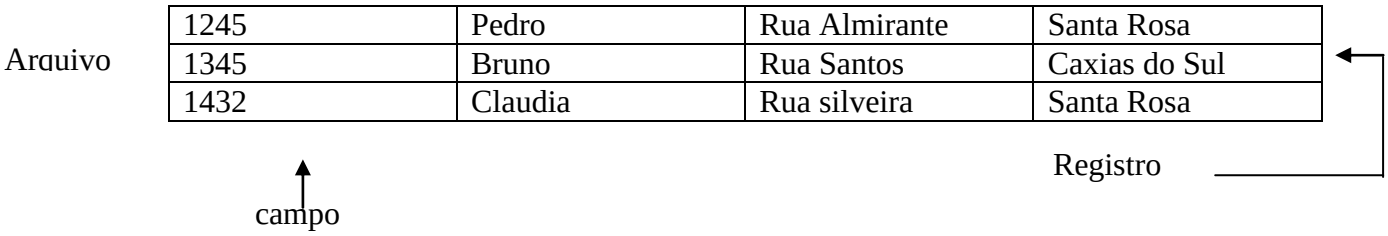
ARQUIVOS – FILES

Já foi estudado o conceito de tabelas em memória com a utilização de matrizes. Estes conceitos são a base para a utilização de um arquivo. As matrizes são manipuladas por meio de um índice de controle, enquanto os arquivos são manipulados por um ponteiro de registro.
A principal vantagem na utilização de arquivos está no fato de as informações armazenadas poderem ser utilizadas a qualquer momento. Outra vantagem encontrada na utilização de arquivos é o fato de poder armazenar um número maior de registros do que em uma tabela em memória, estando apenas limitado ao tamanho do meio físico utilizado para a sua gravação.

Arquivo: é um conjunto de registros (que pode ser apenas um registro) que, por sua vez, é um conjunto de campos (que pode ser apenas um campo), sendo cada campo o conjunto de informações nele contido.

O objetivo deste capítulo é dar noções de como trabalhar com arquivos.



Formas de acesso a um Arquivo

Os arquivos criados com a linguagem C podem ser acessados para leitura e escrita de duas formas: seqüencial e aleatório (acesso direto).

Seqüencial: ocorre quando o processo de gravação e leitura é feito de forma contínua, um após o outro a partir do primeiro registro, registro a registro, até localizar a primeira posição vazia após o último registro. O processo de leitura também ocorre de forma seqüencial. Se o registro a ser lido é o último, primeiro será necessário ler todos os registros que o antecedem. Este processo é considerado lento.

Aleatório: ocorre por meio da transferência de dados diretamente para qualquer posição do arquivo, sem que para isso, as informações anteriores precisem ser lidas (acesso seqüencial). O acesso aleatório a um arquivo pode ser feito de três formas diferentes, com relação as posicionamento do ponteiro dentro do arquivo: início do arquivo, fim do arquivo ou o posicionamento atual do ponteiro no arquivo.

Operações com arquivos

A manipulação de um arquivo em C ocorre coma definição do tipo FILE, que se caracteriza por ser uma estrutura formada por elementos do mesmo tipo dispostos de forma seqüencial, tendo como objetivo fazer a comunicação entre a memória principal e

a memória secundária, por meio do programa e do sistema operacional. Assim sendo, este tipo deve ser definido com a utilização da seguinte sintaxe:

FILE <*variável ponteiro>
Em que:
<*variável ponteiro> - definição de um ponteiro para a estrutura do tipo FILE

FILE deve ser escrito em **maiúsculo**, sendo esta um estrutura cujos elementos são informações que assinalam a condição de processamento e acesso a um arquivo. Esta estrutura esta presente na biblioteca **stdio.h**, portanto, é necessário indicar no programa a linha #include <stdio.h>, antes da função main().

Abertura e Fechamento de Arquivos

Para manipular um arquivo (ler ou escrever), é necessário executar duas operações básicas: abertura e fechamento, sendo conseguidas com a utilização das instruções: **fopen()** e **fclose()**, desde que o arquivo exista.

Para efetuar as operações de leitura e escrita de um arquivo, ele deve ser aberto e depois de utilizado precisa ser fechado. Para a abertura de um arquivo deve ser utilizada em um programa, a seguinte linha de código:

<variável ponteiro> = **fopen** (“nome do arquivo” , “tipo de abertura”);

Onde:

<variável ponteiro> - é a variável declarada como ponteiro do tipo FILE
Nome do arquivo - é o nome do arquivo a ser manipulado, deve ser valido no sistema operacional utilizado
Tipo de abertura - o tipo de abertura do arquivo (ver tabela abaixo)

Para o fechamento de um arquivo deve ser utilizada em um programa a seguinte linha de código:

fclose (<variável ponteiro>);

Onde:

<variável ponteiro> - é a mesma variável associada a função **fopen()**

O tipo de abertura de um arquivo é especificado por três códigos do tipo string, onde **r** – para leitura (read), letra **w** – para gravação (write) e letra **a** para adicionar dados (append).

A tabela abaixo mostra os valores de modo válidos:

Modo	Significado

"r"	Abre um arquivo texto existente para leitura, se não existir, irá ocorrer erro
"w"	Abre um novo arquivo texto para gravação de dados, se o arquivo já existir, irá destruir o arquivo existente.
"a"	Abre um arquivo texto para operações de anexação de dados, isto é, acrescenta dados no fim do arquivo ("append"), se o arquivo não existir, será criado um novo arquivo.
"rb"	Abre um arquivo binário existente para leitura
"wb"	Abre um novo arquivo binário para escrita
"ab"	Abre um arquivo binário e acrescenta dados binários no fim do arquivo
"r+b"	Abre um arquivo binário para leitura e escrita
"w+b"	Cria um arquivo binário para leitura e escrita
"a+b"	Acrescenta dados ou cria uma arquivo binário para leitura e escrita
"rt"	Abre um arquivo texto para leitura
"wt"	Cria um arquivo texto para escrita
"at"	Acrescenta dados no fim do arquivo texto
"r+t"	Abre um arquivo texto para leitura e escrita
"w+t"	Cria um arquivo texto para leitura e escrita
"a+t"	Acrescenta dados ou cria uma arquivo texto para leitura e escrita

Poderíamos então, para abrir um arquivo binário, escrever:

```
FILE *fp;
fp=fopen ("exemplo.bin","wb");
if (!fp)
        printf ("Erro na abertura do arquivo.");
```

A condição **!fp** testa se o arquivo foi aberto com sucesso porque no caso de um erro a função **fopen()** retorna um ponteiro nullo (**NULL**).

exit

Aqui abrimos um parênteses para explicar a função **exit()**

Esta função aborta a execução do programa. Pode ser chamada de qualquer ponto no programa e faz com que o programa termine e retorne, para o sistema operacional, o código_de_retorno. A convenção mais usada é que um programa retorne zero no caso de um término normal e retorne um número não nulo no caso de ter ocorrido um problema. A função **exit()** se torna importante em casos como alocação dinâmica e abertura de arquivos pois pode ser essencial que uma determinada memória seja alocada ou que um arquivo seja aberto. Poderíamos reescrever o exemplo da seção anterior usando agora o exit() para garantir que o programa não deixará de abrir o arquivo:

```
#include <stdio.h>
main (void)
```

```
{
FILE *fp;
...
fp=fopen ("exemplo.bin","wb");
if (!fp)
        {
        printf ("Erro na abertura do arquivo. Fim de
programa.");
        exit (1);
        }
...

}
```

Arquivos Textos

O arquivo do tipo texto possibilita a criação de registros armazenados com tamanhos diferentes (o que não ocorre com os outros tipos de arquivo).

Os arquivos do tipo texto estão capacitados a armazenar caracteres, palavras, frases e também dados numéricos. Os números, entretanto, serão armazenados como caracteres do tipo alfanumérico, e desta forma ocuparão muito mais espaço em disco do que ocupariam na memória de um computador. A solução para este detalhe é utilizar funções que manipulem os números em formato binário.

Vantagens:

- Facilidade em criar um arquivo textual, basta utilizar um editor de texto como bloco de notas ou o próprio editor do ambiente DevC++.
- A versatilidade de verificar o conteúdo de um arquivo textual, pois as informações estão codificadas como caracteres.

Desvantagens:

- A codificação do conteúdo do arquivo consome 1 byte por caracter, o que pode inviabilizar a utilização do arquivo para grandes volumes de dados. Ex: para armazenar os valores 10.00, 1000.00 e 10000000.00 como caracteres ASCII , serão necessários 5,7 e 10 bytes, respectivamente, ao passo que um valor do tipo float codificado em binário, independente de seu valor, consome sempre 8 bytes.
- O processamento do arquivo é lento, pois as informações codificadas como caracteres ASCII precisam ser convertidas para a codificação binária para serem armazenadas na memória do computador.
- A modificação de informações já armazenadas no arquivo requer a utilização de um arquivo temporário.

// CRIACAO DE ARQUIVO TEXTO

```
#include <stdio.h>
#include <stdlib.h>
main(void)
{
   FILE *ARQ;      // definicao do ponteiro para o arquivo
```

```
    ARQ=fopen("arquivo01.txt","a"); // ABERTURA DO ARQUIVO arquivo01.txt
    printf(" arquivo01.txt  criado no diretorio corrente - verifique");
    fclose(ARQ); // fechamento do arquivo
    system("pause");
  }
```

Verifique se o arquivo arquivo01.txt encontra-se no diretório corrente do seu programa.

A string “a” efetua a criação do arquivo, caso ele não exista. O arquivo01.txt é criado com um tamanho de zero byte.

Após a criação do arquivo, ele pode ser utilizado para a gravação dos dados. Verifique exemplo abaixo.

// grava palavra em arquivo texto

```
#include <stdio.h>// utilizada para arquivos FILE
#include <stdlib.h>
main(void)
{
  FILE *ARQ;
  char palavra[20];
  ARQ=fopen("arquivo01.txt","w");
  printf(" Escreva uma palavra: ");
  scanf("%s",&palavra);
  fprintf(ARQ, "%s", palavra); // faz a saida para disco, para dentro do arquivo ARQ
  fclose(ARQ);
  system("pause");
}
```

Leitura de uma arquivo

```
// leitura de um arquivo texto
#include <stdio.h>// utilizada para arquivos FILE
#include <stdlib.h>
main(void)
{
    FILE *ARQ;
    char palavra[20];
    ARQ=fopen("arquivo01.txt","r");  // abre somente para leitura
    fscanf(ARQ,"%s",&palavra); // faz a leitura do arquivo em disco, transferindo o
conteudo para a palavra, somente uma palavra até encontrar espaço.//fscanf
    printf(" A palavra = %s ",palavra); // imprime no video o conteudo da palavra
    fclose(ARQ);
    system("pause");
}
```

// Cria e grava frase caractere a caractere

```
#include <stdio.h>
#include <stdlib.h>
main(void)
{
  FILE *ARQ;
```

```
  char letra;
  ARQ=fopen("frase01.txt","w");  // cria o arquivo frase01.txt
  printf("Escreva a frase desejada :\n\n ");
  while ((letra=getchar()) !='\n')
        putc(letra,ARQ); // enquanto a tecla for diferente de enter grava
                                                        //caracter por caracter
  fclose(ARQ);
  system("pause");
}
```

// le frase caractere por caractere

```
#include <stdio.h>
#include <stdlib.h>
main(void)
{
  FILE *ARQ;
  char letra;
  ARQ=fopen("frase01.txt","r");  // abre o arquivo frase01.txt para leitura
  printf("Frase do arquivo =  ");
  while ((letra=fgetc(ARQ)) !=EOF) // Enquanto nao encontra o final do arquivo EOF
        {
                printf("%c", letra);
        }
  fclose(ARQ);     system("pause");
}

// fgetc - efetua a leitura de apenas um caractere armazenado
```

// Cadastro e impressão de nomes em arquivo texto

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <ctype.h>
FILE *ARQ;

//---------------------------------
void criar();
void cadastrar();
char menu();
//=========================================
char menu()
        {
          printf(" \n1 - Criar");
          printf(" \n2 - Cadastrar");
          printf(" \n3 - Relatorio");
          printf(" \n4 - Sair");
          printf("\nEntre com a opcao : ");

          return (getche());
```

```
        }
//=============================================
void criar()
        {
         system("cls");
         printf("\nCriacao do arquivo\n");
         ARQ=fopen("nomes.txt","w");
         printf("\n========= Arquivo foi criado ========= \n");
         system("pause");
         fclose(ARQ);

        }
//=============================================
void cadastrar()
        {

         char nome[41],op;
         system("cls");
         printf("\nCadastrar registros\n");
         ARQ=fopen("nomes.txt","a");
         op='S';
         while (op =='S' )
         {
           printf("\nNome = ");
           fgets(nome,41,stdin);//le do teclado 41 caracteres
           fputs(nome,ARQ);//grava no arquivo
           printf(" \nContinuar s/n ?");
           op=toupper(getche());
         }
         fclose(ARQ);
        }
 //=============================================
void relatorio()
        {char nome[41];
         system("cls");
         printf("\n==========Relatorio de Registros======\n");
         ARQ=fopen("nomes.txt","r");
         if (ARQ == NULL)
         {  printf ("\nerro ao abrir o arquivo ");
            return;
          }
         printf("\n_____________________________________________________\n");
         while (fgets(nome,41,ARQ)!=NULL)
         {
           printf("%s",nome);
         }
          printf("\n_____________________________________________________\n");

         printf("\n\n\n\n\n");
         system("pause");
```

```
        fclose(ARQ);

       }
//================================================
main(void)
       {
        char op;
        while(1)
        {
          system("cls");
          op=menu();
          switch(op)
          {
            case '1': criar(); break;
            case '2': cadastrar();break;
            case '3': relatorio();break;
            case '4': exit(0);
          }

        }
        system("pause");
       }

//=============== fim ====================
```

Arquivos Binários

Quando temos um arquivo binário, podemos armazenar números e caracteres, pois o armazenamento se dá através do sistema de numeração binária.

Arquivo binário é um conjunto de Registro, colocados seqüencialmente, como se fosse um fichário comum, onde cada ficha, registro, é composto por campos. É interessante para o tratamento de grandes volumes de dados que precisam sofrer modificações freqüentes.

Declaração de um Arquivo Binário

```
Algoritmo Forma_Geral
        Tipo
                Tipo_Arquivo = Arquivo de Tipo_Existente
        Variáveis
                Arq - Tipo_Arquivo
        Inicio
                Associe(Arq,"Nome_Arquivo.Extensão")
        Fim
```

Tipo_Arquivo é o nome do novo tipo que você acaba de criar, você escolhe o nome conforme o problema.

Tipo_Existente um arquivo pode conter o tipo que você deseja, assim você poderá ter um arquivo que armazene inteiros, caracter ou um registro.

Arq será a variável do tipo arquivo, e sempre que você quiser fazer referencia ao arquivo será através desta variável. Você escolhe o nome para esta variável da mesma forma que você escolhia para as demais variáveis.

Dentro do programa principal existe um comando, Associe, este comando é que vai fazer a conecção entre a variável e o arquivo que está no disco. O parâmetro Nome_Arquivo.Extensão é o nome do arquivo em disco e poderia ser por exemplo estoque.dat. Da maneira que está declarado, este arquivo está no diretório corrente, o que nada impede de você redirecionar o caminho deste arquivo, por exemplo: c:\tp\meus\estoque.xxx

O comando Associe, deve anteceder todos os demais comandos que veremos a seguir.

Apontador

Como vimos, um arquivo binário é um conjunto de registro disposto seqüencialmente, de forma que o computador tem que ter o controle de qual ficha, registro, está mexendo no momento. Para fazer este controle a linguagem que você for trabalhar terá um Apontador, o qual aponta sempre para o registro corrente, isto é, para o registro que se está manipulando. Suponha que você tenha em disco um arquivo conforme a figura abaixo:

1	'Lápis'	30	1.00
2	'Borracha'	40	1.50
3	'Caneta'	25	1.20
4	'Apontador'	35	0.90

Para os arquivos binários os registros devem ser definidos por uma estrutura de dados conveniente. Por exemplo:

```
typedef struct cliente
{
  int conta;
 char nome[30];
float limite;
float saldo;
} cliente;
```

Portanto cada registro do arquivo irá corresponder a 40 byte, pois serão consumidos:

4 bytes par ao campo nconta (int)
20 bytes para o campo nome (string)
8 bytes para o campo limite (float)
8 bytes para o campo saldo (float)

Ou seja: sizeof(cliente)=40 BYTES

nconta	nome	saldo	limite

4 bytes 20 bytes 8 bytes 8 bytes

Gravação e leitura em arquivo binários

As operações de leitura e escrita em arquivo binários são realizadas, respectivamente, pelas funções **fread()** e **fwrite()**. Essas funções requerem os seguintes parâmetros:

buffer: endereço de memória para onde a informação será lida (ou o endereço de memória de onde provem a informação a ser escrita);
tamanho: tamanho (em bytes) do bloco de memória a ser lido ou escrito;
contador: especifica o número de itens a serem acessados (lidos ou escritos);
variável/ponteiro: é a variável do tipo ponteiro de arquivo

fwrite(buffer, tamanho, contador, ponteiro)
fread(buffer, tamanho,contador, ponteiro)

A função **sizeof()** retorna o tamanho em bytes da expressão nela indicada.

Exemplo de gravação de arquivos binários

```
// GRAVACAO de um arquivo BINARIO
// entrada de 5 valores inteiros em uma matriz e gravada de uma só vez no arquivo binário.

#include <stdio.h>// utilizada para arquivos FILE
#include <stdlib.h>
main(void)
{
        FILE *ARQ;
        int i,a[5];
        ARQ=fopen("matriz.bin","wb"); // wb se existe o arquivo ele cria novamente
        i=0;
        while (i<=4)
        {
                printf(" Digite o %d elemento : ",i+1);
                fflush(stdin);
                scanf("%d",&a[i]);
                i++;
        }
        fwrite(a,sizeof(a),1,ARQ); // grava no arquivo de uma so vez a matriz
        fclose(ARQ);
        system("pause");
}
```

Exemplo de leitura de arquivos binários

```
// LEITURA de um arquivo BINARIO, COM MATRIZ

#include <stdio.h>// utilizada para arquivos FILE
#include <stdlib.h>
main(void)
{
        FILE *ARQ;
        int i,a[5];
        ARQ=fopen("matriz.bin","rb");
        fread(a,sizeof(a),1,ARQ);
        i=0;
        while (i<=4)
        {
                printf(" Elemento %d da matriz: %d\n",i+1,a[i]);
                i++;
        }
        fclose(ARQ);
        system("pause");
}
```

As funções **fread()** e **fwrite()** podem efetuar a leitura de uma só vez de todos os dados de um arquivo e transferi-los para a memória. Uma das suas maiores aplicações é o fato de elas serem usadas para ler e escrever tipos de dados definidos pelo programador, como é o caso das **struct**.

O acesso às informações contidas em um arquivo é, normalmente feito registro a registro. Assim quando é necessário recuperar informações do arquivo ou quando é necessário gravar em um arquivo a informação envolvida refere-se ao conteúdo de um **registro**.

Exercícios

1. Ler cinco valores de flutuantes em uma matriz B de uma dimensão. Após a leitura os valores devem ser armazenados em um arquivo binário. Efetuar a leitura do arquivo binário e calcular a média dos valores, bem como mostrar o conteúdo do arquivo e a media dos valores como relatório.
2. Elaborar um programa que efetue a leitura de duas matrizes A e B de dimensões 2x2. Construa uma matriz C que será a soma de elemento a elemento da matriz A e B. Gravar a matriz C em arquivo binário. Em seguida o programa deverá ler todos os dados do arquivo e transferi-los para a matriz D que será então apresentada na tela.

Arquivo de Acesso Direto

Todas as funções de acesso a arquivos vistos até o momento acessam os dados de um arquivo de forma seqüencial a partir de um indicador de posição corrente. Para que seja possível efetuar um acesso direto a uma determinada informação de um arquivo, utiliza-se a função **fseek(),** que retornará 0 (zero) quando completar com sucesso a operação d posicionamento, ou retornará -1 (um negativo) quando ocorrer algum erro na localização da informação.

A função fseek() é utilizada para posicionar o ponteiro de acesso aos registros em uma posição qualquer da seqüência de bytes que constitui o arquivo.

A função **fseek()** possui a seguinte sintaxe:

fseek(ponteiro, bytes, origem)

Em que:
Ponteiro: é um ponteiro para o arquivo;

Exemplo

fseek(ARQ, -sizeof(cliente),SEEK_CUR);

fseek(ARQ, (n-1) * sizeof(cliente), SEEK_SET); onde n é o número do registro

Bytes: são os bytes de deslocamento em relação a origem, a ser dado ao ponteiro de acessos registros; um valor positivo significa deslocamento para frente; um valor negativo significa um deslocamento para trás;
Origem: valor inteiro que representa a posição do arquivo.. a posição a partir da qual será efetuado o deslocamento; as seguintes constantes podem ser usadas para identificar essa posição:

SEEK_SET : indica o **início** do arquivo ou **0(zero)**;
SEEK_CUT : indica a **posição atual** do ponteiro de acesso aos registros ou **1 (um);**
SEEK_END : indica o **fim** do arquivo ou **2 (dois)**.

EXEMPLO

```
// Cadastro , impressao de nomes e IDADE em arquivo BINARIOS
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
FILE *ARQ;
typedef struct alunos
{
   int cod;
   char nome[20];
   int idade;
};
alunos alu;
//--------------------------------
void criar();
void cadastrar();
char menu();
//=========================================
char menu()
{
  printf(" \n1 - Criar");
  printf(" \n2 - Cadastrar");
```

```
  printf(" \n3 - Relatorio");
  printf(" \n4 - Sair");
  printf("\nEntre com a opcao : ");

  return (getche());
 }
//=============================================
void criar()
{
 system("cls");
 printf("\nCriacao do arquivo\n");
 ARQ=fopen("nomesbin.bin","wb");
 printf("\n========= Arquivo foi criado ========= \n");
 system("pause");
 fclose(ARQ);

}
//=============================================
void cadastrar()
{
  int codigo;
 char op;
 system("cls");
 printf("\nCadastrar registros\n");
 ARQ=fopen("nomesbin.bin","ab");
 op='S';
 system("cls");
 printf("\nCodigo = ");
 fflush(stdin);
 scanf("%d",&codigo);
 while(codigo!=0)
 {

    alu.cod=codigo;
    printf("\nNome = ");
    fflush(stdin);
    gets(alu.nome);
    printf("\nIdade = ");
    scanf("%d",&alu.idade);
    fwrite(&alu,sizeof(alunos),1,ARQ);
    system("cls");
    printf("\nCodigo = ");
    fflush(stdin);
    scanf("%d",&codigo);

 }
 fclose(ARQ);
}
//==================================
```

```
void relatorio()
{
 system("cls");
 printf("\n=========Relatorio de Registros======\n");
 ARQ=fopen("nomesbin.bin","rb");
 if (ARQ == NULL)
 {  printf ("\nerro ao abrir o arquivo ");
    return;
  }
 printf("\n_____________________________________________________\n");
 fread(&alu,sizeof(alunos),1,ARQ);
 while (!feof(ARQ))
 {
   printf("%03d  |  %20s  |  %d\n",alu.cod,alu.nome,alu.idade);
   fread(&alu,sizeof(alunos),1,ARQ);
 }
 printf("\n_____________________________________________________\n");
 printf("\n\n\n\n\n");
 system("pause");
 fclose(ARQ);

}

main(void)
{
 char op;
 while(1)
 {
    system("cls");
    op=menu();
    switch(op)
    {
      case '1': criar(); break;
      case '2': cadastrar();break;
      case '3': relatorio();break;
      case '4': exit(0);
    }

 }
 system("pause");
}
```