



Instituto Politécnico de Viseu  
Escola Superior de Tecnologia e Gestão de Viseu  
Departamento de Informática

Unidade Curricular: Algoritmos e Programação

Relatório Relativo ao Trabalho Prático

Tema: Registo de Nascimento

Realizado por: Filipe Correia - 25005

Eurico Pinho - 25205

Eduardo Barbosa - 25218

Martim Marques - 23009

Viseu, 2023

Instituto Politécnico de Viseu  
Escola Superior de Tecnologia e Gestão de Viseu  
Departamento de Informática

Relatório relativo ao Trabalho Prático

Curso de Licenciatura em Engenharia Informática

Unidade Curricular de Algoritmos e Programação

## Registo de Nascimento

Ano Letivo 2022/23

Viseu, 2023

---

## RESUMO

Ao longo deste trabalho, procuramos desenvolver uma aplicação que gere o recenseamento das pessoas nascidas num país. No fundo, a pretensão seria criar uma aplicação que estivesse por exemplo num hospital, onde pudéssemos registar um recém nascido e ele ficar como residente no país que efetuou o registo.

Temos como principal objetivo, com o presente trabalho, que as pessoas poupem tempo, ao registar o infante, e que esta aplicação possa ser associada a outras plataformas digitais já existentes, tais como a segurança social, a loja do cidadão, etc.

Este trabalho teve pontos positivos, mas como é normal também tivemos pontos negativos, tais como problemas na criação do código, bugs, e o programa funcionar consistentemente no computador de um membro do grupo, mas no de outro não.

Resolvemos estes problemas, recorrendo à internet, a colegas e a fóruns online, a fim de achar a melhor solução para o nosso trabalho.

Por fim acho que o trabalho correu bem, mas pode ser sempre melhorada, acrescentando-lhe funcionalidades, já que a estrutura está preparada para tal.

---

# ÍNDICE

<b>1. Introdução</b>	<b>1</b>
<b>2. Projeto</b>	<b>2</b>
<b>2.1. Estruturas de dados</b>	<b>2</b>
<b>2.1.1. DATA</b>	
<b>2.1.2. PESSOAS</b>	
<b>2.2. Funções Principais</b>	<b>3</b>
2.2.1. Carregar Ficheiro	
2.2.2. Inserir Pessoas	4
2.2.3. Eliminar Pessoas	5
2.2.4. Pesquisar Pessoas	6
2.2.5. Listar todas as Pessoas	7
2.2.6. Listar Pessoas por Cidade	8
2.2.7. Listar Pessoas que pertencem a uma determinada faixa Etária	9
2.2.8. Listar Pessoas nascidas em uma determinada data	10
2.2.9. Listar Pessoas por Ordem Alfabética de Apelido	11
2.2.10. Cidade onde Nasceram mais Pessoas	12
2.2.11. Média de Idades numa Cidade	13
2.2.12. Mês onde Ocorreram mais Nascimento	14
2.2.13. Listar Pessoas Nascidas a um Domingo	15
2.2.14. Listar Pessoas com aniversário na quaresma num determinado ano	16
2.2.15. Salvar Modificações	17
<b>2.3. Funções Secundárias</b>	<b>18</b>
2.3.1. Comparar Strings	18
2.3.2. Guardar num ficheiro 'cod_nome.txt'	19
2.3.3. Remover linhas em branco do ficheiro	20
2.3.4. Guardar num ficheiro 'cod_dataN_cidade.txt'	21
2.3.5. Mês de Páscoa	21
2.3.6. Dia de Páscoa	22
2.3.7. Mês do Carnaval	22
2.3.8. Dia do Carnaval	23
<b>2.4. Funções de Aspeto</b>	<b>24</b>
2.4.1. Página Inicial	24
2.4.2. Menu	25
2.4.3. Página Final	26
<b>3. Programa em Execução</b>	<b>27</b>
3.1. Início	27
3.2. Menu	27

---

<b>3.3. Listar Pessoas</b>	<b>28</b>
<b>3.4 Menu</b>	<b>28</b>
<b>3.5 Inserir Pessoa</b>	<b>28</b>
<b>3.6 Menu</b>	<b>29</b>
<b>3.7 Guardar Alterações</b>	<b>29</b>
<b>3.8 Menu</b>	<b>29</b>
<b>3.9. Terminar o Programa</b>	<b>29</b>
<b>3.10. Ficheiros antes de Guardar Alterações</b>	<b>30</b>
<b>3.11. Ficheiros depois de Guardar Alterações</b>	<b>31</b>
<b><i>4. Conclusões</i></b>	<b>32</b>
<b><i>5. Bibliografia</i></b>	<b>33</b>

# 1. Introdução

O objetivo principal deste relatório é mostrar e explicar como foi realizado o nosso projeto final em C, de acordo com o enunciado que nos foi fornecido.

Ao longo do relatório serão abordados vários temas. Entre os quais estão:

- Desenvolvimento da estrutura do trabalho;
- Soluções encontradas para os problemas encontrados ao longo do trabalho;
- Explicação detalhada das diferentes partes do programa (structs, funções, menus, etc.)

Como tal, iremos apresentar imagens do código do programa desenvolvido em linguagem C no IDE VSCODE(visual studio code),

Iremos também apresentar imagens do programa na IDE.

## 2. Projeto

Este capítulo aborda a XXX.

### 2.1. Estruturas de dados

Dos elementos referidos XXX

#### 2.1.1. DATA:

Estrutura que distribui a data de nascimento pelas variáveis, 'dia', 'mes' e 'ano', estas inteiras, o que nos permite realizar operações com as mesmas.

```
typedef struct data{  
    int dia;  
    int mes;  
    int ano;  
}DATA;
```

#### 2.1.2. PESSOAS:

Estrutura que distribui os dados das pessoas guardadas nos ficheiros pelas variáveis, 'cod', 'nome', 'cidade', 'dataN', sendo a última uma variável do tipo de dados criado anteriormente.

```
typedef struct pessoas  
{  
    int cod;  
    char *nome;  
    char *cidade;  
    DATA dataN;  
} PESSOAS;
```

## 2.2. Funções Principais

### 2.2.1. Carregar ficheiro (ficheiro\_para\_struct)

Função que abre o ficheiro e carrega para a estrutura.(linhas 986-1049)

```
PESSOAS *ficheiro_para_struct(){
    FILE *File_cod_nome = fopen("cod_nome.txt", "r");

    if (File_cod_nome == NULL)
    {
        perror("Erro na abertura do ficheiro");
        exit(1);
    }
    FILE *File_cod_data_cidade = fopen("cod_dataN_cidade.txt", "r");

    if (File_cod_data_cidade == NULL)
    {
        perror("Erro na abertura do ficheiro");
        exit(1);
    }

    int num_linhas = 0;
    int i;
    char *linha = (char *)malloc(sizeof(char) * MAX_LINHA);
    while (fgets(linha, MAX_LINHA, File_cod_nome) != NULL)
    {
        num_linhas++;
    }

    PESSOAS *pessoa;
    pessoa = (PESSOAS *)malloc(sizeof(PESSOAS) * num_linhas);

    fseek(File_cod_nome, 0, SEEK_SET);
    while (fgets(linha, MAX_LINHA, File_cod_nome) != NULL)
    {
        pessoa->cod = atoi((char *)strtok(linha, "\t"));

        pessoa->nome = (char *)malloc(sizeof(char) * 100);
        strcpy(pessoa->nome, (char *)strtok(NULL, "\n"));

        while (fgets(linha, MAX_LINHA, File_cod_data_cidade) != NULL)
        {
            int cod = atoi((char *)strtok(linha, "\t"));
            if (pessoa->cod == cod)
            {
                pessoa->dataN.dia = atoi((char *)strtok(NULL, "-"));
                pessoa->dataN.mes = atoi((char *)strtok(NULL, "-"));
                pessoa->dataN.ano = atoi((char *)strtok(NULL, "\t"));

                pessoa->cidade = (char *)malloc(sizeof(char) * 100);
                strcpy(pessoa->cidade, (char *)strtok(NULL, "\n"));

                break;
            }
        }
        pessoa++;
    }

    fclose(File_cod_data_cidade);
    fclose(File_cod_nome);

    for (i = 0; i < num_linhas; i++)
    {
        pessoa--;
    }

    return pessoa;
}
```



## 2.2.2. Inserir Pessoas (inserir\_pessoas):

Função que insere pessoas na estrutura 'PESSOAS' para serem guardadas no fim pela função 'guardar\_struct\_ficheiro'. (linhas 828-886)

```
void inserir_pessoas(PESSOAS *pessoa) {
    int cod = 0;
    int dia = 0;
    int mes = 0;
    int ano = 0;
    char aux[5];

    char *nome = (char *)malloc(sizeof(char) * MAX_LINHA);
    char *cidade = (char *)malloc(sizeof(char) * MAX_LINHA);

    system("clear");

    printf("Introduza o código: ");
    scanf("%d", &cod);

    if(cod == 4785392){
        printf("Este código já está reservado para outros fins");
        inserir_pessoas(pessoa);
    }

    printf("\n Introduza o nome: ");
    scanf("%c",&aux);
    fgets(nome, MAX_LINHA, stdin);

    while(dia < 1 || dia > 31 || mes < 1 || mes > 12){
        printf("\n Dia de nascimento formato (dd-mm-yyyy): ");
        scanf("%d-%d-%d", &dia, &mes, &ano);
    }

    printf("\n Cidade: ");
    scanf("%c",&aux[1]);
    fgets(cidade, MAX_LINHA, stdin);

    pessoa->cod = cod;
    pessoa->nome = nome;
    pessoa->dataN.dia = dia;
    pessoa->dataN.mes = mes;
    pessoa->dataN.ano = ano;
    pessoa->cidade = cidade;

    printf("Pessoa adicionada com sucesso.\n");

    char ch;
    fflush(stdin);
    printf(" digite qualquer tecla para voltar ao menu");
    scanf("%c", &ch);
    getchar();
    system("clear");
    menu(pessoa);
}
```

## 2.2.3. Eliminar Pessoas (eliminar\_pessoas)

Função que elimina as pessoas que se encontram no ficheiro, encontra a pessoa através do código que o utilizador tem de inserir.(linhas 888-923)

```
void eliminar_pessoas(PESSOAS *pessoa){
    int cod;

    printf("Insira o código da pessoa que pretende eliminar: ");
    scanf("%d", &cod);
    int iterador = 0;
    while (pessoa->cod != 0)
    {
        if (pessoa->cod == cod)
        {
            pessoa->cod = 4785392;
            pessoa->nome = NULL;
            pessoa->dataN.dia=0;
            pessoa->dataN.mes=0;
            pessoa->dataN.ano=0;
            pessoa->cidade = NULL;
        }

        pessoa++;
        iterador++;
    }

    pessoa -= iterador;

    printf("Pessoa eliminada com sucesso");

    char ch;
    fflush(stdin);
    printf(" digite qualquer tecla para voltar ao menu");
    scanf("%c", &ch);
    getchar();
    system("clear");
    menu(pessoa);
}
```

## 2.2.4. Pesquisar Pessoas (pesquisar\_por\_parte\_nome)

Função que procura no ficheiro uma pessoa, apenas parte do nome é necessário para a encontrar.(linhas 795-826)

```
void pesquisar_por_parte_nome(PESSOAS *pessoa){

    int iterador = 0;

    char *nome = (char *)malloc (sizeof(char) * 100);

    printf("Insira a parte do nome que pretende pesquisar: ");
    scanf("%s", nome);

    while (pessoa->cod != 0){

        if (strstr(pessoa->nome, nome) != NULL)
        {
            printf("\n Codigo: %d    Nome: %s    \n", pessoa->cod, pessoa->nome);
        }

        pessoa++;
        iterador++;
    }

    pessoa -= iterador;

    char ch;
    printf("\n\n");
    fflush(stdin);
    printf(" digite qualquer tecla para voltar ao menu");
    scanf("%c", &ch);
    getchar();
    menu(pessoa);
}
```

## 2.2.5. Listar todas as Pessoas (listar\_pessoas)

Função que lista todas as pessoas do ficheiro que estão guardadas na estrutura. (linhas 963-984) {imagem de lado pois o código é muito comprido}

```
void listar_pessoas(PESSOAS *pessoa){
    int iterador = 0;
    while (pessoa->cod != 0)
    {
        iterador++;
        printf("\n Codigo: %d Nome: %s \n", pessoa->cod, pessoa->nome);
        printf("\t Data de nascimento: %d-%d-%d Cidade: %s \n", pessoa->dataN.dia, pessoa->dataN.mes, pessoa->dataN.ano, pessoa->cidade);
        pessoa++;
    }
    pessoa=iterador;

    char ch;
    fflush(stdin);
    printf("\n digite qualquer tecla para voltar ao menu");
    scanf("%c", &ch);
    getchar();
    menu(pessoa);
}
```

## 2.2.6. Listar Pessoas por Cidade (pesquisar\_por\_cidade)

Função que lista as pessoas que estão em uma cidade. (linhas 925-961)

```
void pesquisar_por_cidade(PESSOAS *pessoa){  
  
    char *cidade = (char *)malloc(sizeof(char) *100);  
  
    printf("Insira a cidade que pretende pesquisar: ");  
    scanf("%s", cidade );  
  
    printf("Pessoas que nasceram na cidade %s: \n\n", cidade);  
  
    int iterador = 0;  
  
    while (pessoa->cod != 0)  
    {  
        if (compare(pessoa->cidade, cidade) == 0)  
        {  
            printf("   Codigo: %d    Nome: %s\n", pessoa->cod, pessoa->nome);  
        }  
  
        pessoa++;  
        iterador++;  
    }  
  
    pessoa -= iterador;  
  
    printf("\n\n");  
  
    char ch;  
    fflush(stdin);  
    printf("digite qualquer tecla para voltar ao menu");  
    scanf("%c", &ch);  
    getchar();  
    menu(pessoa);  
}
```

## 2.2.7. Listar Pessoas que pertencem a uma determinada faixa etária (cidadaos\_faixa\_etaria)

Função que lista as pessoas que se encontram entre idades à escolha do utilizador.  
(linhas 743-793)

```
void cidadaos_faixa_etaria(PESSOAS *pessoa){

    int ano_inicio_faixa_etaria, ano_fim_faixa_etaria;

    int iterador = 0;

    printf("Insira a idade de inicio da faixa etaria : ");
    scanf("%d", &ano_inicio_faixa_etaria);

    printf("Insira a idade de fim da faixa etaria : ");
    scanf("%d", &ano_fim_faixa_etaria);

    while (pessoa->cod != 0){

        int aux = 2023 - pessoa->dataN.ano;

        if (aux >= ano_inicio_faixa_etaria){

            if (aux <= ano_fim_faixa_etaria){

                printf("Codigo: %d \n", pessoa->cod);
                printf("Nome: %s \n", pessoa->nome);

                pessoa++;
                iterador++;

            }else{

                pessoa++;
                iterador++;

            }

        }

        pessoa++;
        iterador++;
    }

    pessoa -= iterador;

    printf("\n\n");

    char ch;
    fflush(stdin);
    printf(" digite qualquer tecla para voltar ao menu");
    scanf("%c", &ch);
    getchar();
    system("clear");
    menu(pessoa);

}
```

## 2.2.8. Listar Pessoas nascidas em uma determinada data (cidadaos\_nascidos\_data)

Função que lista as pessoas que nasceram em uma data inserida pelo utilizador.  
(linhas 680-741)

```
void cidadaos_nascidos_data(PESSOAS *pessoa){
    int dia, mes, ano, iterador = 0;

    printf("Insira o dia que pretende pesquisar: ");
    scanf("%d", &dia);

    printf("Insira o mes que pretende pesquisar: ");
    scanf("%d", &mes);

    printf("Insira o ano que pretende pesquisar: ");
    scanf("%d", &ano);

    while (pessoa->cod != 0){
        if (pessoa->dataN.dia == dia ){
            if (pessoa->dataN.mes == mes){
                if (pessoa->dataN.ano == ano){
                    printf("Codigo: %d \n", pessoa->cod);
                    printf("Nome: %s \n", pessoa->nome);

                    pessoa++;
                    iterador++;
                }else{
                    pessoa++;
                    iterador++;
                }
            }
        }

        pessoa++;
        iterador++;

    }else{
        pessoa++;
        iterador++;
    }

    }

    pessoa -= iterador;

    char ch;
    fflush(stdin);
    printf("\n\n");
    printf(" digite qualquer tecla para voltar ao menu");
    scanf("%c", &ch);
    getchar();
    system("clear");
    menu(pessoa);
}
}
```

## 2.2.9. Listar Pessoas por ordem alfabética de apelido (listar\_pessoas\_alfabetica\_apelido)

Função que lista as pessoas que se encontram na estrutura por ordem alfabética de apelidos. (linhas 176-231)

```
void listar_pessoas_alfabetica_apelido(PESSOAS *pessoa){
    //ordena por ordem alfabetica do ultimo nome
    int i, j, iterador = 0;
    char *aux;
    printf("Lista, por ordem alfabetica do apelido (ultima palavra do nome completo):\n");

    while(pessoa->cod != 0){
        iterador++;
        pessoa++;
    }

    pessoa -= iterador;

    for(i = 0; i < iterador - 1; i++){
        for(j = i + 1; j < iterador; j++){
            char *nome1 = strrchr(pessoa[i].nome, ' ');
            char *nome2 = strrchr(pessoa[j].nome, ' ');

            if(nome1 == NULL){
                nome1 = pessoa[i].nome;
            }

            if(nome2 == NULL){
                nome2 = pessoa[j].nome;
            }
            if(strcmp(nome1, nome2) > 0){
                aux = pessoa[i].nome;
                pessoa[i].nome = pessoa[j].nome;
                pessoa[j].nome = aux;
            }
        }
    }

    for(i = 0; i < iterador; i++){
        printf("\n Cod: %d \t Nome: %s \n", pessoa[i].cod, pessoa[i].nome);
    }

    printf("\n");

    char ch;
    fflush(stdin);
    printf(" digite qualquer tecla para voltar ao menu");
    scanf("%c", &ch);
    getchar();
    system("clear");
    menu(pessoa);
}
```



## 2.2.10. Cidade onde nasceram mais Pessoas (cidade\_mais\_pessoas)

Função que analisa as pessoas que se encontram na estrutura, em especial as cidades a que pertencem, e depois mostra a cidade que tem o maior número de pessoas no ficheiro. (linhas 336-374)

```
void cidade_mais_pessoas(PESSOAS *pessoa){
    int num_pessoas = 0;
    while (pessoa->cod != 0)
    {
        num_pessoas++;
        pessoa++;
    }
    pessoa -= num_pessoas;
    int city_counts[num_pessoas];
    for (int i = 0; i < num_pessoas; i++) {
        city_counts[i] = 0;
    }

    // Count the number of occurrences of each city
    for (int i = 0; i < num_pessoas; i++) {
        int city_index = -1;
        for (int j = 0; j < i; j++) {
            if (strcmp((pessoa + i)->cidade, (pessoa + j)->cidade) == 0) {
                city_index = j;
                break;
            }
        }
        if (city_index == -1) {
            city_index = i;
        }
        city_counts[city_index]++;
    }

    // Find the city with the most occurrences
    int most_popular_city_index = 0;
    for (int i = 1; i < num_pessoas; i++) {
        if (city_counts[i] > city_counts[most_popular_city_index]) {
            most_popular_city_index = i;
        }
    }

    printf("\n\nA cidade com mais pessoas é %s com %d pessoas\n\n", (pessoa + most_popular_city_index)->cidade, city_counts[most_popular_city_index]);
}
```

## 2.2.11. Média de idades numa cidade (media\_idades\_cidade)

Função que calcula a média de idades em uma cidade. linhas (linhas 376-414)

```
void media_idades_cidade(PESSOAS *pessoa){
    char* cidade;
    cidade = (char*)malloc(sizeof(char) * MAX_LINHA);

    printf("\nInsira a cidade cuja quer saber a idade media:\t");
    scanf("%s", cidade);

    int iterador = 0;
    int soma = 0;
    int contador = 0;

    float media = 0;

    while (pessoa->cod != 0)
    {
        if (compare(pessoa->cidade, cidade) == 0)
        {
            soma += (2023 - (pessoa->dataN.ano));
            contador++;
        }

        pessoa++;
        iterador++;
    }

    pessoa -= iterador;
    media = soma / contador;

    printf("\n\nA media de idades das pessoas de %s é de %f\n", cidade, media);

    char ch;
    fflush(stdin);
    printf(" digite qualquer tecla para voltar ao menu");
    scanf("%c", &ch);
    getchar();
    system("clear");
    menu(pessoa);
}
```

## 2.2.12. Mês onde ocorreram mais nascimentos (mes\_mais\_nascimentos)

Função que analisa a estrutura e vê qual é o mês com mais pessoas. (linhas 416-546)

primeira parte

segunda parte

```
void mes_mais_nascimentos(PESSOAS *pessoa){
    int iterador = 0;
    int janeiro = 0, fevereiro = 0, marco = 0, abril = 0;
    int maio = 0, junho = 0, julho = 0, agosto = 0;
    int setembro = 0, outubro = 0, novembro = 0, dezembro = 0;
    int mes1 = pessoa->dataN.mes;

    while (pessoa->cod != 0){
        switch (mes1)
        {
            case 1:
                janeiro++;
                break;
            case 2:
                fevereiro++;
                break;
            case 3:
                marco++;
                break;
            case 4:
                abril++;
                break;
            case 5:
                maio++;
                break;
            case 6:
                junho++;
                break;
            case 7:
                julho++;
                break;
            case 8:
                agosto++;
                break;
            case 9:
                setembro++;
                break;
            case 10:
                outubro++;
                break;
            case 11:
                novembro++;
                break;
            case 12:
                dezembro++;
                break;
            default:
                printf("mes invalido");
                break;
        }

        pessoa++;
        iterador++;
        mes1 = pessoa->dataN.mes;
    }
}
```

```
if (fevereiro > maior)
{
    maior = fevereiro;
    mes_maior = 2;
}
if (marco > maior)
{
    maior = marco;
    mes_maior = 3;
}
if (abril > maior)
{
    maior = abril;
    mes_maior = 4;
}
if (maio > maior)
{
    maior = maio;
    mes_maior = 5;
}
if (junho > maior)
{
    maior = junho;
    mes_maior = 6;
}
if (julho > maior)
{
    maior = julho;
    mes_maior = 7;
}
if (agosto > maior)
{
    maior = agosto;
    mes_maior = 8;
}
if (setembro > maior)
{
    maior = setembro;
    mes_maior = 9;
}
if (outubro > maior)
{
    maior = outubro;
    mes_maior = 10;
}
if (novembro > maior)
{
    maior = novembro;
    mes_maior = 11;
}
if (dezembro > maior)
{
    maior = dezembro;
    mes_maior = 12;
}

printf("O mes com mais nascimentos é %d\n", mes_maior);

pessoa -= iterador;

char ch;
fflush(stdin);
printf(" digite qualquer tecla para voltar ao menu");
scanf("%c", &ch);
getchar();
system("clear");
menu(pessoa);
}
```

## 2.2.13. Listar Pessoas nascidas a um domingo (nascidos\_a\_um\_domingo)

Função que lista as pessoas que nasceram a um domingo, ao fazer os cálculos necessários com base no ano em que a pessoa nasceu. (linhas 100-174)

```
void nascidos_a_um_domingo(PESSOAS *pessoa){

    int i, d, m, y, ano, mes, dia, iterador = 0;
    system("clear");
    printf("Lista das pessoas nascidas a um domingo:\n");

    while(pessoa->cod != 0){

        y = pessoa->dataN.ano;
        m = pessoa->dataN.mes;
        d = pessoa->dataN.dia;

        ano = y - 1900;
        ano = ano / 4;
        ano = ano + y - 1900;

        switch(m){
            case 1:
            case 10:
                mes = 1;
                break;
            case 2:
            case 3:
            case 11:
                mes = 4;
                break;
            case 7:
            case 4:
                mes = 0;
                break;
            case 5:
                mes = 2;
                break;
            case 6:
                mes = 5;
                break;
            case 8:
                mes = 3;
                break;
            case 9:
            case 12:
                mes = 6;
                break;
        }

        ano = ano + mes;
        ano = ano + d;
    }
}
```

```
if(( y > 1900 ) && ( y % 4 == 0 ) && ( m < 2 ) ){
    ano--;
}

dia = ano % 7;

if(dia == 1){
    printf("\n Cod: %d \t Nome: %s \n",pessoa->cod , pessoa->nome);
}
pessoa++;
iterador++;
}

printf("\n");

pessoa -= iterador;

char ch;
fflush(stdin);
printf(" digite qualquer tecla para voltar ao menu");
scanf("%c", &ch);
getchar();
system("clear");
menu(pessoa);
}
```

## 2.2.14. Listar Pessoas com aniversário na quaresma num determinado ano

Função que lista as pessoas que fazem anos na época da quaresma num ano escolhido pelo utilizador. (linhas 614-678)

```
void aniversario_quaresma(PESSOAS *pessoa){

    printf("digite o ano que quer saber os aniversariantes da quaresma:\t");
    int ano;
    scanf("%d", &ano);

    int iterador = 0;

    int mes_carnaval1 = mes_carnaval(ano);
    int mes_pascoa = calcular_mes_pascoa(ano);
    int dia_pascoa = calcular_dia_pascoa(ano);

    int carnaval_dia = dia_pascoa - 47;

    if (carnaval_dia < 1){

        mes_carnaval1--;

        if (mes_carnaval1 < 1){
            mes_carnaval1 = 12;
        }
        if (mes_carnaval1 == 2){
            carnaval_dia += 28;
        }
        else if (mes_carnaval1 == 4 || mes_carnaval1 == 6 || mes_carnaval1 == 9 || mes_carnaval1 == 11){
            carnaval_dia += 30;
        }
        else{
            carnaval_dia += 31;
        }
    }

    while(pessoa->cod != 0){

        if(pessoa->dataN.mes >= mes_carnaval1 && pessoa->dataN.mes <= mes_pascoa){

            if(pessoa->dataN.dia <= dia_pascoa || pessoa->dataN.dia >= carnaval_dia){

                printf("\n Cod: %d",pessoa->cod);
                printf("\n\tName: %s", pessoa->nome);
            }
        }

        pessoa++;
        iterador++;
    }

    pessoa -= iterador;
    char ch;
    fflush(stdin);
    printf(" digite qualquer tecla para voltar ao menu");
    scanf("%c", &ch);
    getchar();
    system("clear");
    menu(pessoa);
}
}
```

## 2.2.15. Salvar modificações (guardar\_struct\_ficheiro)

Função que guarda no ficheiro todas as alterações que foram feitas aos dados que se encontram na estrutura. (linhas 318-336)

```
void guardar_struct_ficheiro(PESSOAS *pessoa){  
  
    guardar_cod_nome(pessoa);  
  
    guardar_cod_dataN_cidade(pessoa);  
  
    remover_linhas_vazias("cod_nome.txt");  
    remover_linhas_vazias("cod_dataN_cidade.txt");  
  
    printf("Ficheiro guardado com sucesso!\n");  
  
    char ch;  
    fflush(stdin);  
    printf(" digite qualquer tecla para voltar ao menu");  
    scanf("%c", &ch);  
    getchar();  
    menu(pessoa);  
  
}
```

## 2.3. Funções Secundárias

- 2.3.1. Comparar strings (compare)  
Função que compara strings pois 'strcmp' fazia conflitos no programa  
(linhas 73-96)

```
int compare(char a[],char b[]) {  
    int flag=0,i=0;  
  
    while(a[i]!='\0' && b[i]!='\0'){  
        if(a[i]!=b[i]){  
            flag=1;  
            break;  
        }  
        i++;  
    }  
  
    if(flag==0){  
        return 0;  
    }else{  
        return 1;  
    }  
}
```

- 2.3.2. Guardar num ficheiro 'cod\_nome.txt'(guardar\_cod\_nome)  
Função que guarda as informações que estão na estrutura no ficheiro 'cod\_nome.txt'  
(linhas 231-251)

```
void guardar_cod_nome(PESSOAS *pessoa){
FILE *ficheiro_cod_nome;

ficheiro_cod_nome = fopen("cod_nome.txt", "w");

while(pessoa->cod != 0){
    if(pessoa->cod == 4785392){
        pessoa++;
    }else{
        fprintf(ficheiro_cod_nome, "%d\t%s\n", pessoa->cod, pessoa->nome);
        pessoa++;
    }
}

fclose(ficheiro_cod_nome);
}
```



- 2.3.3. Remover linhas em branco do ficheiro (remover\_linhas\_vazias)  
Função que remove as linhas em branco ou vazias que estejam no ficheiro.

```
void remover_linhas_vazias(const char *filename) {
    FILE *fp_in = fopen(filename, "r");

    if (fp_in == NULL) {
        fprintf(stderr, "Erro a abrir o ficheiro %s \n", filename);
        exit(1);
    }

    FILE *fp_out = fopen("temp.txt", "w");

    if (fp_out == NULL) {
        fprintf(stderr, "Erro a abrir o ficheiro temporario\n");
        exit(1);
    }

    char line[1024];

    while (fgets(line, sizeof(line), fp_in)) {
        if (strcmp(line, "\n") != 0) {
            fputs(line, fp_out);
        }
    }

    fclose(fp_in);
    fclose(fp_out);

    rename("temp.txt", filename);
}
```

- 2.3.4. Guardar num ficheiro 'cod\_dataN\_cidade.txt' (guardar\_cod\_dataN\_cidade)  
 Função que guarda as informações que estão na estrutura no ficheiro  
 'cod\_dataN\_cidade.txt'. (linhas 289-314)

```
void guardar_cod_dataN_cidade(PESSOAS *pessoa){
FILE *ficheiro_cod_dataN_cidade;

ficheiro_cod_dataN_cidade = fopen("cod_dataN_cidade.txt", "w");

while(pessoa->cod != 0){
    if(pessoa->cod == 4785392){
        pessoa++;
    }else{
        char *string = (char*) malloc(100*sizeof(char));
        sprintf(string, "%d\t%d-%d-%d\t%s\n", pessoa->cod, pessoa->dataN.dia, pessoa->dataN.mes, pessoa->dataN.ano, pessoa->cidade);

        size_t count = fwrite(string, sizeof(char), strlen(string), ficheiro_cod_dataN_cidade);

        printf("%s \n", string);

        pessoa++;
    }
}

fclose(ficheiro_cod_dataN_cidade);
}
```

- 2.3.5. Mês de Páscoa (calcular\_mes\_pascoa)  
 Função que calcula o mês em que a páscoa calha em um ano designado pelo  
 utilizador. (linhas 548-564)

```
int calcular_mes_pascoa(int ano){
    int a = ano % 19;
    int b = ano / 100;
    int c = ano % 100;
    int d = b / 4;
    int e = b % 4;
    int f = (b + 8) / 25;
    int g = (b - f + 1) / 3;
    int h = (19 * a + b - d - g + 15) % 30;
    int i = c / 4;
    int k = c % 4;
    int l = (32 + 2 * e + 2 * i - h - k) % 7;
    int m = (a + 11 * h + 22 * l) / 451;
    int mes = (h + l - 7 * m + 114) / 31;

    return mes;
}
```

## 2.3.6. Dia de Páscoa (calcular\_dia\_pascoa)

Função que calcula em que dia do mês obtido na função 'calcular\_mes\_pascoa' é que a páscoa vai calhar. (linhas 566-582)

```
int calcular_dia_pascoa(int ano){
    int a = ano % 19;
    int b = ano / 100;
    int c = ano % 100;
    int d = b / 4;
    int e = b % 4;
    int f = (b + 8) / 25;
    int g = (b - f + 1) / 3;
    int h = (19 * a + b - d - g + 15) % 30;
    int i = c / 4;
    int k = c % 4;
    int l = (32 + 2 * e + 2 * i - h - k) % 7;
    int m = (a + 11 * h + 22 * l) / 451;
    int dia = ((h + l - 7 * m + 114) % 31) + 1;

    return dia;
}
```

## 2.3.7. Mês do Carnaval (mes\_carnaval)

Função que calcula o mês em que o carnaval calha com base nas funções 'mes\_pascoa' e 'dia\_pascoa'. (linhas 584-597)

```
int mes_carnaval(int ano){
    int mes_pascoa = calcular_mes_pascoa(ano);
    int dia_pascoa = calcular_dia_pascoa(ano);

    if (mes_pascoa == 3 && dia_pascoa > 21)
    {
        return 2;
    }
    else
    {
        return 3;
    }
}
```

## 2.3.8.

Dia do Carnaval (dia\_carnaval)

Função que calcula o dia em que o carnaval calha com base nas funções 'mes\_pascoa' e 'dia\_pascoa'. (linhas 599-612)

```
int dia_carnaval(int ano){  
    int mes_pascoa = calcular_mes_pascoa(ano);  
    int dia_pascoa = calcular_dia_pascoa(ano);  
  
    if (mes_pascoa == 3 && dia_pascoa > 21)  
    {  
        return dia_pascoa - 47;  
    }  
    else  
    {  
        return dia_pascoa - 46;  
    }  
}
```

## 2.4. Funções de Aspeto

- 2.4.1. Página Inicial (paginainicial)  
 Função criada para a abertura de um ficheiro externo ao código principal para apresentar o programa ao utilizador. (linhas 1051-1098)

```
void paginainicial(PESSOAS *pessoa){
    system("clear");
    int option;

    printf("\n");
    FILE *fptr;
    // Open file
    fptr = fopen("estgv.txt", "r");
    if (fptr == NULL)
    {
        printf("Não é possível abrir o ficheiro \n");
        exit(0);
    }

    // Read contents from file
    int c;
    c = fgetc(fptr);
    while (c != EOF)
    {
        printf ("%c", c);
        c = fgetc(fptr);
    }
    printf("\n");
    fclose(fptr);

    printf("\n");
    printf("-----\n");
    printf("      |                                     |\n");
    printf("      |      BEM VINDO AO PROJETO VERSÃO B  CLICA NA TECLA 1 PARA COMEÇAR      |\n");
    printf("      |      PARA SAIR DO PROGRAMA CLICAR NA TECLA 0                          |\n");
    printf("      |-----\n");
    printf("\n Option:\t");
    scanf(" %d", &option);
    system("clear");

    switch (option){
    case 0:
        creditos_finais();
        break;
    case 1:
        menu(pessoa);
        break;
    }
}
```

## 2.4.2. Menu (menu)

A função que contém o menu com as opções do que o utilizador pode usar no programa para escolher as operações desejadas. (linhas 1114-1195)

```
void menu(PESSOAS *pessoa)
{
    system("clear");
    fflush(stdin);

    int op;

    printf("\n");
    printf("-----\n");
    printf("Menu Principal\n");
    printf("Selecione uma opção:\n");
    printf("-----\n");
    printf("1 - Inserir pessoa\n");
    printf("3 - Pesquisar por parte do nome\n");
    printf("5 - Listar cidadão que nasceu numa determinada cidade\n");
    printf("7 - Listar cidadãos que nasceram numa determinada data\n");
    printf("9 - Determinar a cidade onde nasceram mais pessoas\n");
    printf("11 - Determinar o mês em que ocorreram mais nascimentos\n");
    printf("13 - Listar Cidadãos cujo aniversário, num determinado ano, é na quaresma\n");
    printf("0 - Sair da aplicação\n");
    printf("-----\n");
    printf("\n Opção:\t");
    scanf("%d", &op);
    system("clear");
    fflush(stdin);
    switch (op){
        case 0:
            credits_finais();
            break;
        case 1:
            inserir_pessoas(pessoa);
            break;
        case 2:
            eliminar_pessoas(pessoa);
            break;
        case 3:
            pesquisar_por_parte_nome(pessoa);
            break;
        case 4:
            listar_pessoas(pessoa);
            break;
        case 5:
            pesquisar_por_cidade(pessoa);
            break;
        case 6:
            cidadaos_faixa_etaria(pessoa);
            break;
        case 7:
            cidadaos_nascidos_data(pessoa);
            break;
        case 8:
            listar_pessoas_alfabetica_apelido(pessoa);
            break;
        case 9:
            cidade_mais_pessoas(pessoa);
            break;
        case 10:
            media_idades_cidade(pessoa);
            break;
        case 11:
            mes_mais_nascimentos(pessoa);
            break;
        case 12:
            nascidos_a_um_domingo(pessoa);
            break;
        case 13:
            aniversario_quaresma(pessoa);
            break;
        case 14:
            guardar_struct_ficheiro(pessoa);
            break;
        default:
            printf("Opção inválida\n");
            getchar();
            menu(pessoa);
            break;
    }
}
```

## 2.4.3. Página final (creditos\_finais)

A função criada para informar e mostrar ao utilizador que o programa foi parado e que já é possível fechar a janela (linhas 1100-1112)

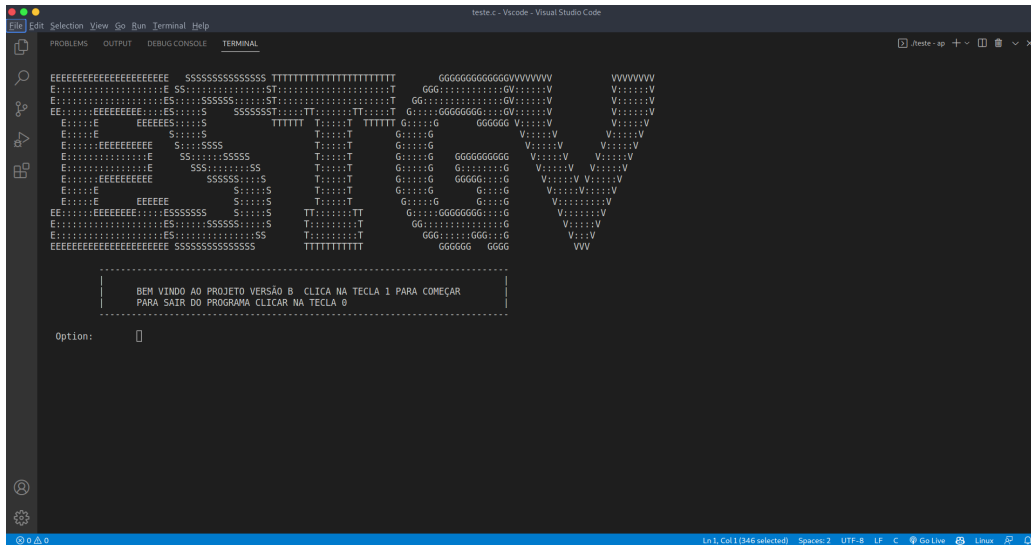
```
void creditos_finais(){  
    system("clear");  
    printf("\n");printf("\n");  
    printf(" ----- \n");  
    printf(" |          Programa:          | \n");  
    printf(" |          Registo de Nascimentos          | \n");  
    printf(" |      É seguro fechar a janela      | \n");  
    printf(" |          < Grupo 12 | Tema B >          | \n");  
    printf(" ----- \n");  
    printf("\n");printf("\n");printf("\n");  
    exit(0);  
}
```

## 3. Programa em Execução

Nesta Parte será possível ver os outputs de um percurso que se pode seguir no programa

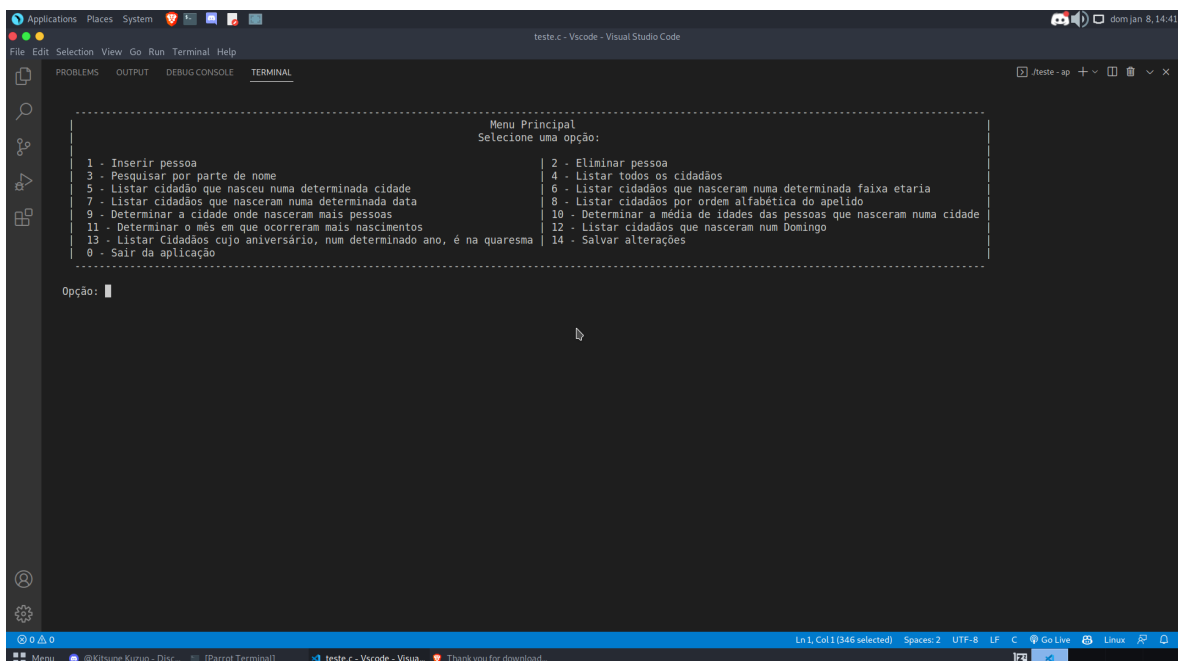
### 3.1. Início

Quando o programa começa a rodar a primeira coisa que aparece é a tela da página inicial.



### 3.2. Menu

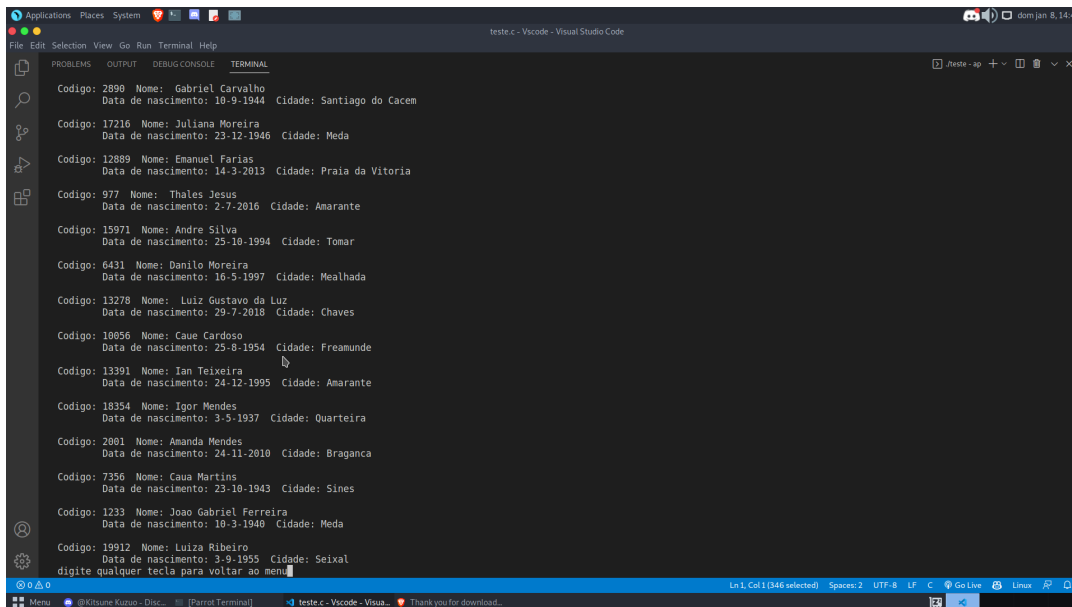
Ao escolher a opção ‘1’ o programa vai redirecionar o utilizador ao ‘menu’ onde é possível escolher o que se quer fazer.





### 3.3. Listar pessoas

Ao escolher a opção ‘4’ é possível listar todos os cidadãos que estão na estrutura,



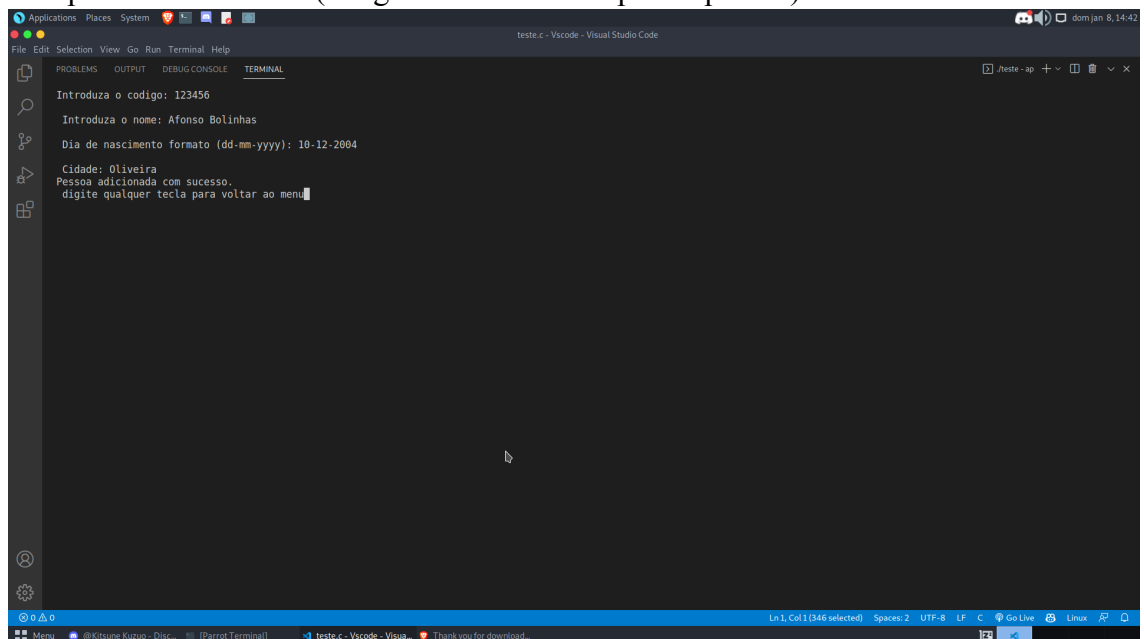
```
teste.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Codigo: 2890 Nome: Gabriel Carvalho
Data de nascimento: 10-9-1944 Cidade: Santiago do Cacem
Codigo: 17216 Nome: Juliana Moreira
Data de nascimento: 23-12-1946 Cidade: Meda
Codigo: 12889 Nome: Emanuel Farias
Data de nascimento: 14-3-2013 Cidade: Praia da Vitoria
Codigo: 977 Nome: Thales Jesus
Data de nascimento: 2-7-2016 Cidade: Amarante
Codigo: 15971 Nome: Andre Silva
Data de nascimento: 25-10-1994 Cidade: Tomar
Codigo: 6431 Nome: Danilo Moreira
Data de nascimento: 16-5-1997 Cidade: Mealhada
Codigo: 13278 Nome: Luiz Gustavo da Luz
Data de nascimento: 29-7-2018 Cidade: Chaves
Codigo: 10056 Nome: Caue Cardoso
Data de nascimento: 25-8-1954 Cidade: Freamunde
Codigo: 13391 Nome: Ian Teixeira
Data de nascimento: 24-12-1995 Cidade: Amarante
Codigo: 18354 Nome: Igor Mendes
Data de nascimento: 3-5-1937 Cidade: Quarteira
Codigo: 2001 Nome: Amanda Mendes
Data de nascimento: 24-11-2010 Cidade: Braganca
Codigo: 7356 Nome: Caua Martins
Data de nascimento: 23-10-1943 Cidade: Sines
Codigo: 1233 Nome: Joao Gabriel Ferreira
Data de nascimento: 10-3-1940 Cidade: Meda
Codigo: 18912 Nome: Luiza Ribeiro
Data de nascimento: 3-9-1955 Cidade: Seixal
digite qualquer tecla para voltar ao menu
```

### 3.4. Menu

Ao clicar no enter o programa traz o utilizador de volta ao menu principal.

### 3.5. Inserir Pessoa

Ao escolher a opção ‘1’ o utilizador é direcionado a uma tela onde é possível adicionar uma pessoa ao ficheiro. (imagem com um exemplo de pessoa)



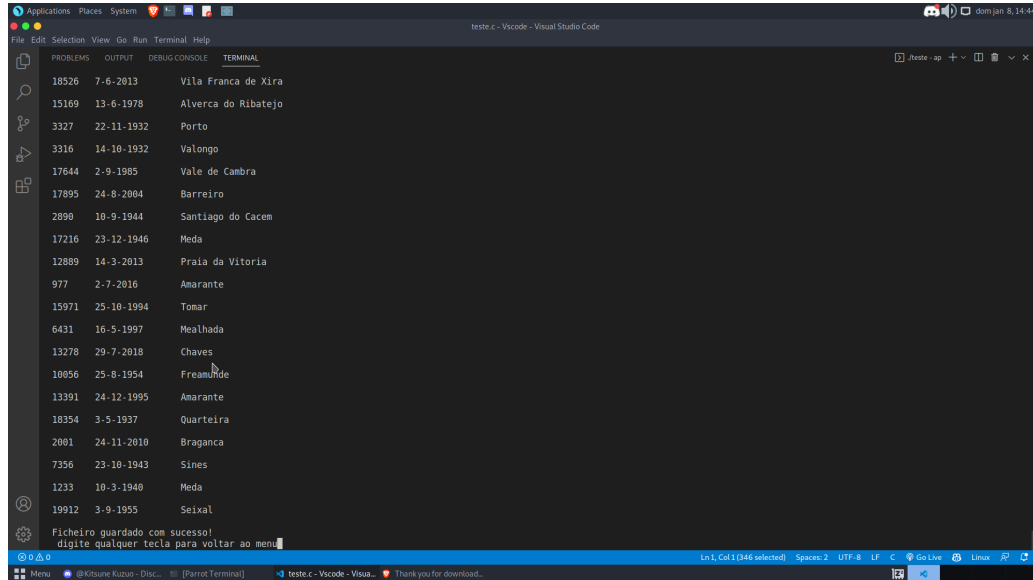
```
teste.c - Visual Studio Code
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Introduza o codigo: 123456
Introduza o nome: Afonso Bolinhas
Dia de nascimento formato (dd-mm-yyyy): 10-12-2004
Cidade: Oliveira
Pessoa adicionada com sucesso.
digite qualquer tecla para voltar ao menu
```

### 3.6. Menu

Ao clicar no enter o programa traz o utilizador de volta ao menu principal.

### 3.7. Guardar alterações

Ao escolher a opção '14' todas as alterações feitas são guardadas nos ficheiros.

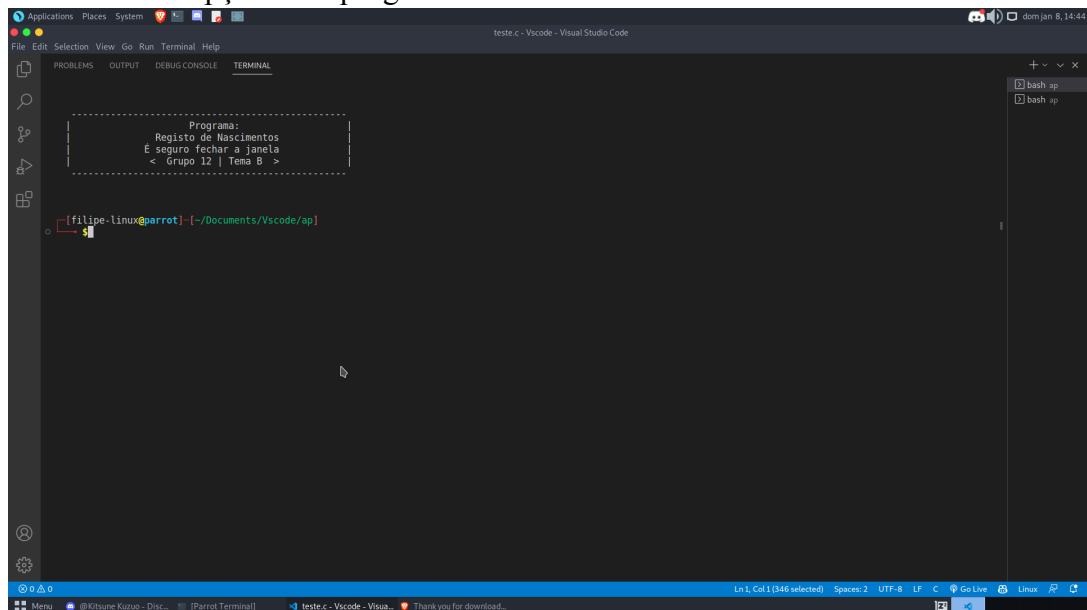


### 3.8. Menu

Ao clicar no enter o programa traz o utilizador de volta ao menu principal.

### 3.9. Terminar o programa

Ao escolher a opção '0' o programa termina e mostra ao utilizador uma tela de saída



## 3.10. Ficheiros antes de guardar alterações

cod\_dataN\_cidade.txt

Visual Studio Code interface showing the file `cod_dataN_cidade.txt` in the nano editor. The file contains a list of names and dates, such as:

```
13772 13-8-1988 Albergaria-a-Velha
3308 14-4-1933 Seixal
10060 15-3-2008 Valbom
6516 23-8-1978 Tavira
17439 12-4-1990 Leiria
7757 1-3-2011 Alcaccer do Sal
17305 14-5-2000 Barreiro
10457 13-9-1989 Gandra
15774 1-8-1982 Praia da Vitoria
5831 9-2-2003 Oliveira de Azemeis
10438 4-7-1944 Vila Real
13732 18-3-1988 Abrantes
5786 13-11-2012 Leiria
15028 21-7-1987 Odivelas
3853 1-10-1986 Costa da Caparica
3108 10-9-1966 Tondela
11661 7-12-1931 Vila Nova de Famalicao
96 13-3-2019 Vila Real de Santo Antonio
7173 26-2-1930 Viseu
2701 11-5-2006 Alcaccer do Sal
16354 22-8-1982 Sao Pedro do Sul
8217 4-3-1973 Torres Novas
16800 14-3-1991 Ribeira Grande
2102 28-7-1997 Anadia
4966 3-4-1976 Vendas Novas
380 19-6-1944 Lagos
19168 30-7-1985 Vila Real
6933 10-2-1954 Gandra
9869 24-5-1983 Miranda do Douro / Miranda de l Douro
275 20-3-2007 Castelo Branco
7288 30-3-1994 Lixa
2551 27-1-2006 Alverca do Ribatejo
12579 11-8-1935 Sabugal
8062 15-10-1936 Barcelos
605 16-9-1979 Pinhel
11634 11-8-1959 Santa Cruz
10061 22-6-1970 Torres Novas
15593 4-7-2004 Reguengos de Monsaraz
```

The status bar at the bottom indicates: `Ln1, Col1 (346 selected) Spaces:2 UTF-8 LF C`.

cod\_nome.txt

Visual Studio Code interface showing the file `cod_nome.txt` in the nano editor. The file contains a list of names, such as:

```
13772 Erick Barros
3308 Alexandre Araujo
10060 Ana Beatriz Silva
6516 Alana Sales
17439 Emanuella Pires
7757 Carolina Viana
17305 Nicolas da Rosa
10457 Luis Lima
15774 Lara Souza
5831 Sabrina Souza
10438 Ana Clara Costa
13732 Elisa Silva
5786 Ana Julia Caldeira
15028 Caua Cardoso
3853 Julia Barbosa
3108 Ana Julia Santos
11661 Yasmin das Neves
96 Marcelo Carvalho
7173 Caroline Fogaca
2701 Gabrielly da Luz
16354 Luiz Gustavo Dias
8217 Melissa da Rocha
16800 Maria Alice Jesus
2102 Thales Aragao
4966 Leandro Ribeiro
380 Helena da Mata
19168 Bernardo Barbosa
6933 Andre Almeida
9869 Eloah Alves
275 Lucca Martins
7286 Fernanda Lima
2551 Ana Sophia Gomes
12579 Breno Melo
8062 Ana Julia Cunha
605 Noah da Luz
11634 Pedro Henrique Novaes
10061 Catarina Rezende
15593 Joao Lucas da Cunha
```

The status bar at the bottom indicates: `Ln1, Col1 (346 selected) Spaces:2 UTF-8 LF C`. A notification box is visible in the bottom right corner.

## 3.11. Ficheiros depois de guardar alterações

cod\_dataN\_cidade.txt

```

GNU nano 5.4 cod_dataN_cidade.txt
123456 10-12-2004 Oliveira
3308 14-4-1933 Seixal
10060 15-3-2008 Valbom
6516 23-8-1978 Tavira
17439 12-4-1990 Leiria
7757 1-3-2011 Alcaccer do Sal
17305 14-5-2000 Barreiro
10457 13-9-1989 Gandra
15774 1-8-1982 Praia da Vitoria
5831 9-2-2003 Oliveira de Azemeis
10438 4-7-1944 Vila Real
13732 18-3-1988 Abrantes
5786 13-11-2012 Leiria
15028 21-7-1987 Odivelas
3853 1-10-1986 Costa da Caparica
3108 10-9-1966 Tondela
11661 7-12-1931 Vila Nova de Famalicao
96 13-3-2019 Vila Real de Santo Antonio
7173 26-2-1930 Viseu
2701 11-5-2006 Alcaccer do Sal
16354 22-8-1982 Sao Pedro do Sul
8217 4-3-1973 Torres Novas
16800 14-3-1991 Ribeira Grande
2102 28-7-1997 Anadia
4966 3-4-1976 Vendas Novas
380 19-6-1944 Lagos
19168 30-7-1985 Vila Real
6933 10-2-1954 Gandra
9869 24-5-1983 Miranda do Douro / Miranda de l Douro
275 20-3-2007 Castelo Branco
7288 30-3-1994 Lixa
2551 27-1-2006 Alverca do Ribatejo
12579 11-8-1935 Sabugal
8062 15-10-1936 Barcelos
605 16-9-1979 Pinhel
11634 11-8-1959 Santa Cruz
10061 22-6-1970 Torres Novas
15593 4-7-2004 Reguengos de Monsaraz
  
```

cod\_nome.txt

```

GNU nano 5.4 cod_nome.txt
123456 Afonso Bolinhas
3308 Alexandre Araujo
10060 Ana Beatriz Silva
6516 Alana Sales
17439 Emanuella Pires
7757 Carolina Viana
17305 Nicolas da Rosa
10457 Luis Lima
15774 Lara Souza
5831 Sabrina Souza
10438 Ana Clara Costa
13732 Elisa Silva
5786 Ana Julia Caldeira
15028 Caua Cardoso
3853 Julia Barbosa
3108 Ana Julia Santos
11661 Yasmin das Neves
96 Marcelo Carvalho
7173 Caroline Fogaca
2701 Gabrielly da Luz
16354 Luiz Gustavo Dias
8217 Melissa da Rocha
16800 Maria Alice Jesus
2102 Thales Aragao
4966 Leandro Ribeiro
380 Helena da Mata
19168 Bernardo Barbosa
6933 Andre Almeida
9869 Eloah Alves
275 Lucca Martins
7288 Fernanda Lima
2551 Ana Sophia Gomes
12579 Breno Melo
8062 Ana Julia Cunha
605 Noah da Luz
11634 Pedro Henrique Novaes
10061 Catarina Rezende
15593 Joao Lucas da Cunha
  
```

## 4. Conclusões

Para concluir, é importante refletir sobre o que foi feito neste projeto e na elaboração deste programa.

A elaboração deste programa permitiu-nos aplicar aquilo que aprendemos ao longo deste primeiro semestre.

Serviu também para experimentar em prática todo o tipo de funções, desde o uso de estruturas, alocação de memória, ficheiros, etc.

Numa reflexão mais geral, a realização de projetos deste género ajudam-nos a perceber a forma como é feito um programa.

Apesar de muitas vezes ser frustrante por o código parar de funcionar e logo a seguir sem uma pessoa lhe mexer voltar a funcionar, é algo bastante importante para perceber a realidade da construção de um programa/ aplicação.



## 5. Bibliografia

- 5.1. <https://www.javatpoint.com/program-to-remove-all-the-white-spaces-from-a-string>
- 5.2. [https://www.tutorialspoint.com/c\\_standard\\_library/c\\_function\\_fwrite.htm](https://www.tutorialspoint.com/c_standard_library/c_function_fwrite.htm)
- 5.3. <https://www.geeksforgeeks.org/taking-string-input-space-c-3-different-methods/>
- 5.4. [https://pt.wikipedia.org/wiki/C%C3%A1culo\\_da\\_P%C3%A1scoa](https://pt.wikipedia.org/wiki/C%C3%A1culo_da_P%C3%A1scoa)
- 5.5. [https://www.suapesquisa.com/carnaval/data\\_definida.ht](https://www.suapesquisa.com/carnaval/data_definida.ht)





