FICHEIROS

Todos os dados com que se tem trabalhado até aqui são dados voláteis – quer dizer que são armazenados temporariamente na RAM do computador, enquanto o programa está a correr. Quando se sai do programa, toda a informação desaparece.

Surge a necessidade de guardar essa informação em suportes de memória secundária ou externa (como um disco ou pen drive), para além do tempo em que o programa está a correr no computador e reutilizá-la, nesse ou noutro programa.

Ficheiro – unidade de informação armazenada fisicamente num suporte de memória secundária.

NOTA:

Há ficheiros que são legíveis diretamente no sistema operativo (Type ou Edit), pois são constituídos por carateres ASCII – são os **Ficheiros de Texto**.

Outros tipos de ficheiros não são formados apenas por carateres, ou não têm qualquer caráter legível – são os ficheiros em código – máquina (executáveis). Estes últimos são chamados **ficheiros binários**, porque a informação se encontra em formato binário.

Assim, em C, os ficheiros, em que se armazenam os dados com que os programas trabalham, são de 2 tipos:

1- "TEXT FILES" – ficheiros de texto

Aqui a informação é totalmente armazenada em formato de carateres ASCII. Podem ser criados, consultados, modificados no Sistema Operativo ou no editor de texto fora do programa que os usa.

O **acesso aos dados** é do tipo **SEQUENCIAL,** ou seja, a leitura dos dados não pode ser iniciada num ponto qualquer escolhido pelo utilizador, mas sempre o início e percorrer todos os elementos até chegar ao ponto pretendido.

2- "TYPED FILES" – ficheiros de tipo não texto. Ficheiros definidos pelo utilizador.

Estes agrupam dados simples ou estruturados, no formato binário, não legíveis nem manipuláveis fora do programa em que foram criados. São ficheiros de tipos definidos pelo programador.

Podemos ter ficheiros de números inteiros ou reais, matrizes, estruturas, etc.

Em particular, os ficheiros de estruturas permitem manipular dados num formato bem estruturado para trabalho com informação externa (como um disco ou pen drive).

O **acesso aos dados** pode ser feito de forma **aleatória,** ou seja, por escolha da posição pretendida (**ACESSO DIRETO**).

NOTA:

Um ficheiro de dados, permite o armazenamento da informação num suporte externo e reunir uma coleção de dados sem tamanho fixo. Por conseguinte, um ficheiro pode ser acrescentado com mais dados ou rescrito com um número diferente de elementos.

Qualquer um dos outros dados estruturados (vetores e estruturas) tem um tamanho que é determinado na declaração (do tipo ou variável).

OPERAÇÕES COM FICHEIROS

Como se opera com ficheiros?

As estruturas de dados Vetores e Estruturas são criadas e manipuladas apenas ao nível da memória interna ou primária. Quanto aos ficheiros, a situação é diferente. Nos ficheiros temos uma manipulação a 2 níveis:

A nível interno do programa: com variáveis que identificam os ficheiros e os dados a ler ou escrever nessas unidades de informação.

A nível externo do programa: ou de interação entre o programa e os dispositivos físicos, onde são armazenados os ficheiros.

Operações Principais (no trabalho com ficheiros) são:

_ Declaração de tipos e variáveis de ficheiros;

_ Associação de uma variável de ficheiro com o nome externo de ficheiro;

_ Criação de novos ficheiros ou reescrita total de um ficheiro já existente;

_ Escrita de informação num ficheiro;

_ Abertura de um ficheiro para leitura;

_ Procura de dados num ficheiro;

_ Fecho de um ficheiro aberto.

STREAMS

Conjunto sequencial de carateres, isto é, um conjunto de bytes, sem qualquer estrutura interna.

Em C, cada stream está ligada a um ficheiro, o qual pode não corresponder fisicamente a um ficheiro existente no disco, como é o caso do teclado.

DECLARAÇÃO DE VARIÁVEIS DE FICHEIROS

A declaração de variáveis para o processamento de ficheiros faz-se usando o **tipo FILE** definido na biblioteca <stdio.h>

A declaração de uma variável do tipo **FILE***, faz com que esta seja um apontador para o tipo FILE.

A abertura de um ficheiro é realizada utilizando a função **fopen**, incluída em stdio.h, e que é uma operação básica de I/O.

SINTAXE:

```
FILE * fopen (const char *filename, const char * mode)
```

Filename — string que contém o nome físico do ficheiro (Exemplo: **dados.dat**);
Mode — string que contém o modo de abertura do ficheiro:

Nota: Os parâmetros são constantes (const), pois não irão ser alterados dentro da função.

Se a função fopen conseguir abrir um ficheiro com sucesso, cria em memória uma estrutura (do tipo FILE) que representa toda a informação necessária relativa ao ficheiro que estiver a processar, devolvendo o endereço em que essa estrutura foi criada. Caso não tenha conseguido abrir o ficheiro, devolve NULL.

Nome de um ficheiro

O nome de um ficheiro é armazenado numa string e deve representar fielmente e na totalidade o nome do ficheiro, tal como é visto pelo sistema operativo em que se está a operar.

MODOS DE ABERTURA DE UM FICHEIRO

"r"— read (abertura de ficheiro para leitura)
Caso não o possa abrir a função devolve NULL (por este não existir ou por não ter permissões para tal);

"W"— write (Abertura do ficheiro para escrita)
É criado um novo ficheiro com o nome passado à função. Se já existir um ficheiro com o mesmo nome, é apagado e criado um novo, perdendo-se assim toda a informação nele contida. Caso não o possa criar a função devolve NULL (por este conter um nome inválido, por não existir espaço em disco, por se tentar criar o ficheiro num disco protegido, por se estar a tentar criar um ficheiro como nome de um diretório ou por não ter permissões para tal).

"a"— append (Abertura de ficheiro para acrescento)
Se o ficheiro não existir, cria-o e funciona tal e qual como o modo "w". Se o ficheiro já existir, o apontador para o ficheiro coloca-se no final deste, de modo a permitir a escrita de dados de forma sequencial e a partir dos dados já existentes.

Existe a possibilidade de abrir um ficheiro de modo a permitir, em simultâneo, operações de leitura e escrita, colocando um sinal de + a seguir ao modo:

"r+" "w+" "a+"

Com funcionamentos idênticos aos descritos acima.
O modo de abertura pode ainda ser combinado com o tipo de processamento que se pretende dar ao ficheiro:

"t"— ficheiro de texto;
"b"— ficheiro binário. **Exemplos:** "rb", "wt"

SÍNTESE

Modo de abertura	Descrição	Leitura	Escrita	Se ficheiro não existe	Se ficheiro já existe	Posição inicial
r	Ler	SIM	Não	NULL	OK	Início
w	Escrever	Não	SIM	Cria	Recria	Início
a	Acrescentar	Não	SIM	Cria	OK	Fim
r+	Ler/Escrever	SIM	SIM	Cria	Altera dados	Início
w+	Ler/Escrever	SIM	SIM	Cria	Recria	Início
a+	Ler/Escrever	SIM	SIM	Cria	Acrescenta dados	Fim

FECHO DE UM FICHEIRO

Sintaxe:

```
int fclose (FILE *fich)
```

Esta função devolve 0 em caso de sucesso ou a constante EOF em caso de erro.

Nota: embora os ficheiros sejam automaticamente fechados quando uma aplicação termina, deve-se proceder ao seu fecho porque no caso de falha de energia ou desligar súbito do PC, os dados presentes no buffer não irão ser colocados no ficheiro, podendo ocorrer perda dos mesmos.

Exemplo de utilização das funções fopen e fclose (abertura e fecho de um ficheiro existente):

```
#include <stdio.h>
main()
{
        FILE *fp;
        char s[50];
        puts("Introduza o nome do ficheiro "); gets(s);
        /*abrir o ficheiro*/
        fp=fopen (s,"r");       /*modo de abertura: leitura, sendo colocado em fp o endereço que o
                                representa e que foi criada em memória pela função fopen
                               (associa o ficheiro físico à variável fp)*/

        /*verificar se a abertura do ficheiro foi efetuada com sucesso*/
        if(fp==NULL)
                printf("Impossível abrir o ficheiro %s\n", s);
        else
                printf("Ficheiro %s aberto com sucesso!!!\n", s);
        fclose (fp);
}
```

LEITURA DE CARATERES DE UM FICHEIRO

Existem várias funções para ler carateres de um ficheiro, sendo a mais utilizada a seguinte:

```
int fgetc (FILE *fich)
```

A função **fgetc** lê um caráter no ficheiro, passado como parâmetro (previamente aberto pela função fopen) e devolve-o como resultado da função. Se não existir qualquer caráter no ficheiro, isto é, se a função detetar uma situação de **end-of-file**, devolve a constante **EOF**.

- A constante simbólica EOF(-1) serve de constante indicadora de end-of-file.
- Todas as funções de leitura se posicionam automaticamente a seguir ao valor lido do ficheiro, não tendo o utilizador que movimentar o apontador para a posição seguinte no ficheiro.

ESCRITA DE CARATERES NUM FICHEIRO

A escrita de carateres num ficheiro pode ser realizada recorrendo à função:

```
int fputc (int ch, FILE *fich)
```

que escreve o caráter *ch* no ficheiro *fich*. Esta função devolve o carater *ch* em caso de sucesso ou EOF, caso contrário.

Exemplo (escrita e leitura de carateres em ficheiro):

```
#include <stdio.h>
#include <stdlib.h>  /* por causa da função exit*/
main()
{
        FILE *fp;
        char s[ ]="carateres.txt";
        int ch;  /*para ler os carateres*/
        fp=fopen (s,"w");     /* abre o ficheiro em modo de escrita associando-o a fp*/
        /*verificar se a abertura do ficheiro foi efetuada com sucesso*/
        if(fp==NULL)
        {
                printf("Impossível abrir o ficheiro %s\n", s);
                exit(1);          /*terminar programa*/
        }
        do
        {
                ch=getchar();
                fputc(ch, fp);

        } while(ch!='\n');
        fclose (fp);
        fp=fopen(s,"r");
        printf("Conteúdo do ficheiro %s\n", s);
        while ((ch=fgetc(fp))!=EOF)
                putchar(ch);
        fclose (fp);
        }
}
```

Função fgets

```
char * fgets (char * str, int num, FILE * stream)
```

A função **fgets** é a variante do gets para ficheiros. O fgets tem 3 argumentos:

- um array de carateres onde a linha vai ser guardada;
- um número que indica o número máximo de carateres que pode ser lido (incluindo o '\0').
- o ficheiro de onde pretendemos ler.

Se chegarmos ao fim do ficheiro ou acontecer outro erro qualquer, o fgets retorna NULL, caso contrário a função devolve a string lida e apontada por str.

Função fputs

```
int fputs ( const char * str, FILE * stream)
```

que devolve um valor não negativo, caso a escrita seja realizada com sucesso, ou EOF caso contrário.

Exemplo 1:

```
#include <stdio.h>
#include <stdlib.h>
main()
{
   char s[30];
   FILE *fout;
   fout=fopen("exp.txt","w");
   gets(s);
   fputs(s,fout);
   fclose(fout);
   fout=fopen("exp.txt","r");
   printf("\nConteúdo do ficheiro:\n")
   fgets(s,30, fout);
   puts(s);
}
```

Exemplo 2:

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    char s[3][30];
    FILE *fout;
    int i;
    fout=fopen("exp.txt","w");
```

```
  for (i=0; i<3; i++)
  {
        gets(s[i]); fputs(s[i], fout);
        fputc('\n', fout);        /* para armazenar as string em linhas diferentes*/
  }
  fclose(fout);
  fout=fopen("exp.txt","r");
  printf("\nconteúdo do ficheiro terminada\n");
  i=0;
  while(fgets(s[i],30, fout)!=NULL)
        puts(s[i++]);
}
```

Questão: o que fazem exatamente os programas dos exemplos 1 e 2?

INPUT E OUTPUT FORMATADO

Para o I/O formatado utilizam-se as funções **fscanf** e **fprintf**, que funcionam da mesma forma que as funções scanf e printf, tendo apenas mais um parâmetro inicial que corresponde ao ficheiro onde o processamento irá ser realizado.

int fscanf(FILE *fich, const char *format,…)

int fprintf(FILE *fich, const char *format,…)

- A função **fscanf** devolve **EOF,** se tiver detetado end-of-file, ou devolve o n.º de parâmetros que conseguiu ler com sucesso.
- Estas funções apenas podem ser utilizadas para ficheiros abertos em modo de texto.

Exemplo:

```
while (fscanf(fp,"%c",&ch)!=EOF)
        fprintf(fp,"%c",ch);          /*devendo a variável ch ser declarada com o tipo char*/
```

Exercício resolvido 1:

Registar os nomes e preços de alguns produtos e respetivas quantidades num ficheiro de texto. Devem ainda ser lidos os dados do ficheiro e exibidos no ecrã.

```
#include <stdio.h>
#include <stdlib.h>
main()
{
    char s[10];
    int qtd;
    float preco;
    FILE *fout;
    int i,q;
    fout=fopen ("c:/artigos.txt","w");
    printf ("quantos produtos quer registar?"); scanf("%d", &q);
   for (i=0; i<q; i++)
   {
      printf ("Nome:"); scanf("%s", s);
      printf ("Preço: "); scanf("%f", &preco);
      printf ("Quantidade: "); scanf("%d", &qtd);
      fprintf (fout,"%s %f %d\n", s, preco, qtd);
   }
```

```
    fclose(fout);
    fout=fopen("c:/artigos.txt","r");
    printf ("\nConteúdo do ficheiro\n");

    while (fscanf(fout,"%s %f %d\n", s, &preco, &qtd)!=EOF)
       printf ("Nome: %s  Preço: %.2f €  Quantidade: %d\n", s, preco, qtd);
}
```

EXERCÍCIOS:

1. Escreve um programa que leia uma frase para um ficheiro de texto, e que imprima no ecrã o seu conteúdo, convertido para maiúsculas.

2. Escreve um programa que crie um ficheiro de texto (de nome fornecido pelo utilizador) com 10 linhas, contendo cada uma delas a palavra "Olá".

3. O ficheiro de texto números deverá conter uma lista de 10 números reais, um por cada linha.
   Escreve um programa que leia para o ficheiro números os números referidos.

4. Escreve um programa que calcule e exiba a média dos números contidos no ficheiro números.

5. Escreve um programa que permita acrescentar produtos à lista contida no ficheiro do exercício resolvido 1.

6. Escreve um programa que crie um ficheiro de texto contendo 4 nomes de cidades e respetivas temperaturas médias, num dado dia.
   Deverá ainda exibir no ecrã a cidade com média de temperatura mais alta.