

Engenharia Informática

Estruturas de Dados

2019/2020

Relatório

Ficheiros de Texto e Binários

António Pereira Nº2019141051

David Ferreira Nº2019141281

Data: 05/04/2020

Índice

1	INTRODUÇÃO.....	3
2	TRABALHO REALIZADO	4
2.1	ESCREVA TRÊS SUBPROGRAMAS QUE EXECUTEM AS SEGUINTE TAREFAS:	4
2.2	ELABORE SUBPROGRAMAS QUE CONTABILIZEM, O NÚMERO DE LINHAS, O NÚMERO DE PALAVRAS, O NÚMERO DE CARACTERES, O NÚMERO DE OCORRÊNCIAS DE UM CARACTERE INDICADO PELO UTILIZADOR EXISTENTES NUM DETERMINADO FICHEIRO DE TEXTO INDICADO PELO UTILIZADOR.	5
2.3	ALTERE OS SUBPROGRAMAS ANTERIORES DE FORMA QUE SEJA ESCRITO, NO FINAL DO FICHEIRO, A INFORMAÇÃO DE CADA UMA DAS ALÍNEAS, ASSIM COMO A HORA EM QUE FORAM REGISTADAS. NOTA: PROCURE NA BIBLIOTECA <TIME.H> AS FUNÇÕES TIME(...), CTIME(...).	7
2.4	ELABORE OS SUBPROGRAMAS QUE CONSIDERAR NECESSÁRIOS PARA CRIAR FICHEIROS DE N NÚMEROS NATURAIS (100, 500, 1000) ORDENADOS POR ORDEM CRESCENTE, POR ORDEM DECRESCENTE E ALEATORIAMENTE.	9
2.5	DESENVOLVA UM SUBPROGRAMA QUE ESCREVA NUM FICHEIRO OS PRIMEIROS 50 NÚMEROS DO FICHEIRO ALEAT100.TXT, OS 100 PRIMEIROS NÚMEROS DO FICHEIRO ALEAT500.TXT E OS PRIMEIROS 500 NÚMEROS DO FICHEIRO ALEAT1000.TXT. ANALISE O FICHEIRO OBTIDO E SEPARA EM DOIS NOVOS FICHEIROS OS NÚMEROS PARES (EM PARES.TXT) DOS NÚMEROS ÍMPARES (EM IMPARES.TXT). AVERIGUE SE HÁ MAIS NÚMEROS PARES OU NÚMEROS ÍMPARES NO PRIMEIRO FICHEIRO CRIADO.	11
2.6	DESENVOLVA UM PROGRAMA QUE FAÇA A GESTÃO DOS ALUNOS INSCRITOS NUMA TURMA. PARA CADA ALUNO DEVE CONHECER-SE O NOME E O NÚMERO.	14
2.7	PARA AVERIGUAR AS DIFERENÇAS ENTRE FICHEIROS DE TEXTO E FICHEIROS BINÁRIOS, CRIE SUBPROGRAMAS PARA CADA UMA DAS SEGUINTE ALÍNEAS. TODAS AS ALÍNEAS DEVEM SER TESTADAS NO MAIN	18
2.8	REFAÇA O EXERCÍCIO 6: USANDO FICHEIROS BINÁRIOS, DESENVOLVA UM PROGRAMA QUE FAÇA A GESTÃO DOS ALUNOS INSCRITOS NUMA TURMA. PARA CADA ALUNO DEVE CONHECER-SE O NOME E O NÚMERO	24
2.9	PRETENDE-SE GUARDAR NUM FICHEIRO BINÁRIO CHAMADO TEMPERATURAS.DAT OS DADOS RECOLHIDOS POR UM TERMÓMETRO DIGITAL AUTOMÁTICO QUE REGISTA A TEMPERATURA DE 15 EM 15 MINUTOS	25
3	CONCLUSÃO	30
4	REFERÊNCIAS	31
5	ANEXOS	ERRO! MARCADOR NÃO DEFINIDO.

1 Introdução

Com este trabalho pretende-se criar programas que guardem um determinado conteúdo, introduzido pelo utilizador, em ficheiros texto. Também pretende-se conseguir ler de um ficheiro texto um determinado conteúdo seja ele já existente ou introduzido pelo o utilizador. Outro objetivo deste trabalho é conseguir guardar um determinado conteúdo num ficheiro binário e ler desse ficheiro esse conteúdo.

2 Trabalho realizado

- 2.1 Escreva três subprogramas que executem as seguintes tarefas:
- criar um ficheiro com o nome e conteúdo indicados pelo utilizador;
 - acrescentar informação, fornecida por um utilizador, a um ficheiro;
 - escrever na consola o conteúdo de um ficheiro.

Nome do Ficheiro: exe1_ED.c

Neste primeiro exercício foi-nos pedido para fazer três tarefas. A primeira é criar um ficheiro com o nome e o conteúdo introduzido pelo utilizador. A segunda tarefa é para o utilizador conseguir acrescentar informação ao ficheiro, para isso, abríamos o ficheiro no modo de abertura "a". Este modo de abertura deixa o utilizador acrescentar informação e se o ficheiro não existir cria-o e se existir coloca o ponteiro no final e deste modo não apaga o conteúdo do ficheiro. A última tarefa é escrever na consola o conteúdo do ficheiro. Para isso o main tem que ter dois argumentos: o primeiro argumento é o argc e o segundo é o argv. O argumento argc indica o número de argumentos da linha de comando, já o segundo argumento argv é um array de ponteiros de caracteres que lista todos os argumentos. O programa deste exercício é o seguinte:

```
#include <stdio.h>
int main(int argc , char * argv[])
{
    FILE * fp;
    char ficheiro[30];
    char nome[50];
    printf("Entre com o nome do ficheiro: ");
    scanf("%s", ficheiro);
    fp = fopen(ficheiro, "a");
    printf("Introduza uma frase: ");
    fflush(stdin);
    fgets(nome, 50, stdin);

    if (fp == NULL)
    {
        printf("Erro ao abrir o ficheiro\n");
    }
    else
    {
        fprintf(fp, "%s", nome);
    }
    fclose(fp);
}
```

Figura 1- Programa para escrever conteúdos num ficheiro

Output do exercício 1 no terminal do Atom:

```
Ca: exe1_ED
Entre com o nome do ficheiro: teste1
Introduza uma frase: Hoje e dia de ferias
Press any key to continue . . .
```

Figura 2- Criação do ficheiro teste1 e escrever conteúdo nesse ficheiro

Output deste exercício na linha de comandos do Windows:

```
C:\Users\olive\Desktop\LEI- 2ºSemestre\ED_exercícios2>gcc -o exe1_ED exe1_ED.c
C:\Users\olive\Desktop\LEI- 2ºSemestre\ED_exercícios2>exe1_ED
Entre com o nome do ficheiro: teste1
Introduza uma frase: Amanha vou as compras
```

Figura 3- Acrescentar conteúdo ao ficheiro teste1

2.2 Elabore subprogramas que contabilizem, o número de linhas, o número de palavras, o número de caracteres, o número de ocorrências de um caractere indicado pelo utilizador existentes num determinado ficheiro de texto indicado pelo utilizador.

Nome do Ficheiro: exe2_ED.c

Para este exercício fizemos todo num único programa. Este exercício consistia em pedir ao utilizador o nome do ficheiro para isso, abrimos o ficheiro no modo de leitura(r), pois neste modo o programa não vai apagar o ficheiro, mas vai apontar o ponteiro para o início do ficheiro e depois o que o programa faz é contar o número de linhas, palavras, caracteres e o número de ocorrências de um determinado caractere introduzido pelo o utilizador. Para contar o número de linhas o que o programa faz é percorrer o conteúdo do ficheiro e quando for igual ao “\n” incrementa uma variável contadora que é para contar o número de linhas. Já para contar as palavras, sempre que for igual a um espaço vazio ou igual ao “\n” é uma palavra, no entanto para contar o número de caracteres é o contrário de contar as palavras, ou seja, sempre que for diferente de um espaço vazio ou diferente do “\n” é um caractere. Por último, para contar o número de ocorrências de um determinado caractere pedimos ao utilizador que insira o caractere que quer procurar e depois o que o programa faz é ver se o caractere que o utilizador inseriu é igual aos caracteres que estão no ficheiro, sempre que for igual incrementa uma variável contador que é para saber o número de ocorrências desse caractere.

O programa deste exercício é o seguinte:

```
#include <stdio.h>

int main()
{
    FILE * fp;
    char ficheiro[30];
    char nome[50], caractere;
    int contador=0;
    int contador_pal = 0, contador_cara = 0, contador_caractere = 0;
    printf("Entre com o nome do ficheiro: ");
    scanf("%s", ficheiro);
    fp = fopen(ficheiro, "r");

    if (fp == NULL)
    {
        printf("Erro ao abrir o ficheiro\n");
    }
    else
    {
        printf("Introduza o caractere que quer procurar: ");
        fflush(stdin);
        scanf("%c",&caractere );
        while ((nome[50] = fgetc(fp)) != EOF)
        {
            if (nome[50] == '\n')
            {
                contador++;
            }
        }
    }
}
```

Figura 4- Programa para contar o número de linhas, palavras, caracteres e de um determinado caractere introduzido pelo o utilizador

```
if (nome[50] == ' ' || nome[50] == '\n')
{
    contador_pal++;
}
if (nome[50] != ' ' && nome[50] != '\n')
{
    contador_cara++;
}
if(caractere == nome[50])
    contador_caractere++;
}
printf("numero de linhas: %d", contador);
printf("\n");
printf("numero de palavras: %d", contador_pal);
printf("\n");
printf("numero de caracteres: %d", contador_cara);
printf("\n");
printf("Numero de caracteres encontrados: %d", contador_caractere);
fclose(fp);
}
}
```

Figura 5- Programa para contar o número de linhas, palavras, caracteres e de um determinado caractere introduzido pelo o utilizador

Exemplo de output deste exercício:

exe2_ED

```
Entre com o nome do ficheiro: teste1
Introduza o caractere que quer procurar: a
numero de linhas: 3
numero de palavras: 12
numero de caracteres: 48
Numero de caracteres encontrados: 6
Press any key to continue . . .
```

Figura 6- Output do exercício 2

2.3 Altere os subprogramas anteriores de forma que seja escrito, no final do ficheiro, a informação de cada uma das alíneas, assim como a hora em que foram registadas. **NOTA:** procure na biblioteca <time.h> as funções time(...), ctime(...).

Nome do Ficheiro: exe3_ED.c

Neste exercício, é pedido para acrescentarmos aos ficheiros dos exercícios anteriores a informação de cada uma das alíneas e a hora em que foram registados. Para isso, utilizamos a variável time_t que serve para guardar a hora. Já para guardar a data utilizamos a função time e para apresentar a data e a hora utilizamos a função ctime.

Para utilizar estas funções é preciso a biblioteca time.h. O programa é o seguinte:

```
1 #include <stdio.h>
2 #include <time.h>
3 int main()
4 {
5     FILE * fp;
6     char ficheiro[30];
7     char nome[50], caractere;
8     time_t dataNewLine;
9     time_t dataCarac;
10    time_t dataCarac;
11    time_t dataPalavras;
12    int contador=0;
13    int contador_pal = 0, contador_cara = 0, contador_caractere = 0;
14    printf("Entre com o nome do ficheiro: ");
15    scanf("%s", ficheiro);
16    fp = fopen(ficheiro, "r");
17
18    if (fp == NULL)
19    {
20        printf("Erro ao abrir o ficheiro\n");
21    }
22    else
23    {
24        printf("Introduza o caractere que quer procurar: ");
25        fflush(stdin);
26        scanf("%c",&caractere );
27        while ((nome[50] = fgetc(fp)) != EOF)
28        {
29            if (nome[50] == '\n')
30            {
```

Figura 7- Programa para apresentar a hora e data de cada alínea

```
31     contador++;
32     time(&dataNewLine);
33 }
34 if (nome[50] == ' ' || nome[50] == '\n')
35 {
36     contador_pal++;
37     time(&dataPalavras);
38 }
39 if (nome[50] != ' ' && nome[50] != '\n')
40 {
41     contador_cara++;
42     time(&dataCara);
43 }
44 if(caractere == nome[50])
45     contador_caractere++;
46     time(&dataCarac);
47 }
48 printf("numero de linhas: %d", contador);
49 printf("\nAltura: %s", ctime(&dataNewLine));
50 printf("\n");
51 printf("numero de palavras: %d", contador_pal);
52 printf("\nAltura: %s", ctime(&dataPalavras));
53 printf("\n");
54 printf("numero de caracteres: %d", contador_cara);
55 printf("\nAltura: %s", ctime(&dataCarac));
56 printf("\n");
57 printf("Numero de caracteres encontrados: %d", contador_caractere);
58 printf("\nAltura: %s", ctime(&dataCara));
59 }
60 fclose(fp);
```

Figura 8- Programa para apresentar a hora e data de cada alínea

O output deste exercício é o seguinte:

```
C:\ ex3bED
Entre com o nome do ficheiro: teste1
Introduza o caractere que quer procurar: a
numero de linhas: 3
Altura: Tue Apr 07 21:18:44 2020

numero de palavras: 12
Altura: Tue Apr 07 21:18:44 2020

numero de caracteres: 48
Altura: Tue Apr 07 21:18:44 2020

Numero de caracteres encontrados: 6
Altura: Tue Apr 07 21:18:44 2020
```

Figura 9- Output do exercício 3

2.4 Elabore os subprogramas que considerar necessários para criar ficheiros de n números naturais (100, 500, 1000) ordenados por ordem crescente, por ordem decrescente e aleatoriamente.

Nome do Ficheiro: exe4RandomED.c

Neste exercício a professora pediu-nos para criar ficheiros que estivessem ordenados por ordem decrescente, crescente e aleatoriamente. O nome dos ficheiros tem de indicar os números de elementos e o modo como estão ordenados, para isso o que fizemos foi pedir ao utilizador que introduzisse a quantidade de números (100, 500, 1000) que quer ter nos ficheiros e depois dentro das funções esse número e o nome do ficheiro, que é diferente para cada função e já é indicado em cada função, vão ser concatenados através da função `snprintf` para indicar o nome de cada ficheiro, isto é, o ficheiro `cresc500.txt` contém os números entre 1 a 500 por ordem crescente. A primeira função é para ordenar os números por ordem crescente, a segunda função é a da ordenar os números por ordem decrescente e a última função é para ordenar por ordem aleatória. Na última função para introduzir os números por ordem aleatória utilizamos a função `rand` que vai escrever no ficheiro os números por ordem aleatória entre 1 e a variável quantidade (variável referente ao número de elementos que o utilizador quer introduzir), por exemplo, se o utilizador quiser inserir 100 números o que esta função vai fazer é escrever 100 números entre 1 a 100. As funções são as seguintes:

```
int Cresc(int quantidade)
{
    FILE * fp;
    char ficheiro[100]="cresc";
    snprintf(ficheiro, sizeof ficheiro, "%s%d", ficheiro, quantidade);
    strcat(ficheiro, ".txt");
    fp = fopen(ficheiro, "w");
    if (fp == NULL)
    {
        printf("Erro ao abrir o ficheiro\n");
    }
    else
    {
        for(int i=1;i<quantidade+1;i++)
            fprintf(fp, "%d ", i);
        fclose(fp);
    }
}
```

Figura 10- Função para escrever no ficheiro os números por ordem crescente

```
int Decrescente(int quantidade)
{
    FILE * fp;
    char ficheiro[100]="decr";
    snprintf(ficheiro, sizeof ficheiro, "%s%d", ficheiro, quantidade);
    strcat(ficheiro, ".txt");
    fp = fopen(ficheiro, "w");
    if (fp == NULL)
    {
        printf("Erro ao abrir o ficheiro\n");
    }
    else
    {
        for(int i=quantidade ;i>0; i--)
            fprintf(fp, "%d ", i);
    }
    fclose(fp);
}
```

Figura 12- Função para escrever no ficheiro os números por ordem decrescente

```
int aleat(int quantidade)
{
    FILE * fp;
    int c;
    char ficheiro[100]="aleat";
    snprintf(ficheiro, sizeof ficheiro, "%s%d.txt", ficheiro, quantidade);
    fp = fopen(ficheiro, "w+");
    if (fp == NULL)
    {
        printf("Erro ao abrir o ficheiro\n");
    }
    else
    {
        for(int i=1; i<quantidade; i++)
        {
            fprintf(fp, "%d ", rand() % quantidade+1);
        }
    }
    fclose(fp);
}
```

Figura 11- Função para escrever no ficheiro os números por ordem aleatória

Os ficheiros são os seguintes:

aleat100.txt

1	42	68	35	1	70	25	79	59	63	65	6	46	82	28	62	92	96	43	28	37	92	5	3	54	93	83	22	17	19	96	48	27	72	39	70	13	68	100	36	95	4	12	23	34	74	65
---	----	----	----	---	----	----	----	----	----	----	---	----	----	----	----	----	----	----	----	----	----	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----	----	----	---	----	----	----	----	----

Figura 13- Ficheiro onde estão os números por ordem aleatória

cresc100.bt																																																
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48

Figura 14- Ficheiro onde estão os números por ordem crescente

decr100.txt																																													
1	100	99	98	97	96	95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64	63	62	61	60	59	58	57	

Figura 15- Ficheiro onde estão os números por ordem decrescente

2.5 Desenvolva um subprograma que escreva num ficheiro os primeiros 50 números do ficheiro *aleat100.txt*, os 100 primeiros números do ficheiro *aleat500.txt* e os primeiros 500 números do ficheiro *aleat1000.txt*. Analise o ficheiro obtido e separe em dois novos ficheiros os números pares (em *pares.txt*) dos números ímpares (em *impares.txt*). Averigue se há mais números pares ou números ímpares no primeiro ficheiro criado.

Nome do Ficheiro: exe5_ED.c

Para fazer este exercício criámos 2 funções, sendo a primeira esta:

```
void MostrarNumeros()
{
    int primeirosNumeros[1000];
    FILE * aleat50;
    FILE * aleat100;
    FILE * aleat500;
    FILE * PrimeirosNumeros;

    aleat50=fopen("aleat100.txt","r");
    aleat100=fopen("aleat500.txt","r");
    aleat500=fopen("aleat1000.txt","r");
    PrimeirosNumeros=fopen("PrimeirosNumeros.txt","w+");
}
```

Figura 16- Ficheiros onde vão ser escritos os números dos ficheiros do exercício 4

Aqui, criámos 3 ficheiros com conteúdos dos ficheiros *aleat* do exercício anterior. O 4º ficheiro, vai servir para armazenar os valores dos outros 3, que vai também ser útil para separar em 2 ficheiros os números pares e os números ímpares. Para armazenarmos no ficheiro “PrimeirosNumeros.txt”, fizemos o seguinte:

```
for(int i=0;i<50;i++)
{
    fscanf(aleat50,"%d",&primeirosnumeross[i]);
}
for(int i=0;i<50;i++)
{
    fprintf(PrimeirosNumeros,"%d ",primeirosnumeross[i]);
}

for(int i=0;i<100;i++)
{
    fscanf(aleat100,"%d",&primeirosnumeross[i]);
}
for(int i=0;i<100;i++)
{
    fprintf(PrimeirosNumeros,"%d ",primeirosnumeross[i]);
}

for(int i=0;i<500;i++)
{
    fscanf(aleat500,"%d",&primeirosnumeross[i]);
}

for(int i=0;i<500;i++)
{
    fprintf(PrimeirosNumeros,"%d ",primeirosnumeross[i]);
}
```

Figura 17- Ciclos para guardar os números no ficheiro PrimeirosNumeros.txt

O que fizemos aqui foi simplesmente guardar dentro de um vetor o conteúdo dos 3 ficheiros de acordo com o que a professora pretendia. Depois de guardarmos no vetor, simplesmente colocámos o que estava dentro do vetor, no ficheiro “PrimeirosNumeros.txt”.

Para separarmos o conteúdo do ficheiro em 2 ficheiros, de números pares e números ímpares, fizemos a seguinte função:

```
void ParesImpares()
{
    int Numeros[1000];
    FILE * PrimeirosNumeros;
    FILE * NumerosPares;
    FILE * NumerosImpares;

    PrimeirosNumeros=fopen("PrimeirosNumeros.txt", "r");
    NumerosPares=fopen("Pares.txt", "w+");
    NumerosImpares=fopen("Impares.txt", "w+");
    for(int i=0;i<650;i++)
    {
        fscanf(PrimeirosNumeros, "%d", &Numeros[i]);
    }

    for(int i=0;i<650;i++)
    {
        if(Numeros[i]%2==0)
        {
            fprintf(NumerosPares, "%d ", Numeros[i]);
        }
        else
            fprintf(NumerosImpares, "%d ", Numeros[i]);
    }
}
```

Figura 18- Função para escrever nos ficheiros os números pares e ímpares

O que fizemos nesta função foi simplesmente colocar os números do ficheiro “PrimeirosNumeros.txt” num vetor e de seguida, um ciclo que verificasse os números do vetor. Caso sejam pares, vão para o ficheiro “Pares.txt”, caso contrário, caso sejam ímpares, vão para o ficheiro “Ímpares.txt”.

2.6 Desenvolva um programa que faça a gestão dos alunos inscritos numa turma. Para cada aluno deve conhecer-se o nome e o número.

Nome do Ficheiro: exe6_ED.c

Neste exercício, foi-nos pretendido pela professora criar um programa que faça a gestão de alunos inscritos numa turma. O programa tem que permitir listar alunos, remover alunos, inscrever alunos e para cada execução apresentar os dados dos alunos inscritos. Para isso fizemos um menu de opções pois torna-se mais fácil para o utilizador usar e para nós. O menu de opções é o seguinte:

```
printf("Menu do Programa\n");
printf("1. Inserir Alunos\n");
printf("2. Remover Alunos\n");
printf("3. Apresentar Alunos\n");
printf("4. Gravar Dados dos Alunos\n");
printf("0. Sair\n");
printf("Entre com a opcao: ");
scanf("%d", &opcao);
fflush(stdin);

switch(opcao)
{
    case 1: total_alunos = inserir_alunos(ids_aluno, nomes, total_alunos); break;
    case 2: total_alunos = remover_varios_alunos(ids_aluno, nomes, total_alunos); break;
    case 3: apresentar_dados(ids_aluno, nomes, total_alunos); break;
    case 4: grava_ficheiros(nomes, ids_aluno, total_alunos); break;
    case 0: printf("opcao Sair selecionada\n"); break;
    default: printf("opcao invalida\n");
}
getchar();
}
```

Figura 19- Menu de opções do exercício 6

A primeira opção do programa é de inserir alunos. Esta função permite que o utilizador consiga inserir os dados dos alunos, isto é, nome e número de aluno. A função é a seguinte:

```
int inserir_alunos(char ids_aluno[][MAX_ID], char nomes[][MAX_NOME], int total_alunos)
{
    int numero_alunos;
    printf("Introduza o numero de alunos: ");
    scanf("%d", &numero_alunos);

    for (int i = total_alunos; (i < total_alunos + numero_alunos) && (i < MAX_ALUNOS); i++)
    {
        char nome[50];
        char id_aluno[11];
        printf("Introduza o nome do aluno: ");
        fflush(stdin);
        fgets(nome, 50, stdin);
        printf("numero de aluno: ");
        fflush(stdin);
        scanf("%s", id_aluno);

        strcpy(nomes[i], nome);
        strcpy(ids_aluno[i], id_aluno);
        printf("\n");
    }
    if (total_alunos + numero_alunos > MAX_ALUNOS)
    {
        printf("Nao e possivel inserir porque ultrapassou o numero maximo de alunos\n");
        return MAX_ALUNOS;
    }
    else
        return total_alunos + numero_alunos;
}
```

Figura 20- Função para inserir alunos

Output da função de inserir alunos:

```
Menu do Programa
1. Inserir Alunos
2. Remover Alunos
3. Apresentar Alunos
4. Gravar Dados dos Alunos
0. Sair
Entre com a opcao: 1
Introduza o numero de alunos: 2
Introduza o nome do aluno: Antonio Pereira
numero de aluno: 2019141051

Introduza o nome do aluno: David Ferreira
numero de aluno: 2019141281
```

Figura 21- Interface da função inserir_alunos

A segunda opção do nosso programa é a de remover alunos. Nesta função o utilizador tem que inserir o número do aluno que quer remover e o programa vai comparar o número que o utilizador inseriu com os números que se encontram na string `ids_alunos` se for igual essa posição vai ficar com os dados dos alunos seguintes. A função de puxar os elementos atrás e a função de remover alunos são as seguintes:

```
void puxar_alunos_atras(char ids_aluno[][MAX_ID], char nomes[][MAX_NOME], int total_alunos, int posicao_apagar)
{
    for (int i = posicao_apagar; i < total_alunos; i++)
    {
        strcpy(ids_aluno[i], ids_aluno[i+1]);
        strcpy(nomes[i], nomes[i+1]);
    }
}
```

Figura 22- Função para puxar os dados dos alunos uma posição atrás

```
int remover_aluno(char ids_aluno[][MAX_ID], char nomes[][MAX_NOME], int total_alunos)
{
    char numero_pesquisar[MAX_ID];
    printf("Introduza o numero de aluno que quer remover: ");
    scanf("%s", numero_pesquisar);
    int posicao_apagar, apagou=0;
    for (int i = 0; i < total_alunos; i++)
    {
        if (strcmp(ids_aluno[i], numero_pesquisar) == 0)
        {
            apagou = 1;
            posicao_apagar = i;
            break;
        }
    }
    if (apagou == 1)
    {
        puxar_alunos_atras(ids_aluno, nomes, total_alunos, posicao_apagar);
        printf("Aluno Removido com sucesso\n");
        return total_alunos - 1;
    }
    else
    {
        printf("Aluno nao encontrado\n");
        return total_alunos;
    }
}
```

Figura 23- Função para remover alunos

Para o utilizador conseguir remover mais que um aluno fizemos uma função que pede ao utilizador que insira quantos alunos quer remover. É realizado um for que vai até ao número de alunos que o utilizador quer remover e dentro deste for é chamada a função que remove os alunos. Esta função é a que é chamada no menu de opções. A função para remover mais que um aluno é a seguinte:

```
int remover_varios_alunos (char ids_aluno[][MAX_ID], char nomes[][MAX_NOME],int total_alunos)
{
    int n_alunos_remove;
    printf("Introduza quantos alunos pretende remover: ");
    scanf("%d", &n_alunos_remove);
    for (int i = 0; i < n_alunos_remove; i++){
        total_alunos = remover_aluno( ids_aluno,nomes,total_alunos);
    }
    return total_alunos;
}
```

Figura 24- Função para remover mais que um aluno

Exemplo de output da função de remover alunos:

```
Entre com a opcao: 2
Introduza quantos alunos pretende remover: 1
Introduza o numero de aluno que quer remover: 2019141051
Aluno Removido com sucesso
```

Figura 25- Interface da função remover_varios_alunos

A terceira opção do menu é a de apresentar alunos. Esta função o que faz é imprimir o nome e número de aluno. Para isto fizemos um for a percorrer a lista dos alunos e dentro deste for são realizados dois printf's um para o nome dos alunos e outro para os números dos alunos. A função é a seguinte:

```
void apresentar_dados(char ids_aluno[][MAX_ID], char nomes[][MAX_NOME], int total_alunos)
{
    for(int i = 0; i < total_alunos; i++)
    {
        printf("Nome: %s", nomes[i]);
        printf("Numero: %s", ids_aluno[i]);
        printf("\n");
    }
}
```

Figura 26- Função para apresentar alunos

O output desta função é o seguinte:

```
Entre com a opcao: 3
Nome: David Ferreira
Numero: 2019141281
```

Figura 27- Interface da função apresentar_dados

A última opção do nosso menu é para gravar os dados dos alunos. Nesta função o ficheiro é aberto no modo de escrita(w) e o que a função faz é escrever no ficheiro de texto o nome e número de alunos através de dois fputs. Para o programa conseguir ler os dados dos alunos do ficheiro fizemos uma função para ler os dados, para isso, abríamos o ficheiro anterior no modo de leitura. Depois disto, é realizado um while que percorre todo o ficheiro e dentro deste while é realizado um fgets, que serve para ler os nomes dos alunos, e a leitura dos números é feita no while através também de um fgets. As funções de gravar e ler os dados dos alunos são as seguintes:

```
void grava_ficheiros(char nomes[][MAX_NOME], char ids_aluno[][MAX_ID],int total_alunos)
{
    FILE *f=fopen("alunos.txt","w");
    for (int i=0; i < total_alunos; i++)
    {
        fputs(ids_aluno[i], f);
        fputs("\n", f);
        fputs(nomes[i], f);
    }
    fclose(f);
}
```

Figura 28- Função para gravar os dados dos alunos

```
int ler_ficheiro_alunos(char nomes[][MAX_NOME], char ids_aluno[][MAX_ID])
{
    FILE *f=fopen("alunos.txt","r");
    int i=0;
    while (fgets(ids_aluno[i], MAX_ID, f) != NULL)
    {
        if(ids_aluno[i][strlen(ids_aluno[i])-1]!='\0')
            ids_aluno[i][strlen(ids_aluno[i])-1]='\0';
        fgets(nomes[i], MAX_NOME, f);
        i++;
    }

    fclose(f);
    return i;
}
```

Figura 29- Função para ler os dados dos alunos

- 2.7 Para averiguar as diferenças entre ficheiros de texto e ficheiros binários, crie subprogramas para cada uma das seguintes alíneas. Todas as alíneas devem ser testadas no main

Nome do Ficheiro: exe7_ED.c

Neste exercício, foi-nos pretendido pela professora verificar as diferenças entre os ficheiros de texto e ficheiros binários. Para isso, a professora pediu que inseríssemos os nomes dos alunos e guardar esses nomes tanto em ficheiros de texto como em ficheiros binários e ler os nomes dos ficheiros de texto e dos ficheiros binários. Deste modo, para se tornar mais simples e fácil de compreender para nós e para o utilizador decidimos fazer um menu de opções como fizemos para o exercício anterior. O menu é o seguinte:

```
printf("Menu do Programa\n");
printf("1. Inserir Nomes\n");
printf("2. Gravar Nomes\n");
printf("3. Apresentar Nomes\n");
printf("4. Guardar um nome de cada vez em binario\n");
printf("5. Ler tabela de nomes em binario\n");
printf("6. Guradar tabela de nomes de uma so vez\n");
printf("0. Sair\n");
printf("Entre com a opcao: ");
scanf("%d", &opcao);
fflush(stdin);
```

Figura 30- Menu de opções do exercício 7

Para a alínea a) utilizamos parte da função de inserir alunos do exercício anterior, no entanto, o que fizemos de diferente foi pedir ao utilizador que inserisse os nomes dos

```
int inserir_nomes(char nomes[][MAX_NOME], int total_alunos)
{
    int numero_alunos;
    printf("Opcao Inserir Alunos selecionada\n");
    printf("Introduza o numero de alunos: ");
    scanf("%d", &numero_alunos);

    for (int i = total_alunos; i < total_alunos + numero_alunos; i++)
    {
        char nome[50];
        printf("Introduza o nome do aluno: ");
        fflush(stdin);
        fgets(nome, 50, stdin);

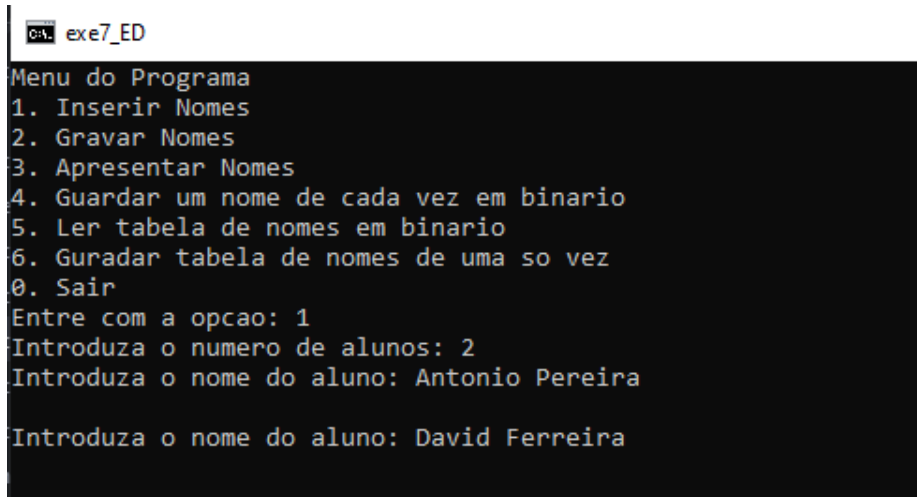
        strcpy(nomes[i], nome);
        printf("\n");
    }

    return total_alunos + numero_alunos;
```

Figura 31- Função para inserir os nomes dos alunos

alunos pois nesta alínea não nos é pedido para o utilizador inserir o número do aluno. A função é a seguinte:

O output desta função é o seguinte:



```

C:\> exe7_ED
Menu do Programa
1. Inserir Nomes
2. Gravar Nomes
3. Apresentar Nomes
4. Guardar um nome de cada vez em binario
5. Ler tabela de nomes em binario
6. Guradar tabela de nomes de uma so vez
0. Sair
Entre com a opcao: 1
Introduza o numero de alunos: 2
Introduza o nome do aluno: Antonio Pereira

Introduza o nome do aluno: David Ferreira
  
```

Figura 32- Interface da função inserir_nomes

Para a alínea b), a função é a mesma do exercício anterior de guardar os alunos, só que aqui utilizamos um único fputs que é para gravar os nomes dos alunos. A função de ler os nomes dos alunos, alínea c), também é a mesma do exercício anterior, mas tal como fizemos na função de gravar os nomes só utilizamos um fgets que é também para ler os nomes. As funções são as seguintes:

```

void gravar_nomes(char nomes[][MAX_NOME],int total_alunos)
{
    FILE *f=fopen("nomes.txt","w");
    for (int i=0; i < total_alunos; i++)
    {
        fputs(nomes[i], f);
        fputs("\n", f);
    }
    fclose(f);
    printf("Nomes guardados com sucesso");
}
  
```

Figura 33- Função para gravar os nomes dos alunos

```
int ler_ficheiro_nomes(char nomes[][MAX_NOME], int total_alunos)
{
    FILE *f=fopen("nomes.txt","r");
    int i = 0;
    while(fgets(nomes[i], MAX_NOME, f) != NULL)
    {
        if(nomes[i][strlen(nomes[i])-1]!='\0')
            nomes[i][strlen(nomes[i])-1]='\0';
        i++;
    }
    fclose(f);
    return i;
}
```

Figura 34- Função para ler os nomes dos alunos

O output da função de gravar nomes é o seguinte:

```
Menu do Programa
1. Inserir Nomes
2. Gravar Nomes
3. Apresentar Nomes
4. Guardar um nome de cada vez em binario
5. Ler tabela de nomes em binario
6. Guardar tabela de nomes de uma so vez
0. Sair
Entre com a opcao: 2
Nomes guardados com sucesso
```

Figura 35- Interface da função gravar_nomes

Para alínea d) foi-nos pedido para guardar os nomes dos alunos em dois ficheiros binários diferentes e de duas formas diferentes. Num tínhamos que guardar os nomes de uma só vez e no outro guardar um nome de cada vez. Para a primeira forma, guardar de uma só vez, usamos uma função e dentro desta função abríamos o ficheiro no modo de abertura de escrita(wb) e fizemos um único fwrite e dentro deste no número de elementos a escrever multiplicamos pela variável total_alunos, que é a variável que diz os números de alunos que foram inseridos. Já para a segunda forma, guardar um nome de cada vez, também fizemos uma função e o modo de abertura também é o mesmo, só que dentro desta função fizemos um for a percorrer a lista dos alunos e dentro deste for fizemos um fwrite e no número de elementos a escrever pusemos 1 e assim vai guardar um nome de cada vez. As funções são as seguintes:

```
void guardar_tabelanomes_bin(char nomes[][MAX_NOME],int total_alunos)
{
    FILE *fp=fopen("tabelaNomes.dat","wb");
    int i = 0;
    if( fp == NULL)
    {
        printf("Impossivel abrir o ficheiro");
    }
    else
    {
        {
            fwrite(nomes[i], sizeof(nomes[i]),1*total_alunos, fp);
            i++;
        }
        fclose(fp);
        printf("Nomes guardados com sucesso");
    }
}
```

Figura 36- Função para gravar os nomes de uma só vez

```
void guardar_nomes_bin(char nomes[][MAX_NOME],int total_alunos)
{
    FILE *fp=fopen("nomesBinario.dat","wb");
    if( fp == NULL)
    {
        printf("Impossivel abrir o ficheiro");
    }
    else
    {
        {
            for(int i = 0; i < total_alunos; i++)
                fwrite(nomes[i], sizeof(nomes[i]),1, fp);
        }
        fclose(fp);
        printf("Nomes guardados com sucesso");
    }
}
```

Figura 37- Função para gravar um nome de cada vez

Na última alínea é pedido para lermos os nomes dos alunos dos ficheiros da alínea anterior. Para isso, fizemos duas funções e dentro destas abríamos os ficheiros no modo de abertura de ler em binário(rb) e fizemos um while. No while é realizado um fread, que serve para ler os nomes dos ficheiros binários, para as duas funções. Para apresentar os nomes que estão guardados nos ficheiros binários, fizemos um for, a percorre a variável i, que é a que diz quantos nomes é que foram lidos, e dentro deste for é realizado um printf dos nomes. As funções desta última alínea são as seguintes:

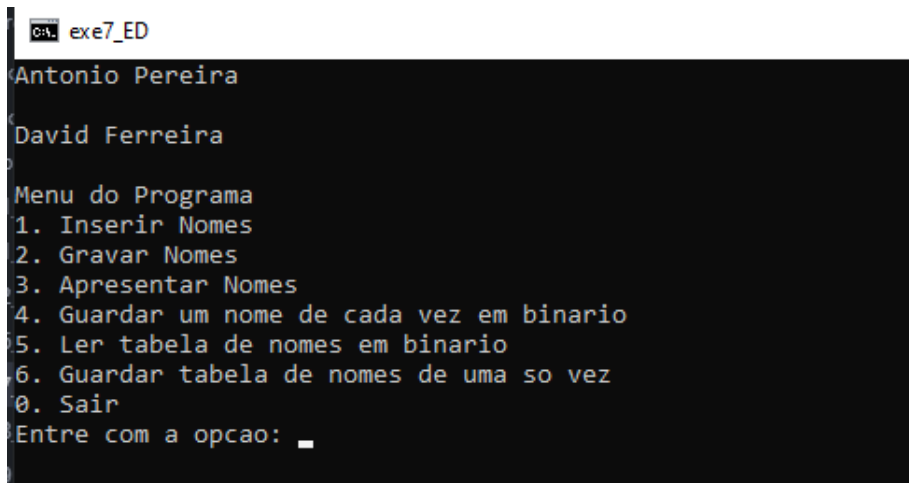
```
void ler_tabelanomes_bin(char nomes[][MAX_NOME],int total_alunos)
{
    FILE *fp=fopen("tabelaNomes.dat","rb");
    int i = 0;
    if( fp == NULL)
    {
        printf("Impossivel abrir o ficheiro");
    }
    else
    {
        while(fread(nomes[i], sizeof(nomes[i]),1, fp)!=0)
        {
            i++;
        }
        for (int j = 0; j < i; j++)
        {
            printf("%s", nomes[j]);
            printf("\n");
        }
    }
    fclose(fp);
}
```

Figura 38- Função para ler a tabela dos nomes

```
void ler_nomes_bin(char nomes[][MAX_NOME])
{
    FILE *fp=fopen("nomesBinario.dat","rb");
    int i = 0;
    if( fp == NULL)
    {
        printf("Impossivel abrir o ficheiro");
    }
    else
    {
        while(fread(nomes[i], sizeof(nomes[i]),1, fp) != 0)
        {
            i++;
        }
        for (int j = 0; j < i; j++)
        {
            printf("%s", nomes[j]);
            printf("\n");
        }
    }
    fclose(fp);
}
```

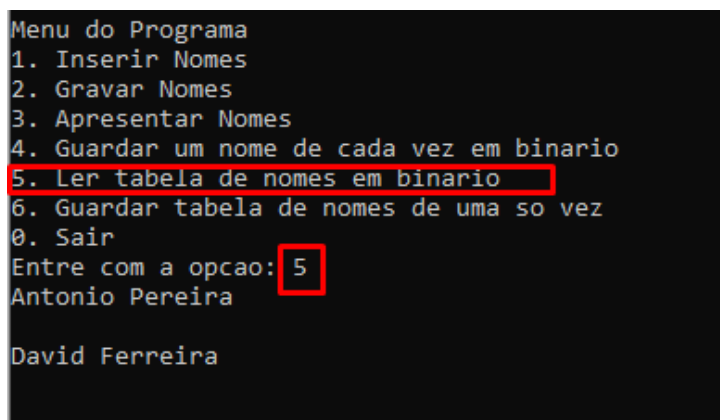
Figura 39- Função para ler os nomes de cada vez

Os outputs destas funções são os seguintes:



```
exe7_ED
Antonio Pereira
David Ferreira
Menu do Programa
1. Inserir Nomes
2. Gravar Nomes
3. Apresentar Nomes
4. Guardar um nome de cada vez em binario
5. Ler tabela de nomes em binario
6. Guardar tabela de nomes de uma so vez
0. Sair
Entre com a opcao: _
```

Figura 40- Interface da função ler_nomes_bin



```
Menu do Programa
1. Inserir Nomes
2. Gravar Nomes
3. Apresentar Nomes
4. Guardar um nome de cada vez em binario
5. Ler tabela de nomes em binario
6. Guardar tabela de nomes de uma so vez
0. Sair
Entre com a opcao: 5
Antonio Pereira
David Ferreira
```

Figura 41- Interface da função ler_tabelanomes_bin

2.8 Refaça o exercício 6: usando ficheiros binários, desenvolva um programa que faça a gestão dos alunos inscritos numa turma. Para cada aluno deve conhecer-se o nome e o número

Nome do Ficheiro: exe8_ED.c

Neste exercício é pedido para utilizarmos as funções do exercício 6, só que em vez de guardar os dados dos alunos num ficheiro texto temos que guardar num ficheiro binário e ler do ficheiro binário esses dados. Para isso, para gravar os dados dos alunos fizemos uma função diferente em que dentro desta abríamos o ficheiro binário no modo de escrita em binário(wb) e fizemos um for a percorrer a lista dos alunos. Dentro deste for fizemos dois fwrite um para os nomes e outro para os números. Já para ler os dados dos alunos fizemos um while em que esta a ler o número dos alunos e dentro deste é realizado um fread para ler os nomes dos alunos. O resto é igual ao exercício 6. As funções de gravar e ler são as seguintes:

```
int ler_ficheiro_alunos_bin(char nomes[][MAX_NOME], char ids_aluno[][MAX_ID])
{
    FILE *f=fopen("alunos.dat","rb");
    int i=0;
    while (fread(ids_aluno[i], sizeof(ids_aluno[i]),1, f) != 0)
    {
        fread(nomes[i], sizeof(nomes[i]),1, f);
        i++;
    }

    fclose(f);
    return i;
}
```

Figura 42- Função para gravar os dados dos alunos

```
void grava_ficheiros_bin(char nomes[][MAX_NOME], char ids_aluno[][MAX_ID],int total_alunos)
{
    FILE *f=fopen("alunos.dat","wb");
    if(f == NULL)
    {
        printf("Impossivel abrir o ficheiro");
    }
    else
    {
        for (int i=0; i < total_alunos; i++)
        {
            fwrite(ids_aluno[i], sizeof(ids_aluno[i]),1, f);
            fwrite(nomes[i], sizeof(nomes[i]),1, f);
        }
        fclose(f);
    }
    printf("Dados dos alunos guardados com sucesso\n");
}
```

Figura 43- Função para ler os dados dos alunos

2.9 Pretende-se guardar num ficheiro binário chamado *temperaturas.dat* os dados recolhidos por um termómetro digital automático que regista a temperatura de 15 em 15 minutos

Nome do Ficheiro: exe9_ED.c

Neste último exercício foi-nos pretendido pela professora fazer um programa que consiga inserir os dados introduzidos pelo utilizador como a data, a hora e a temperatura. O programa também tem que ser capaz de fazer a amplitude térmica e a media das temperaturas de um determinado dia. Por último o programa tem que gravar estes dados num ficheiro binário. Deste modo, fizemos um menu de opções a semelhança de alguns exercícios anteriores. O menu é o seguinte:

```
printf("Menu do Programa\n");
printf("1. Inserir Registos\n");
printf("2. Apresentar Registos\n");
printf("3. Calcular Amplitude Termica\n");
printf("4. Calcular Media de Temperaturas de um dia\n");
printf("5. Gravar dados recolhidos\n");
printf("0. Sair\n");
printf("Entre com a opcao: ");
scanf("%d", &opcao);
fflush(stdin);
```

A primeira opção do nosso menu é a de inserir os dados do termómetro digital. Nesta função é realizado um for a percorrer a lista de registos mais o número de registos que o utilizador que inserir e dentro deste for pedimos ao utilizador que insira a data, a hora e a temperatura. Depois esta função retorna o total de registos mais o número de registos que o utilizador inseriu. A função é a seguinte:

```
int inserir_registos(long long int data_registro[], float hora_registro[], float temperatura_registro[], int total_registos)
{
    int n_registos;
    printf("Introduza quantos registos pretende fazer: ");
    scanf("%d", &n_registos);
    for (int i = total_registos; i < (total_registos + n_registos) && (i < MAX_REGISTOS); i++)
    {
        printf("Introduza a data: ");
        scanf("%lld", &data_registro[i]);
        printf("Introduza a hora: ");
        scanf("%f", &hora_registro[i]);
        printf("Introduza a temperatura: ");
        scanf("%f", &temperatura_registro[i]);
    }
    return total_registos + n_registos;
}
```

Figura 44- Função para inserir registo do termómetro digital

Exemplo de output da função de inserir registos:

```
Menu do Programa
1. Inserir Registos
2. Apresentar Registos
3. Calcular Amplitude Termica
4. Calcular Media de Temperaturas de um dia
5. Gravar dados recolhidos
0. Sair
Entre com a opcao: 1
Introduza quantos registos pretende fazer: 2
Introduza a data: 20200112
Introduza a hora: 13.45
Introduza a temperatura: 15.0
Introduza a data: 20200112
Introduza a hora: 15.35
Introduza a temperatura: 19.0
```

Figura 45- Interface da função inserir_registos

A segunda opção do nosso programa é para calcular a amplitude térmica durante um dia. Nesta função pedimos ao utilizador que insira o dia para calcular a amplitude depois o programa vai ver qual é a maior e menor temperatura desse dia, quando encontrar vai subtrair a maior temperatura com a menor temperatura e retorna o valor da amplitude térmica. Esta função é chamada na função de calcular várias amplitudes térmicas. As funções são as seguintes:

```
float calcular_amtermica(long long int data_registro[], float temperatura_registro[], int total_registos)
{
    long long int data_procurar;
    float maior_temp = 0, menor_temp, amplitude_ter;
    int encontrou = 0;
    while(encontrou == 0)
    {
        printf("Introduza a data para calcular a amplitude termica: ");
        scanf("%lld", &data_procurar);
        for(int i = 0; i < total_registos; i++)
        {
            if (data_procurar == data_registro[i])
            {
                encontrou = 1;
                if( temperatura_registro[i] > maior_temp)
                    maior_temp = temperatura_registro[i];
            }
        }
        menor_temp = maior_temp;
        for(int i = 0; i < total_registos; i++)
        {
            if (data_procurar == data_registro[i])
            {
                if( temperatura_registro[i] < menor_temp)
                    menor_temp = temperatura_registro[i];
            }
        }
    }
    amplitude_ter = maior_temp - menor_temp;
    return amplitude_ter;
}
```

Figura 46- Função para calcular a amplitude térmica de um único dia

```
void calcular_varias_amptermicas(long long int data_registo[], float temperatura_registo[], int total_registos)
{
    int n_amptermica;
    printf("Introduza para quantos dias e que pretende calcular a amplitude termica: ");
    scanf("%d", &n_amptermica);
    for (int j = 0; j < n_amptermica; j++)
    {
        printf("Amplitude Termica: %0.1f\n", calcular_amptermica(data_registo, temperatura_registo, total_registos));
    }
}
```

Figura 47- Função para calcular a amplitude térmica de vários dias

O output da função de calcular a amplitude térmica de vários dias é o seguinte:

```
Menu do Programa
1. Inserir Registos
2. Apresentar Registos
3. Calcular Amplitude Termica
4. Calcular Media de Temperaturas de um dia
5. Gravar dados recolhidos
0. Sair
Entre com a opcao: 3
Introduza para quantos dias e que pretende calcular a amplitude termica: 1
Introduza a data para calcular a amplitude termica: 20200112
Amplitude Termica: 4.0
```

Figura 48- Interface da função calcular_varias_amptermicas

A opção quatro do nosso menu é a de calcular a media de temperaturas de um determinado dia. Para isso, pedimos ao utilizador que insira a data para calcular a media, se existir essa data o programa vai somar todas as temperaturas dessa data e esta soma vai ser dividida pelo número de temperaturas que tem nessa data para calcular assim a média. Esta função é chamada na função de calcular a media das temperaturas de vários dias. As duas funções são as seguintes:

```
float calcular_mediatemp(long long int data_registo[], float temperatura_registo[], int total_registos)
{
    int encontrou = 0;
    int m = 0;
    float soma = 0;
    long long int data_procurar;
    float media_temp;
    while(encontrou == 0)
    {
        printf("Introduza a data para calcular a amplitude termica: ");
        scanf("%lld", &data_procurar);
        for(int i = 0; i < total_registos; i++)
        {
            if (data_procurar == data_registo[i])
            {
                encontrou = 1;
                soma = soma + temperatura_registo[i];
                m++;
                media_temp = soma / m;
            }
        }
    }
    return media_temp;
}
```

Figura 49- Função para calcular a media de temperaturas de um único dia

```
void calcular_varias_mediastemp(long long int data_registo[], float temperatura_registo[], int total_registos)
{
    int n_mediastemp;
    printf("Introduza para quantos dias e que pretende calcular a amplitude termica: ");
    scanf("%d", &n_mediastemp);
    for (int j = 0; j < n_mediastemp; j++)
    {
        printf("Medias das Temperaturas: %0.1f\n", calcular_mediastemp(data_registo, temperatura_registo, total_registos));
    }
}
```

Figura 50- Função para calcular a media de temperaturas de vários dias

O output desta última função é o seguinte:

```
Entre com a opcao: 4
Introduza para quantos dias e que pretende calcular a amplitude termica: 1
Introduza a data para calcular a amplitude termica: 20200112
Medias das Temperaturas: 16.5
```

Figura 51- Interface da função calcular_varias_mediastemp

A última opção do nosso menu é a de gravar os dados recolhidos. Esta função é semelhante as funções de gravar em ficheiros binários dos exercícios anteriores. No entanto, nesta função só grava a data, a hora e a temperatura. A função de ler também é igual as outras funções de ler em binários dos exercícios anteriores só que aqui lê a data, a hora e a temperatura. As funções são as seguintes:

```
void gravar_registos(long long int data_registo[], float hora_registo[], float temperatura_registo[], int total_registos)
{
    FILE *f=fopen("temperaturas.dat","wb");
    for (int i=0; i < total_registos; i++)
    {
        fwrite(data_registo, sizeof(long long int[MAX_REGISTOS]),1, f);
        fwrite(hora_registo, sizeof(float[MAX_REGISTOS]),1, f);
        fwrite(temperatura_registo, sizeof(float[MAX_REGISTOS]),1, f);
    }
    fclose(f);
    printf("Dados guardados com sucesso\n");
}
```

Figura 52- Função para guardar os registos recolhidos

```
int ler_registos(long long int data_registo[], float hora_registo[], float temperatura_registo[], int total_registos)
{
    FILE *f=fopen("temperaturas.dat","rb");
    int i=0;
    while (fread(data_registo, sizeof(long long int[MAX_REGISTOS]),1, f) != 0)
    {
        fread(hora_registo, sizeof(float[MAX_REGISTOS]),1, f);
        fread(temperatura_registo, sizeof(float[MAX_REGISTOS]),1, f);
        i++;
    }
    fclose(f);
    return i;
}
```

Figura 53- Função para ler os dados do ficheiro binário

Neste exercício decidimos fazer uma função para apresentar os registos pois assim torna-se mais fácil para o utilizador saber se o ficheiro já tem alguns dados e quais são os dados que estão no ficheiro. A função é a seguinte:

```
void apresentar_registos(long long int data_registo[], float hora_registo[], float temperatura_registo[], int total_registos)
{
    for(int i = 0; i < total_registos; i++)
    {
        printf("Registo %d: ", i+1);
        printf("\n\n");
        printf("Data: %lld\n", data_registo[i] );
        printf("Hora: %.2f\n", hora_registo[i]);
        printf("Temperatura: %.1f\n", temperatura_registo[i]);
    }
}
```

Figura 54- Função para apresentar os registos do termómetro

O output desta função é o seguinte:

```
Entre com a opcao: 2
Registo 1:

Data: 20200112
Hora: 12.34
Temperatura: 19.0
Registo 2:

Data: 20200112
Hora: 15.45
Temperatura: 14.0
```

Figura 55- Interface da função apresentar_registos

3 Conclusão

Com este trabalho aprendemos melhor a trabalhar com ficheiros texto e binários como por exemplo guardar dados nestes ficheiros e ler os dados destes ficheiros. Assim, este trabalho foi uma mais valia para nos não só para nos lembrarmos de matéria antiga como aprender coisa novas neste caso trabalhar com ficheiros binários.

4 Referências



ficha2-ficheiros.pdf