02d-%02d-2023 | %d
556
                                                                               printf("\t| %04d |
                    %d | %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
                    \label{lem:dieta-mes} \verb||dieta->mes[i]|, plano->dia[j]|, plano->mes[j]|, plano->calorias_min[j]|, plano->calorias_max[j]|, plano->calorias_max[j
                    cal_jantar2); //**< Apresenta linha da tabela */</pre>
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
                    \label{eq:cliente} $0.2f \mid \n", plano->n\_cliente[j], dados->nome\_cliente[i], plano->refeicao[j], dieta->dia[i], plano->refeicao[j], plano->refeicao[j], dieta->dia[i], plano->refeicao[j], plano->refei
                    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
                    cal_almoco2); //**< Apresenta linha da tabela */
561
                                                                     else if (strcmp(plano->refeicao[j], " pequeno almoço") == 0) //**< Verifica a</pre>
                    refeição */
563
                                                                                printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d
564
                   %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 */</pre>
565
566
                                                                    else //**< Outros casos */
567
                    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->calorias_min[j], plano->calorias_max[j]); //**<
568
                    Apresenta linha da tabela */
569
570
571
572
                                                          if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */</pre>
574
                                                                     if (strcmp(plano->refeicao[j], " jantar") == 0) //**< Verifica a refeição */</pre>
575
576
                   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],
577
                    cal_jantar3); //**< Apresenta linha da tabela */</pre>
578
                                                                     else if (strcmp(plano->refeicao[j], " almoço") == 0) //**< Verifica a refeição */
579
580
                    printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d | %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
581
                    dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
                    cal_almoco3); //**< Apresenta linha da tabela */
582
                                                                     else if (strcmp(plano->refeicao[j], " pequeno almoço") == 0) //**< Verifica a</pre>
583
                    refeição */
584
585
                                                                               printf("\t| %04d |
                                                                                                                                                %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |
                    %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 */</pre>
586
                                                                    else //**< Outros casos */
587
588
                                                                                printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d
589
                    %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

Copyright (c) 2023

5.3.2 Function Documentation

5.3.2.1 contador_calorias()

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.

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 */
      int contador = 0; //**< Realiza a contagem dos clientes que ultrapassaram as calorias inseridas */
60
61
      \verb|printf("\nDigite o valor de calorias que pretende ver quantos utilizadores ultrapassaram:");\\
63
      scanf("%f", &calorias);
64
      for (int i = 0; i < 4; i++)
65
66
          if ((mesinicio < dieta->mes[i] && dieta->mes[i] < mesfim) || //** Verifica o período se for
      68
69
              (mesfim == dieta->mes[i] && diainicio <= dieta->dia[i] && dieta->dia[i] <= diafim))</pre>
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
      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.

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 {
       int j, k, temp;
                                           //**< Variáveis auxiliares */
98
       int clientes_verificados[4] = {0}; //**< Array para armazenar os clientes verificados */
99
100
        char nomes_ordenados[max][max]; //**< Matriz para armazenar os nomes dos clientes verificados */
101
102
        for (j = 0; j < 4; j++)
103
            if ((mesinicio < dieta->mes[j] && dieta->mes[j] < mesfim) || //**< Verifica o período se for
104
       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))</pre>
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
111
112
                        if (strcmp(dieta->refeicao[j], plano->refeicao[k]) == 0) //**< Verifica se as</pre>
       refeições existem em planos \star/
113
                             if (dieta->calorias[j] < plano->calorias_min[k] || dieta->calorias[j] >
114
       plano->calorias_max[k]) //**< Verifica se as calorias estão fora do intervalo defenido */
115
116
                                 if (dados->n_cliente[j] != 0)
117
                                     clientes_verificados[j] = dados->n_cliente[j];
                                                                                          //**< Armazena os
118
       clientes verificados num variavel da função*/
                                     strcpy(nomes_ordenados[j], dados->nome_cliente[j]); //**< Armazena os</pre>
119
       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];
clientes_verificados[k] = temp;
 136
137
 138
                                                                               char temp_nome[50];
139
                                                                               strcpy(temp_nome, nomes_ordenados[j]);
140
                                                                                strcpy(nomes_ordenados[j], nomes_ordenados[k]);
141
                                                                               \verb|strcpy|(nomes_ordenados[k]|, temp_nome)|;
 142
                                               }
 143
 144
 145
 146
                                \ensuremath{//} Imprimir os clientes ordenados
 147
                               printf("\nListagem dos pacientes ordenada por ordem decrescente:\n\n");
                                for (j = 0; j < 4; j++)
 148
 149
 150
                                                if (clientes_verificados[j] != 0)
 151
                                               {
 152
                                                               printf("\t^{804d \ %s\n"}, \ clientes\_verificados[j], \ nomes\_ordenados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"}, \ clientes\_verificados[j]); \ //**< Apresenta \ os \ ("\t^{804d \ %s\n"
                            clientes ordenados */
153
                                               }
154
                                }
155 }
```

5.3.2.3 main()

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

Função principal do programa.

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

```
164 {
165
         int modo;
        int diainicio = 0, diafim = 0; //**< Armazena dias */
int mesinicio = 0, mesfim = 0; //**< Armazena meses */</pre>
166
167
168
169
        if (argc < 2) //**< Verifica se o numero de argumentos */</pre>
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 */</pre>
176
            modo = 1;
177
         else if (strcmp(argv[1], "-tsv") == 0) //**< Verifica se o argumento é -tsv */
178
            modo = 2;
        else if (strcmp(argv[1], "-ajuda") == 0) //**< Verifica se o argumento é -ajuda */
179
180
            modo = 3;
        else //**< Outros casos */
181
182
        {
183
             printf("Caso necessite de ajuda: s -ajuda n, argv[0]);
184
185
186
        Dados dados; //**< Invocação da struct dados */
187
         Dieta dieta; //**< Invocação da struct dieta */
188
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 */
            listagem(&dieta, &plano, &dados, diainicio, mesinicio, diafim, mesfim); //**< Invoca a função</pre>
200
       contador_calorias*/
            listar_plano_csv(&plano, diainicio, mesinicio, diafim, mesfim);
201
                                                                                       //**< 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 */</pre>
208
209
            ficheiro_tsv_1(argv[2], &dados);
                                                                                       //**< Invoca a função
       ficheiro_tsv_1*/
210
                                                                                       //**< Invoca a função
            ficheiro tsv 2(argv[3], &dieta);
       ficheiro_tsv_2 */
211
            ficheiro_tsv_3(argv[4], &plano);
                                                                                       //**< Invoca a função
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 */
            listar_plano_tsv(&plano, diainicio, mesinicio, diafim, mesfim);
                                                                                       //**< Invoca a função
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 */</pre>
2.2.2
223
            printf("\nEste programa utiliza os seguintes formatos (CSV ou TSV). Existindo também uma versão
       binária de cada ficheiro \n");
            printf("\nModo de utilização:\n");
224
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]);
    printf("\tPara utilizar com arquivos (TSV|bin), execute: %s -tsv dados.tsv dietas_tsv.bin
228
       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.

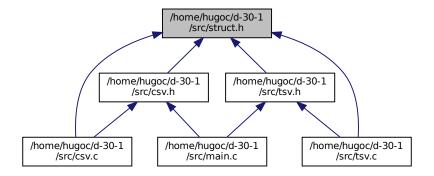
Parameters

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

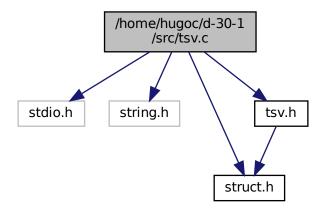
#define min 10

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()

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.

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
                                                                 //**< Armazena as contagens */
312
313
       //**< Armazena as contagens */
314
315
316
       float c3_pequeno = 0, c3_almoco = 0, c3_jantar = 0;
317
                                                                //**< Armazena as contagens */
       float cal_pequeno3 = 0, cal_almoco3 = 0, cal_jantar3 = 0; //**< Armazena as calorias */
318
319
320
       printf("\nMédia de todos os clientes:\n\n");
321
322
       for (i = 0; i < 6; i++)
323
           if ((mesinicio < dieta->mes[i] && dieta->mes[i] < mesfim) || //**< Válida o período */
  (mesinicio == dieta->mes[i] && diainicio <= dieta->dia[i] && dieta->dia[i] <= diafim) ||</pre>
324
325
               (mesfim == dieta->mes[i] && diainicio <= dieta->dia[i] && dieta->dia[i] <= diafim))</pre>
326
327
328
               if (dieta->n_cliente[i] == 1) //**< Verifica o numero de cliente */</pre>
329
330
                   if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */
331
332
333
                       cal_pequeno1 = cal_pequeno1 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
334
                       c1_pequeno++;
335
                   if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */
336
337
                       cal_almoco1 = cal_almoco1 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
338
339
340
341
                   if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */</pre>
342
                       cal_jantar1 = cal_jantar1 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
343
344
                                                                       //**< Contador
                       c1 jantar++;
345
               }
```

```
347
348
                 if (dieta->n_cliente[i] == 2) //**< Verifica o numero de cliente */</pre>
349
350
                     if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */
351
                         cal_pequeno2 = cal_pequeno2 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
352
353
                         c2_pequeno++;
                                                                             //**< Contador */
354
355
                     if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */
356
                         cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
357
                                                                           //**< Contador */
358
                         c2 almoco++;
359
                     if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */
360
361
362
                         cal_jantar2 = cal_jantar2 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
363
                         c2_jantar++;
                                                                           //**< Contador */
364
                     }
365
                }
366
                if (dieta->n_cliente[i] == 3) //**< Verifica o numero de cliente */</pre>
367
368
                     if (strcmp(dieta->refeicao[i], "pequeno almoco") == 0) //**< Verifica a refeição */
369
370
371
                         cal_pequeno3 = cal_pequeno3 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
372
                         c3_pequeno++;
                                                                             //**< Contador */
373
374
                     if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */
375
                         cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
376
377
                                                                           //**< Contador */
                         c3 almoco++;
378
379
                     if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */
380
381
                         cal_jantar3 = cal_jantar3 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
                                                                           //**< Contador */
382
                         c3_jantar++;
383
                     }
384
                }
385
            }
386
        }
387
        // Cliente 1
388
        if (cal_pequeno1 != 0) //**< Se as calorias forem diferentes de 0 entra no loop */
389
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 */</pre>
393
394
395
        if (cal_almocol != 0) //**< Se as calorias forem differentes de 0 entra no loop */
396
397
            float media_cal_almoco1 = cal_almoco1 / c1_almoco;
       //**< Realiza a media */
printf("\tMédia das calorias consumidas pelo cliente 1 ao almoço: %.2f\n", media_cal_almocol);
398
       //**< Apresenta a media */
399
400
401
        if (cal_jantarl != 0) //**< Se as calorias forem differentes 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
        // Cliente 2
407
        if (cal_pequeno2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop */
408
409
410
            float media_cal_pequeno2 = cal_pequeno2 / c2_pequeno;
               //**< Realiza a media
            printf("\t M\'edia das calorias consumidas pelo cliente 2 ao pequeno almoço: <math>.2f\n",
411
       media_cal_pequeno2); //**< Apresenta a media</pre>
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 */
printf("\tMédia das calorias consumidas pelo cliente 2 ao almoço: %.2f\n", media_cal_almoco2);
417
       //**< Apresenta a media */
418
419
420
        if (cal_jantar2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop */
421
            float media_cal_jantar2 = cal_jantar2 / c2_jantar;
422
       //**< Realiza a media
```

```
423
            printf("\tMédia das calorias consumidas pelo cliente 2 ao jantar: %.2f\n", media_cal_jantar2);
       //**< Apresenta a media */
424
425
        // Cliente 3
426
        if (cal_pequeno3 != 0) //**< Se as calorias forem diferentes de 0 entra no loop */
427
428
429
            float media_cal_pequeno3 = cal_pequeno3 / c3_pequeno;
               //**< Realiza a media
            \label{lem:printf("\tM\'edia das calorias consumidas pelo cliente 3 ao pequeno almoço: \$.2f\n",
430
       media_cal_pequeno3); //**< Apresenta a media</pre>
431
432
433
        if (cal_almoco3 != 0) //**< Se as calorias forem differentes de 0 entra no loop */
434
435
            float media_cal_almoco3 = cal_almoco3 / c3_almoco;
       //**< Realiza a media */
printf("\tMédia das calorias consumidas pelo cliente 3 ao almoço: %.2f\n", media_cal_almoco3);
436
       //**< Apresenta a media */
437
438
439
       if (cal_jantar3 != 0) //**< Se as calorias forem differentes de 0 entra no loop */
440
            float media_cal_jantar3 = cal_jantar3 / c3_jantar;
441
       //**< Realiza a media */
printf("\tMédia das calorias consumidas pelo cliente 3 ao jantar: %.2f\n", media_cal_jantar3);
       //**< Apresenta a media */
113
444 }
```

5.5.1.2 ficheiro_tsv_1()

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.

```
26
          {
              printf("Erro: Não foi possível abrir sn'', arquivo); //**< Apresenta uma mensagem de erro*/
28
                                                                 //**< Retorno indicando falha na
      abertura do arquivo.*/
29
30
31
         printf("\nDados clientes:\n\n");
41
          &dados->num_telefone[i]) == 3)
42
              printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
43
      dados->num_telefone[i]);
44
45
46
         fclose(ficheiro); //**< Fecha o arquivo. */
47
48
49
      else if (strcmp(arquivo, "dados_tsv.bin") == 0) //**< Compara as strings, caso sejam iguais entra</pre>
      neste loop */
50
51
         FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
      nome ficheiro. \star/
52
53
          int i = 0;
54
55
          if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
56
              printf("Erro: N\~{a}o foi poss\'{ivel abrir o arquivo\n"}); \ //**< Apresenta uma mensagem de erro*/
57
58
                                                               //**< 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
      a dados binários interios */
70
              fread(dados->nome_cliente[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente</pre>
      a dados binários do tipo string */
             fread(&dados->num_telefone[i], sizeof(int), 1, ficheiro);  //**< Modo leitura relativamente</pre>
71
      a dados binários interios \star/
72
76
          for (int i = 0; i < 3; i++)
78
              printf("\t^{04}d \ ^{s} \ ^{d}n", \ dados->n\_cliente[i], \ dados->nome\_cliente[i],
      dados->num_telefone[i]);
79
80
          fclose(ficheiro);
81
82
83
      else //**< Outros casos */
84
85
          printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro */</pre>
86
          return;
87
      }
88 }
```

5.5.1.3 ficheiro tsv 2()

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

linh	na	Nome da linha do arquivo CSV a ser lido na função.
die	ta	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.

```
if (strcmp(arquivo, "dietas.tsv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
98
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
                                                                                                                                                                             //**< 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
                                       if (fgets(linha, sizeof(linha), ficheiro) == NULL) //**< Verifica se as linhas do arquivo
113
                 CSV estão vazias. */
114
                                      {
115
                                                 break; //**< Sai do loop se não houver mais linhas para ler. */</pre>
116
                                       }
117
                                       sscanf(linha, "%d\t%d-%d-%d-%d)\t%9[^\t]\t%d ", &dieta->n\_cliente[i], &dieta->dia[i], &dieta
126
                 &dieta->mes[i], &dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], &dieta->calorias[i]);
printf("\t%04d %02d-%02d-%d %s %s %d \n", dieta->n_cliente[i], dieta->dia[i], dieta->mes[i],
127
                 dieta->ano[i], dieta->refeicao[i], dieta->alimento[i], dieta->calorias[i]);
128
129
                              fclose(ficheiro); //**< Fecha o arquivo após a leitura dos dados. */
130
131
132
                    else if (strcmp(arquivo, "dietas_tsv.bin") == 0) //**< Compara as strings, caso sejam iguais entra</pre>
                 neste loop */
133
                   {
134
                            FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
                 nome ficheiro. */
135
                              if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
136
137
138
                                       printf("Erro: Não foi possível abrir sn'', 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
                                       fread(&dieta->n_cliente[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a</pre>
150
                 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 */
                                       fread(dieta->refeicao[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a</pre>
154
                 dados binários do tipo string */
155
                                       fread(dieta->alimento[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a</pre>
                 dados binários do tipo string */
156
                                       fread(&dieta->calorias[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a</pre>
                 dados binários interios */
157
161
                              for (int i = 0; i < 4; i++)
 162
163
                                       printf("\t^{8}04d \ ^{0}2d-^{0}2d-^{0}k \ ^{8}s \ ^{d} \ ^{n}", \ dieta->n\_cliente[i], \ dieta->dia[i], \ dieta->mes[i], \ dieta->n\_cliente[i], \ dieta->n\_cli
                 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 }</pre>
```

5.5.1.4 ficheiro_tsv_3()

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.

```
if (strcmp(arquivo, "planos.tsv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
182
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
            if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
187
188
                printf("Erro: Não foi possível abrir %s\n", arquivo); //**< Apresenta uma mensagem de erro*/
189
190
                                                                        //**< Retorno indicando falha na
                 return;
       abertura do arquivo.*/
191
192
           printf("\nPlano nutricional:\n\n");
193
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]); printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal\n", plano->n_cliente[i], plano->dia[i],
211
       plano->mes[i], plano->ano[i], plano->refeicao[i], plano->calorias_min[i], plano->calorias_max[i]);
```

```
213
                        fclose(ficheiro); //**< Fecha o arquivo após a leitura dos dados. */
215
                else if (strcmp(arquivo, "planos_tsv.bin") == 0) //**< Verifica se as linhas do arquivo CSV estão</pre>
              vazias. */
216
                        FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
217
              nome ficheiro. */
218
219
                         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
220
                                 printf("Erro: N\~{a}o foi possível abrir $s\n", arquivo); //**< Apresenta uma mensagem de erro*/
221
                                                                                                                                                   //**< Retorno indicando falha na
222
                                 return:
              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 */
                                 fread(&plano->mes[i], sizeof(int), 1, ficheiro);
2.3.3
                                                                                                                                                          //**< Modo leitura relativamente a
              dados binários interios */
234
                                 fread(&plano->ano[i], sizeof(int), 1, ficheiro);
                                                                                                                                                          //**< Modo leitura relativamente a
              dados binários interios */
                                 fread(plano->refeicao[i], sizeof(char), max, ficheiro);
235
                                                                                                                                                          //**< Modo leitura relativamente a
               dados binários do tipo string */
                                 fread(\&plano->calorias\_min[i], \ sizeof(int), \ 1, \ ficheiro); \ //**< Modo \ leitura \ relativamente \ a \ fread(\&plano->calorias\_min[i], \ sizeof(int), \ 1, \ ficheiro); \ //** = Modo \ leitura \ relativamente \ a \ fread(\&plano->calorias\_min[i], \ sizeof(int), \ 1, \ ficheiro); \ //** = Modo \ leitura \ relativamente \ a \ fread(\&plano->calorias\_min[i], \ sizeof(int), \ 1, \ ficheiro); \ //** = Modo \ leitura \ relativamente \ a \ fread(\&plano->calorias\_min[i], \ sizeof(int), \ 1, \ sizeof(int),
236
              dados binários interios */
237
                                 fread(&plano->calorias_max[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a</pre>
              dados binários interios */
238
242
                         for (int i = 0; i < 3; i++)
243
                                 printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal \n", plano->n_cliente[i], plano->dia[i],
244
              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. */</pre>
248
249
                else //**< Outros casos */
250
251
                         printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro */</pre>
252
253
254 }
```

5.5.1.5 listar_plano_tsv()

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.

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
                                                                          //**< Armazena n° cliente*/
                      int cliente;
272
                      char refeicao[20]; //**< Armazena o nome do cliente*/
273
274
                      \texttt{printf("} \\ \texttt{nDigite o cliente e a refeição que pretende visualizar (numero refeição):");}
                      scanf("%d\t %[^\n]", &cliente, refeicao); //**< Realiza a leitura */</pre>
275
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
                                  \textbf{if ((mesinicio < plano->mes[i] \&\& plano->mes[i] < mesfim) } \\ || //**< Verifica o período se for its following that the second of the plano->mes[i] is the second of 
281
                   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))</pre>
284
                                            if (cliente == plano->n_cliente[i]) //**< Verifica se o cliente existe */</pre>
285
286
287
                                                        if (strcmp(refeicao, plano->refeicao[i]) == 0) //**< Verifica se a refeição existe */
288
                                                        {
                                                                  printf("\t%04d %02d-%02d-%d %s %d Cal, %d Cal\n", plano->n_cliente[i],
289
                   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
                                            }
293
294 }
```

5.5.1.6 tabela tsv()

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.

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

```
456 {
        {\tt float\ cal\_pequeno1\ =\ 0,\ cal\_almoco1\ =\ 0,\ cal\_jantar1\ =\ 0;\ //**< Armazena\ as\ calorias\ cliente\ 1*/}
457
        float cal_pequeno2 = 0, cal_almoco2 = 0, cal_jantar2 = 0; //**< Armazena as calorias cliente 2*/
458
        float cal_pequeno3 = 0, cal_almoco3 = 0, cal_jantar3 = 0; //**< Armazena as calorias cliente 3*/
459
460
461
        for (int i = 0; i < 6; i++)
462
463
             if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */</pre>
464
465
                 if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */</pre>
466
467
                     cal_pequeno1 = cal_pequeno1 + dieta->calorias[i]; //**< Soma as calorias */</pre>
468
469
                 if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */</pre>
470
472
                     cal_almoco1 = cal_almoco1 + dieta->calorias[i]; //**< Soma as calorias */</pre>
473
474
                 if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */</pre>
475
476
                     cal_jantar1 = cal_jantar1 + dieta->calorias[i]; //**< Soma as calorias */</pre>
477
478
```

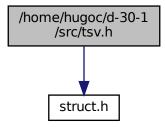
```
if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */</pre>
480
481
                          if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */
482
483
484
                                cal pequeno2 = cal pequeno2 + dieta->calorias[i]: //**< Soma as calorias */
485
                          if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */</pre>
486
487
488
                                cal almoco2 = cal almoco2 + dieta->calorias[i]; //**< Soma as calorias */
489
                          if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */</pre>
490
491
492
                                cal_jantar2 = cal_jantar2 + dieta->calorias[i];
493
494
                   }
495
                   if (dieta->n cliente[i] == 3) //**< Verifica o cliente */
496
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 */</pre>
501
                          if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */</pre>
502
503
504
                                cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Soma as calorias */</pre>
505
506
                          if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */</pre>
507
508
                                cal_jantar3 = cal_jantar3 + dieta->calorias[i]; //**< Soma as calorias */</pre>
509
510
                   }
511
512
513
             printf("\nTabela:\n\n");
514
           printf("\t---
                                                                                                                                                                          ----\n");
             printf("\t| NP | Paciente | Tipo Refeição | Inicio | Fim | Mínimo | Máximo |
515
           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 \ \star /
523
524
525
                                if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */</pre>
526
527
                                       if (strcmp(plano->refeicao[j], "jantar") == 0) //**< Verifica a refeição */</pre>
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 */</pre>
531
                                      else if (strcmp(plano->refeicao[j], "almoço") == 0) //**< Verifica a refeição */
532
533
534
                                             printf("\t| %04d |
                                                                                %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 |
           %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_almocol); //**< Apresenta linha da tabela */</pre>
535
                                       else if (strcmp(plano->refeicao[j], "pequeno almoço") == 0) //**< Verifica a</pre>
536
           refeição */
537
                                             printf("\t| \$04d | \$s | \$-17s | \$02d-\$02d-2023 | \$02d-\$02d-2023 | \$d
538
           %d | %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
           \label{linear_max} 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
           %d | ----\n ", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
           Apresenta linha da tabela */
543
544
545
                                 if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */</pre>
546
                                       if (strcmp(plano->refeicao[j], "jantar") == 0) //**< Verifica a refeição */</pre>
547
548
```

```
549
                        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],
550
                        \label{limits} dieta->mes[i], ~plano->dia[j], ~plano->mes[j], ~plano->calorias\_min[j], ~plano->calorias\_max[j], ~plano-
                        cal_jantar2); //**< Apresenta linha da tabela */</pre>
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
                         \label{eq:continuous} $$ 0.2f \mid n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i], plano->refeicao[j], dieta->refeicao[j], dieta->r
                        \label{linear_mass} \verb|dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j], plano->ca
                        cal_almoco2); //**< Apresenta linha da tabela */</pre>
555
                                                                                    else if (strcmp(plano->refeicao[j], "pequeno almoço") == 0) //**< Verifica a
                         refeição */
557
                        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->calorias_min[j], plano->calorias_max[j],
558
                        cal_pequeno2); //**< Apresenta linha da tabela */</pre>
559
560
                                                                                    else //**< Outros casos */
561
                        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->calorias_min[j], plano->calorias_max[j]); //**<
562
                         Apresenta linha da tabela */
563
564
565
                                                                      if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */</pre>
566
567
568
                                                                                     if (strcmp(plano->refeicao[j], "jantar") == 0) //**< Verifica a refeição */
569
570
                                                                                                 printf("\t| %04d | %s | %-16s | %02d-%02d-2023 | %02d-%02d-2023 | %d
571
                        %d | %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i], dieta->mes[i], 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 */</pre>
574
                         printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d | %0.2f |\n", plano->n_cliente[j], dados->nome_cliente[i], plano->refeicao[j], dieta->dia[i],
575
                        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 | %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 */</pre>
580
581
                                                                                    else //**< Outros casos */
582
                                                                                                 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
                                                                      }
                                                      }
587
588
                           }
589
                        printf("\t-----
                                                                                                                                                                                                                                                                                                                                                    ----\n");
590 }
```

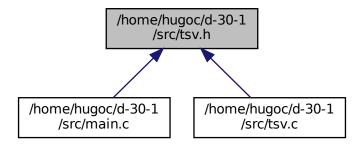
5.6 /home/hugoc/d-30-1/src/tsv.h File Reference

#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()

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.

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*/
        float c1_pequeno = 0, c1_almoco = 0, c1_jantar = 0;
311
                                                                   //**< Armazena as contagens */
       float cal_pequeno1 = 0, cal_almoco1 = 0, cal_jantar1 = 0; //**< Armazena as calorias */
313
       314
                                                                   //**< Armazena as contagens */
315
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) \mid \mid
326
                (\texttt{mesfim} == \texttt{dieta->mes[i]} \& \& \texttt{diainicio} <= \texttt{dieta->dia[i]} \& \& \texttt{dieta->dia[i]} <= \texttt{diafim}))
327
328
                if (dieta->n cliente[i] == 1) //**< Verifica o numero de cliente */</pre>
329
330
                    if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */
331
332
                        cal_pequeno1 = cal_pequeno1 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
333
334
                        c1_pequeno++;
                                                                            //**< Contador */
335
336
                    if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */</pre>
337
338
                        cal_almocol = cal_almocol + dieta->calorias[i]; //**< Acumula as calorias */</pre>
339
                                                                          //**< Contador */
340
                    if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */
341
342
                        cal_jantar1 = cal_jantar1 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
343
344
                                                                          //**< Contador */
345
346
                }
347
348
                if (dieta->n_cliente[i] == 2) //**< Verifica o numero de cliente */</pre>
349
350
                    if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */
351
                        cal_pequeno2 = cal_pequeno2 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
352
353
                                                                            //**< Contador
                        c2_pequeno++;
354
355
                    if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */</pre>
```

```
356
                    {
                        cal_almoco2 = cal_almoco2 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
357
                        c2_almoco++;
358
                                                                           //**< Contador */
359
                     if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */
360
361
362
                        cal_jantar2 = cal_jantar2 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
363
364
365
                }
366
                if (dieta->n cliente[i] == 3) //**< Verifica o numero de cliente */
367
368
                     if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */
369
370
371
                         cal_pequeno3 = cal_pequeno3 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
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 */</pre>
377
                        c3_almoco++;
                                                                           //**< Contador */
378
379
                     if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */</pre>
380
381
                        cal_jantar3 = cal_jantar3 + dieta->calorias[i]; //**< Acumula as calorias */</pre>
382
                                                                           //**< 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
                                      */
            printf("\tMédia das calorias consumidas pelo cliente 1 ao pequeno almoço: %.2f\n",
392
       media_cal_pequeno1); //**< Apresenta a media</pre>
393
394
395
        if (cal almocol != 0) //**< Se as calorias forem diferentes de 0 entra no loop */
396
            float media_cal_almoco1 = cal_almoco1 / c1_almoco;
397
       //**< Realiza a media */
398
            printf("\tMédia das calorias consumidas pelo cliente 1 ao almoço: %.2f\n", media_cal_almocol);
       //**< Apresenta a media */
399
400
        if (cal_jantar1 != 0) //**< Se as calorias forem differentes de 0 entra no loop */
401
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 differentes 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 */</pre>
412
413
        if (cal_almoco2 != 0) //**< Se as calorias forem differentes de 0 entra no loop */
414
415
            float media_cal_almoco2 = cal_almoco2 / c2_almoco;
416
       //**< Realiza a media
417
            printf("\t M\'edia das calorias consumidas pelo cliente 2 ao almoço: \$.2f\n", media\_cal\_almoco2);
       //**< Apresenta a media */
418
419
        if (cal_jantar2 != 0) //**< Se as calorias forem diferentes de 0 entra no loop */
420
421
422
            float media_cal_jantar2 = cal_jantar2 / c2_jantar;
       //**< Realiza a media */
printf("\tMédia das calorias consumidas pelo cliente 2 ao jantar: %.2f\n", media_cal_jantar2);
423
       //**< 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 */</pre>
431
432
433
        if (cal_almoco3 != 0) //**< Se as calorias forem differentes de 0 entra no loop */
434
435
            float media_cal_almoco3 = cal_almoco3 / c3_almoco;
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 differentes de 0 entra no loop */
440
441
            float media_cal_jantar3 = cal_jantar3 / c3_jantar;
       //**< Realiza a media */
printf("\tMédia das calorias consumidas pelo cliente 3 ao jantar: %.2f\n", media_cal_jantar3);
442
      //**< Apresenta a media */
443
```

5.6.1.2 ficheiro_tsv_1()

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.

```
if (strcmp(arquivo, "dados.tsv") == 0) //** Compara as strings, caso sejam iguais entra neste loop
2.0
21
      {
         FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
22
      ficheiro. */
         int i = 0;
23
25
         if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
26
             printf("Erro: N\~{a}o foi possível abrir $s\n", arquivo); //**< Apresenta uma mensagem de erro*/
27
28
                                                             //**< Retorno indicando falha na
             return;
      abertura do arquivo.*/
29
        }
30
         printf("\nDados clientes:\n\n");
31
32
41
         &dados->num_telefone[i]) == 3)
```

```
{
               printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
43
       dados->num_telefone[i]);
44
               i++;
4.5
46
           fclose(ficheiro); //**< Fecha o arquivo. */</pre>
48
49
       else if (strcmp(arquivo, "dados_tsv.bin") == 0) //**< Compara as strings, caso sejam iguais entra</pre>
       neste loop */
50
           FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
51
       nome ficheiro. */
52
53
           int i = 0;
54
           if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
55
56
57
               printf("Erro: Não foi possível abrir o arquivo\n"); //**< Apresenta uma mensagem de erro*/
58
                                                                     //**< Retorno indicando falha na abertura
                return;
       do arquivo.*/
59
           printf("\nDados clientes:\n\n");
60
61
           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</pre>
       a dados binários do tipo string \star/
71
               fread(&dados->num_telefone[i], sizeof(int), 1, ficheiro);  //**< Modo leitura relativamente</pre>
       a dados binários interios */
72
76
           for (int i = 0; i < 3; i++)
77
               printf("\t%04d %s %d\n", dados->n_cliente[i], dados->nome_cliente[i],
78
       dados->num_telefone[i]);
79
80
           fclose(ficheiro);
82
       else //**< Outros casos */
8.3
84
85
           printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro */</pre>
86
87
88 }
```

5.6.1.3 ficheiro_tsv_2()

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.

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.
                  if (strcmp(arquivo, "dietas.tsv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
98
99
                 {
100
                             FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
                  ficheiro. *
                            char linha[max];
101
                                                                                                                          //**< Tamanho para armazenar cada linha do arquivo. */
102
                              if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
103
104
 105
                                       printf("Erro: Não foi possível abrir s\n", arquivo); //**< Apresenta uma mensagem de erro*/
106
                                                                                                                                                                            //**< Retorno indicando falha na
                 abertura do arquivo.*/
107
108
109
                             printf("\nDieta realizada pelo cliente:\n\n");
110
                              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. */</pre>
116
117
126
                                       sscanf(linha, "\$d\t\$d-\$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]);
    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]);
127
128
129
130
                              fclose(ficheiro); //**< Fecha o arquivo após a leitura dos dados. */
131
                   else if (strcmp(arquivo, "dietas_tsv.bin") == 0) //**< Compara as strings, caso sejam iguais entra
132
                 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 sn'', arquivo); //**< Apresenta uma mensagem de erro*/
139
                                                                                                                                                                            //**< Retorno indicando falha na
                                       return;
                 abertura do arquivo.*/
140
141
142
                             printf("\nDieta realizada pelo cliente:\n\n");
143
148
                              for (int i = 0; i < 4; i++)
149
                                       fread(&dieta->n_cliente[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a</pre>
150
                  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 */
                                       fread(dieta->refeicao[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a</pre>
154
                 dados binários do tipo string */
155
                                       fread(dieta->alimento[i], sizeof(char), max, ficheiro); //**< Modo leitura relativamente a</pre>
                  dados binários do tipo string */
156
                                       fread(&dieta->calorias[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a</pre>
                  dados binários interios */
157
161
                              for (int i = 0; i < 4; i++)
162
                                       printf("\t^{04d} \ ^{02d-\02d-\02d} \ ^{s} \ ^{s} \ ^{d} \ ^{n}", \ dieta->n\_cliente[i], \ dieta->dia[i], \ dieta->mes[i], 
163
                  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 */</pre>
171
                    }
172
173 }
```

5.6.1.4 ficheiro_tsv_3()

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.

```
182
         if (strcmp(arquivo, "planos.tsv") == 0) //**< Compara as strings, caso sejam iguais entra neste loop
183
            FILE *ficheiro = fopen(arquivo, "r"); //**< Abre o ficheiro em modo leitura, como o nome
184
       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
                 printf("Erro: N\~{a}o foi possível abrir $s\n", arquivo); //**< Apresenta uma mensagem de erro*/
189
                                                                            //**< Retorno indicando falha na
190
                  return;
       abertura do arquivo.*/
191
192
             printf("\nPlano nutricional:\n\n");\\
193
194
195
             for (int i = 0; i < 100; i++)
196
             {
                 if (fgets(linha, sizeof(linha), ficheiro) == NULL) //**< Verifica se as linhas do arquivo
197
       CSV estão vazias. */
198
199
                     break; //**< Sai do loop se não houver mais linhas para ler. */
200
201
                 sscanf(linha, "%d \t %d-%d-%d \t %99[^{t}] \t %d Cal, %d Cal", &plano->n_cliente[i],
210
        &plano->dia[i], &plano->mes[i], &plano->ano[i], plano->refeicao[i], &plano->calorias_min[i],
       &plano->calorias_max[i]);

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]);
211
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
             FILE *ficheiro = fopen(arquivo, "rb"); //**< Abre o ficheiro em modo leitura (binaria), com o
217
       nome ficheiro. */
218
             if (ficheiro == NULL) //**< Verifica se o ficheiro é nulo, caso seja entra no ciclo*/
219
220
                 printf("Erro: N\~{a}o foi poss\'{vel abrir $s\n", arquivo); //**< Apresenta uma mensagem de erro*/
221
222
                                                                            //**< Retorno indicando falha na
                  return:
       abertura do arquivo.*/
223
             }
```

```
224
225
            printf("\nPlano nutricional:\n\n");
229
            for (int i = 0; i < 3; i++)
230
2.31
                fread(&plano->n_cliente[i], sizeof(int), 1, ficheiro);
                                                                            //**< Modo leitura relativamente a
       dados binários interios */
                fread(&plano->dia[i], sizeof(int), 1, ficheiro);
232
                                                                            //**< Modo leitura relativamente a
       dados binários interios */
233
                fread(&plano->mes[i], sizeof(int), 1, ficheiro);
                                                                            //**< Modo leitura relativamente a
       dados binários interios */
                fread(&plano->ano[i], sizeof(int), 1, ficheiro);
234
                                                                            //**< 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</pre>
       dados binários interios */
                fread(&plano->calorias_max[i], sizeof(int), 1, ficheiro); //**< Modo leitura relativamente a</pre>
237
       dados binários interios */
238
242
            for (int i = 0; i < 3; i++)
243
                printf("\t%04d %02d-%02d-%d %s %d Cal \n", plano->n_cliente[i], plano->dia[i],
244
       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. */</pre>
248
249
        else //**< Outros casos */
250
251
            printf("\nErro: Ficheiro inválido\n"); //**< Mensagem de erro */</pre>
252
            return:
253
```

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.

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

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

5.6.1.6 tabela tsv()

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.

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

```
489
490
                           if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */</pre>
491
492
                                  cal_jantar2 = cal_jantar2 + dieta->calorias[i];
493
494
                    }
495
496
                    if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */</pre>
497
498
                           if (strcmp(dieta->refeicao[i], "pequeno almoço") == 0) //**< Verifica a refeição */</pre>
499
500
                                 cal_pequeno3 = cal_pequeno3 + dieta->calorias[i]; //**< Soma as calorias */</pre>
501
                           if (strcmp(dieta->refeicao[i], "almoço") == 0) //**< Verifica a refeição */
502
503
504
                                  cal_almoco3 = cal_almoco3 + dieta->calorias[i]; //**< Soma as calorias */</pre>
505
                           if (strcmp(dieta->refeicao[i], "jantar") == 0) //**< Verifica a refeição */</pre>
506
507
508
                                 cal_jantar3 = cal_jantar3 + dieta->calorias[i]; //**< Soma as calorias */</pre>
509
510
511
             }
512
             printf("\nTabela:\n\n");
513
514
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
                           if \ (dados->n\_cliente[i] \ == \ plano->n\_cliente[j]) \ //**< \ Responsável \ por \ verificar \ os \ clientes \\ 
522
           que contém planos, para realizar a aprsentação do nome correto */
523
                          -{
524
525
                                 if (dieta->n_cliente[i] == 1) //**< Verifica o cliente */</pre>
526
                                        if (strcmp(plano->refeicao[j], "jantar") == 0) //**< Verifica a refeição */</pre>
527
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],
           \label{limits} \verb|dieta->mes[i]|, plano->dia[j]|, plano->mes[j]|, plano->calorias_min[j]|, plano->calorias_max[j]|, plan
           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_almocol); //**
Apresenta linha da tabela */
535
                                        else if (strcmp(plano->refeicao[j], "pequeno almoço") == 0) //**< Verifica a
            refeição */
537
                                             printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d
538
            %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 */</pre>
539
                                        else //**< Outros casos */
540
541
                                              printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d |
542
            %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
                                  if (dieta->n_cliente[i] == 2) //**< Verifica o cliente */
545
546
547
                                        if (strcmp(plano->refeicao[j], "jantar") == 0) //**< Verifica a refeição */
548
549
           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 */
550
551
552
                                        else if (strcmp(plano->refeicao[j], "almoço") == 0) //**< Verifica a refeição */</pre>
553
                                              printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d
554
                   | %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 */</pre>
555
556
                                 else if (strcmp(plano->refeicao[j], "pequeno almoço") == 0) //**< Verifica a</pre>
          refeição */
557
                                 {
                                      printf("\t| %04d | %s | %-17s | %02d-%02d-2023 | %02d-%02d-2023 | %d
558
          %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 */</pre>
559
                                }
                                else //**< Outros casos */
560
561
                                 {
                                      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],
          \label{linear_mes_in_man_diag} dieta->mes[i], plano->calorias\_min[j], plano->calorias\_max[j]); //**< colorias\_max[i]); //**
         Apresenta linha da tabela */
563
564
565
                           if (dieta->n_cliente[i] == 3) //**< Verifica o cliente */</pre>
566
567
                                 if (strcmp(plano->refeicao[j], "jantar") == 0) //**< Verifica a refeição */</pre>
568
569
570
                                     printf("\t| %04d | %s | %-16s | %02d-%02d-2023 | %02d-%02d-2023 | %d
571
          dieta->mes[i], plano->dia[j], plano->mes[j], plano->calorias_min[j], plano->calorias_max[j],
         cal_jantar3); //**< Apresenta linha da tabela */</pre>
572
                                }
                                else if (strcmp(plano->refeicao[j], "almoço") == 0) //**< Verifica a refeição */
573
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</pre>
         refeição */
578
         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],
579
         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
         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],
583
         dieta->mes[i], plano->calorias_min[j], plano->calorias_max[j]); //**<
         Apresenta linha da tabela */
584
585
586
                     }
                }
587
588
           }
589
         printf("\t-----
590 }
```

Index

/home/hugocid-30-1/src/csv.n, 15 alimento, 10 /home/hugocid-30-1/src/sv.h, 26 ano, 10 /home/hugocid-30-1/src/src/main.c, 37 calorias, 10 /home/hugocid-30-1/src/stv.c, 43 mes, 11 /home/hugocid-30-1/src/tsv.c, 43 mes, 11 /home/hugocid-30-1/src/tsv.h, 53 n_cliente, 11 alimento refeicao, 11 Dieta, 10 ficheiro_csv_1 ano csv.c, 18 Dieta, 10 csv.c, 18 plano, 12 ficheiro_csv_2 caloredia_csv csv.c, 19 csv.c, 19 csv.h, 30 csv.c, 19 csv.h, 30 csv.c, 19 csv.h, 30 csv.b, 27 csv.c, 21 caloredia_csv csv.h, 30 csv.b, 32 tsv.b, 32 tsv.c, 44 ficheiro_csv_1 tsv.b, 54 tsv.c, 46 calorias tsv.h, 57 Dieta, 10 ficheiro_tsv_2 calorias min ficheiro_tsv_2 Plano, 12 tsv.b, 58 calorias tsv.c, 47 plano, 12		
/home/hugoc/d-30-1/src/main.c, 37 /home/hugoc/d-30-1/src/structh, 42 /home/hugoc/d-30-1/src/stv.c, 43 /home/hugoc/d-30-1/src/tsv.c, 43 /home/hugoc/d-30-1/src/tsv.c, 43 /home/hugoc/d-30-1/src/tsv.h, 53 alimento Dieta, 10 Casv.c, 18 Casv.c, 19 Calcmedia_csv Csv.c, 16 Csv.c, 19 Csv.c, 19 Csv.c, 19 Csv.c, 21 Calcmedia_tsv Csv.c, 21 Calcmedia_tsv Csv.c, 44 tsv.h, 54 tsv.h, 54 tsv.h, 54 Calorias Dieta, 10 Dieta, 11 Plano, 12 Dados, 9	/home/hugoc/d-30-1/src/csv.c, 15	alimento, 10
/home/hugoc/d-30-1/src/struct.h, 42 /home/hugoc/d-30-1/src/tsv.c, 43 mes, 11 /home/hugoc/d-30-1/src/tsv.h, 53 n_cliente, 11 refeicao, 11 alimento Dieta, 10 ficheiro_csv_1 ano csv.c, 18 csv.h, 29 ficheiro_csv_2 csv.h, 29 ficheiro_csv_2 csv.c, 19 csv.c, 16 csv.c, 19 csv.h, 30 csv.c, 16 csv.h, 27 calcmedia_tsv csv.h, 32 tsv.c, 44 ficheiro_csv_1 tsv.h, 54 calorias tsv.h, 57 ficheiro_tsv_1 tsv.h, 57 ficheiro_tsv_2 calorias max tsv.h, 57 ficheiro_tsv_2 calorias min plano, 12 tsv.h, 58 ficheiro_tsv_2 calorias min plano, 12 calorias min ficheiro_tsv_3 tsv.c, 49 tsv.c, 49 tsv.h, 59 main.c, 38 csv.c listagem csv.c, 22 csv.h, 33 listar_plano_csv_2 tabela_csv, 27 ficheiro_csv_2, 19 csv.c, 22 csv.h, 33 listar_plano_tsv tsv.h, 61 calorias csv.ficheiro_csv_2, 29 ficheiro_csv_1, 29 ficheiro_csv_2, 30 main.c csv.c, 40 main.c do ficheiro_csv_3, 32 listar_plano_tsv tsv.h, 61 caloriadia_csv, 34 listar_plano_tsv tsv.h, 61 caloriadia_csv, 34 listar_plano_tsv tsv.h, 61 caloriadia_csv, 34 listar_plano_tsv contador_calorias, 38 listagem, 38 main, 40 periodo, 41 mes nome_cliente, 9 nome_cliente, 9 nome_cliente, 9 loieta, 11 plano, 12 loieta, 11 plano, 12 loados, 9	/home/hugoc/d-30-1/src/csv.h, 26	ano, 10
/home/hugoc/d-30-1/src/tsv.c, 43 /home/hugoc/d-30-1/src/tsv.h, 53	/home/hugoc/d-30-1/src/main.c, 37	calorias, 10
/home/hugoc/d-30-1/src/tsv.h, 53 alimento	/home/hugoc/d-30-1/src/struct.h, 42	dia, 11
Trefeicao, 11	/home/hugoc/d-30-1/src/tsv.c, 43	mes, 11
alimento Dieta, 10 Dieta, 10 Dieta, 10 Dieta, 10 Plano, 12 Calcmedia_csv csv.c, 16 csv.h, 27 calcmedia_tsv csv.c, 21 calorias Dieta, 10 csv.c, 16 csv.h, 27 calcmedia_tsv csv.c, 21 csv.h, 32 tsv.c, 44 tsv.h, 54 calorias Dieta, 10 calorias_max Dieta, 10 calorias_max Plano, 12 calorias_min Plano, 12 contador_calorias main.c, 38 csv.c calcmedia_csv, 16 ficheiro_csv_1, 18 ficheiro_csv_2, 1, 18 ficheiro_csv_2, 1, 18 ficheiro_csv_2, 19 ficheiro_csv_2, 20 tistar_plano_csv tsv.h, 33 listar_plano_tsv tabela_csv, 27 ficheiro_csv_2, 3 csv.h calcmedia_csv, 27 ficheiro_csv_2, 30 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 n_cliente, 9 nome_cliente, 9 nome_cliente, 9 nome_cliente, 9 num_telefone, 9 dia Dieta, 11 Plano, 12 Dados, 9	/home/hugoc/d-30-1/src/tsv.h, 53	n_cliente, 11
Dieta, 10 ano Dieta, 10 Dieta, 10 Plano, 12 calcmedia_csv csv.c, 16 csv.h, 27 calcmedia_tsv tsv.c, 44 tsv.h, 54 calorias Dieta, 10 Plano, 12 calorias_max Plano, 12 calorias_min Plano, 12 contador_calorias main.c, 38 csv.c calcmedia_csv, 16 ficheiro_csv_1 tsv.h, 59 main.c, 38 csv.c caloredia_csv calorias_min Plano, 12 contador_calorias main.c, 38 csv.c calcmedia_csv, 16 ficheiro_csv_2, 1, 18 ficheiro_csv_2, 19 ficheiro_csv_2, 19 ficheiro_csv_2, 21 csv.h, 33 listar_plano_csv, 22 tabela_csv, 23 csv.h calcmedia_csv, 27 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 34 Dados, 9 n_cliente, 9 nome_cliente, 9 nom		refeicao, 11
ano	alimento	
Dieta, 10 Plano, 12 Plano, 12 calcmedia_csv	Dieta, 10	
Plano, 12 calcmedia_csv	ano	csv.c, 18
calomedia_csv		
calcmedia_csv csv.h, 30 csv.c, 16 ficheiro_csv_3 csv.h, 27 csv.c, 21 calcmedia_tsv csv.h, 32 tsv.c, 44 ficheiro_tsv_1 tsv.h, 54 tsv.c, 46 calorias tsv.h, 57 Dieta, 10 ficheiro_tsv_2 calorias_max tsv.c, 47 Plano, 12 tsv.h, 58 calorias_min ficheiro_tsv_3 Plano, 12 tsv.c, 49 contador_calorias tsv.c, 49 main.c, 38 tsv.c, 59 main.c, 38 listagem calcmedia_csv, 16 main.c, 38 ficheiro_csv_1, 18 listar_plano_csv ficheiro_csv_2, 21 csv.c, 22 ficheiro_csv_3, 21 csv.c, 22 ficheiro_csv_3, 21 tsv.c, 50 tsv.h, 61 tsv.c, 50 csv.h tsv.c, 50 csv.h tsv.h, 61 calcmedia_csv, 27 main.c ficheiro_csv_1, 29 main.c ficheiro_csv_2, 3 contador_calorias, 38 listar_p	Plano, 12	
csv.c, 16 csv.h, 27 calcmedia_tsv		
csv.h, 27 calcmedia_tsv	_	
calcmedia_tsv		<u> </u>
tsv.c, 44 tsv.h, 54 calorias Dieta, 10 Calorias Dieta, 12 Calorias Dieta, 12 Calorias Dieta, 12 Calorias Dieta, 14 Calorias Dieta, 11 Plano, 12 Calorias Dieta, 11 Plano, 12 Tsv.c, 47 Tsv.h, 58 Csv.c, 47 Tsv.h, 58 Csv.c, 49 Tsv.c, 50 Tsv.h, 59 main.c, 38 Tsv.c, 22 Tsv.c, 22 Tsv.c, 22 Tsv.c, 22 Tsv.c, 22 Tsv.c, 22 Tsv.c, 33 Tsv.c, 50 Tsv.h, 61 Tsv.c, 40 Tsv.c, 47 Tsv.c, 49 Tsv.c, 4		
tsv.h, 54 calorias calorias Dieta, 10 Dieta, 10 Dieta, 10 Calorias_max Plano, 12 Calorias_min Plano, 12 contador_calorias main.c, 38 csv.c calcmedia_csv, 16 ficheiro_csv_2, 19 ficheiro_csv_2, 19 ficheiro_csv_2, 29 ficheiro_csv, 23 csv.h calcmedia_csv, 27 ficheiro_csv_1, 29 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 Dieta, 11 Plano, 12 ficheiro_es calorias main.c, 38 listar_plano_csv csv.c, 22 csv.h, 33 listar_plano_tsv tsv.c, 50 csv.h calcmedia_csv, 27 ficheiro_csv_1, 29 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dieta, 11 Plano, 12 Ticheiro_cs contador_calorias, 38 listagem, 38 main, 40 periodo, 41 mes Dieta, 11 Plano, 12 Dados, 9 Dados, 9	_	
calorias		ficheiro_tsv_1
Dieta, 10 ficheiro_tsv_2 calorias_max tsv.c, 47 Plano, 12 tsv.h, 58 calorias_min ficheiro_tsv_3 Plano, 12 tsv.c, 49 contador_calorias tsv.h, 59 main.c, 38 main.c, 38 csv.c listagem calcmedia_csv, 16 main.c, 38 ficheiro_csv_1, 18 listar_plano_csv ficheiro_csv_2, 19 csv.c, 22 ficheiro_csv_3, 21 csv.h, 33 listar_plano_csv tsv.h, 33 listar_plano_tsv tsv.c, 50 csv.h tsv.h, 61 calcmedia_csv, 23 main. csv.h, 22 main. ficheiro_csv_1, 29 main ficheiro_csv_2, 30 main.c ficheiro_csv_2, 30 main.c ficheiro_csv_3, 32 contador_calorias, 38 listagem, 38 main.c Dados, 9 periodo, 41 n_cliente, 9 poeta, 11 num_telefone, 9 pleta, 11 plano, 12 pados, 9		
calorias_max		
Plano, 12 calorias_min		ficheiro_tsv_2
calorias_min Plano, 12 contador_calorias main.c, 38 csv.c calcmedia_csv, 16 ficheiro_csv_1, 18 ficheiro_csv_2, 19 ficheiro_csv_3, 21 listar_plano_csv, 22 tabela_csv, 23 csv.h calcmedia_csv, 27 ficheiro_csv_1, 29 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 n_cliente, 9 nome_cliente, 9 num_telefone, 9 dia Dieta, 11 Plano, 12 Dieta, 11 Plano, 12 Iistav, 49 tsv.c, 49 tsv.c, 49 tsv.c, 49 tsv.c, 49 tsv.c, 49 tsv.c, 19 tsv.h, 59 main.c, 38 listagem main.c, 38 listar_plano_csv csv.c, 22 csv.h, 33 listar_plano_tsv tsv.c, 50 tsv.h, 61 main main.c main.c, 40 main.c contador_calorias, 38 listagem, 38 main, 40 periodo, 41 Plano, 12 Dados, 9 n_cliente Dados, 9	-	tsv.c, 47
Plano, 12 tsv.c, 49 contador_calorias tsv.h, 59 main.c, 38 tsv.h, 59 csv.c listagem calcmedia_csv, 16 main.c, 38 ficheiro_csv_1, 18 listar_plano_csv ficheiro_csv_2, 19 csv.c, 22 ficheiro_csv_3, 21 csv.h, 33 listar_plano_tsv tsv.c, 50 csv.h tsv.c, 50 csv.h tsv.h, 61 calcmedia_csv, 27 main ficheiro_csv_1, 29 main ficheiro_csv_2, 30 main.c ficheiro_csv_3, 32 main.c listar_plano_csv, 33 contador_calorias, 38 tabela_csv, 34 listagem, 38 main, 40 periodo, 41 mes Dieta, 11 num_telefone, 9 Dieta, 11 plano, 12 Dados, 9		
contador_calorias main.c, 38 csv.c	_	ficheiro_tsv_3
main.c, 38 csv.c		
csv.c listagem calcmedia_csv, 16 ficheiro_csv_1, 18 ficheiro_csv_2, 19 ficheiro_csv_3, 21 listar_plano_csv, 22 tabela_csv, 23 csv.h calcmedia_csv, 27 ficheiro_csv_1, 29 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 n_cliente, 9 nome_cliente, 9 nome_cliente, 9 num_telefone, 9 Dieta, 11 Plano, 12 listar_plano_csv sistar_plano_csv listar_plano_csv, 9 listar_plano_csv listar_plano_tsv listar_	_	tsv.h, 59
calcmedia_csv, 16 ficheiro_csv_1, 18 ficheiro_csv_2, 19 ficheiro_csv_3, 21 listar_plano_csv, 22 ficheiro_csv, 23 listar_plano_csv, 22 tabela_csv, 23 csv.h calcmedia_csv, 27 ficheiro_csv_1, 29 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 nome_cliente, 9 nome_clien		
ficheiro_csv_1, 18 ficheiro_csv_2, 19 ficheiro_csv_3, 21 listar_plano_csv, 22 ficheiro_csv_3, 21 listar_plano_csv, 22 tabela_csv, 23 csv.h calcmedia_csv, 27 ficheiro_csv_1, 29 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 nome_cliente, 9 nome_cliente, 9 nome_cliente, 9 nome_cliente, 9 lieta, 11 Plano, 12 listar_plano_csv listar_plano_tsv l		_
ficheiro_csv_2, 19 ficheiro_csv_3, 21 listar_plano_csv, 22 tabela_csv, 23 csv.h calcmedia_csv, 27 ficheiro_csv_1, 29 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 nome_cliente, 9 nome_cliente, 9 nome_cliente, 9 nome_cliente, 9 Dieta, 11 Plano, 12 csv.c, 22 csv.h, 33 listar_plano_tsv tsv.c, 50 tsv.h, 61 main main.c, 40 main.c contador_calorias, 38 listagem, 38 main, 40 periodo, 41 mes Dieta, 11 plano, 12 Dados, 9	- '	
ficheiro_csv_3, 21 listar_plano_csv, 22 tabela_csv, 23 csv.h calcmedia_csv, 27 ficheiro_csv_1, 29 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 n_cliente, 9 nome_cliente, 9 nome_client		
listar_plano_csv, 22 tabela_csv, 23 csv.h calcmedia_csv, 27 ficheiro_csv_1, 29 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 n_cliente, 9 nome_cliente, 9 nome_cliente		
tabela_csv, 23 csv.h calcmedia_csv, 27 ficheiro_csv_1, 29 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 n_cliente, 9 nome_cliente, 9 nome_cliente, 9 num_telefone, 9 dia Dieta, 11 Plano, 12 pados, 9 n_cliente Plano, 12 pados, 9 n_cliente Dados, 9		
csv.h calcmedia_csv, 27 ficheiro_csv_1, 29 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 n_cliente, 9 nome_cliente, 9 nome_cliente, 9 num_telefone, 9 dia Dieta, 11 Dieta, 11 Plano, 12 Dieto, 12 Dieto, 16 Dieto, 9 Dieto, 11 Dieto, 9 Dieto, 11 Dieto, 9 Dieto, 11 Dieto, 9 Dieto, 9 Dieto, 11 Dieto, 9 Dieto, 12 Dieto, 9 Dieto, 9 Dieto, 9 Dieto, 9 Dieto, 11 Dieto, 9		- · -
calcmedia_csv, 27 ficheiro_csv_1, 29 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 n_cliente, 9 nome_cliente, 9 nome_cliente, 9 num_telefone, 9 dia Dieta, 11 Dieta, 11 Dieta, 11 Plano, 12 main.c contador_calorias, 38 listagem, 38 main, 40 periodo, 41 mes Dieta, 11 Plano, 12 dia n_cliente Dados, 9		
ficheiro_csv_1, 29 ficheiro_csv_2, 30 ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 nome_cliente, 9 nome_cliente, 9 num_telefone, 9 Dieta, 11 Dieta, 11 Dieta, 11 Plano, 12 main.c contador_calorias, 38 listagem, 38 main, 40 periodo, 41 mes Dieta, 11 Plano, 12 dia n_cliente Dados, 9		tsv.h, 61
ficheiro_csv_2, 30 main.c, 40 ficheiro_csv_3, 32 main.c listar_plano_csv, 33 contador_calorias, 38 tabela_csv, 34 listagem, 38 main, 40 periodo, 41 n_cliente, 9 mes nome_cliente, 9 Dieta, 11 num_telefone, 9 Plano, 12 dia Dieta, 11 n_cliente Plano, 12 Dados, 9		
ficheiro_csv_3, 32 listar_plano_csv, 33 tabela_csv, 34 Dados, 9 n_cliente, 9 nome_cliente, 9 num_telefone, 9 Dieta, 11 Plano, 12 main.c contador_calorias, 38 listagem, 38 main, 40 periodo, 41 mes Dieta, 11 Plano, 12		
listar_plano_csv, 33 tabela_csv, 34 listagem, 38 main, 40 Dados, 9 periodo, 41 m_cliente, 9 nome_cliente, 9 num_telefone, 9 Dieta, 11 plano, 12 dia Dieta, 11 Plano, 12 Dados, 9		•
tabela_csv, 34 listagem, 38 main, 40 Dados, 9 periodo, 41 n_cliente, 9 mes nome_cliente, 9 Dieta, 11 num_telefone, 9 Plano, 12 dia Dieta, 11 n_cliente Plano, 12 Dados, 9		
main, 40 Dados, 9 periodo, 41 n_cliente, 9 mes nome_cliente, 9 Dieta, 11 num_telefone, 9 Plano, 12 dia Dieta, 11 n_cliente Plano, 12 Dados, 9		
Dados, 9 periodo, 41 n_cliente, 9 mes nome_cliente, 9 Dieta, 11 num_telefone, 9 Plano, 12 dia Dieta, 11 n_cliente Plano, 12 Dados, 9	tabela_csv, 34	
n_cliente, 9 nome_cliente, 9 num_telefone, 9 Dieta, 11 Plano, 12 dia Dieta, 11 Plano, 12 Dieta, 11 Dieta, 11 Dieta, 12 Dieta, 12 Dieta, 11	Dadas 0	ŕ
nome_cliente, 9 nome_cliente, 9 num_telefone, 9 dia Dieta, 11 Dieta, 11 Plano, 12 Dados, 9	•	•
num_telefone, 9 Plano, 12 dia Dieta, 11 n_cliente Plano, 12 Dados, 9	_	
dia Dieta, 11 Plano, 12 Dieta, 12 Dados, 9	-	· ·
Dieta, 11 n_cliente Plano, 12 Dados, 9	_ ,	Plano, 12
Plano, 12 Dados, 9		n alianta
	•	-
Dieta, 10		
	Dieta, 10	Dieta, 11

66 INDEX

```
Plano, 13
nome_cliente
     Dados, 9
num_telefone
     Dados, 9
periodo
     main.c, 41
Plano, 11
     ano, 12
    calorias_max, 12
    calorias_min, 12
     dia, 12
    mes, 12
     n_cliente, 13
     refeicao, 13
refeicao
     Dieta, 11
     Plano, 13
tabela_csv
    csv.c, 23
    csv.h, 34
tabela tsv
    tsv.c, 51
    tsv.h, 62
tsv.c
    calcmedia_tsv, 44
    ficheiro_tsv_1, 46
    ficheiro_tsv_2, 47
     ficheiro_tsv_3, 49
     listar_plano_tsv, 50
     tabela_tsv, 51
tsv.h
    calcmedia_tsv, 54
     ficheiro_tsv_1, 57
     ficheiro_tsv_2, 58
     ficheiro_tsv_3, 59
     listar_plano_tsv, 61
     tabela_tsv, 62
```