Linguagem C

Armazenamento de Dados em Arquivos - Continuação

Arquivos Binários

- Comandos para gravação e leitura de arquivos (revisão):
  - `FILE *arquivo;`
  - `arquivo = fopen("nome", "modo");`
  - `fwrite (&variavel, sizeof (tipo_var), 1, arquivo);`
  - `rewind (arquivo);`
  - `fread (&variavel, sizeof ( tipo_var), 1, arquivo);`
  - `fclose (arquivo);`
  - `feof(arquivo);`

Arquivos (Revisão)

- Comandos para abertura de arquivos:
  - Definição de variáveis tipo “Arquivo”:

```
FILE *arquivo;
```

  - Abertura (e fechamento) de arquivos:

```
arquivo = fopen(“nome”, “modo”);
if(arquivo!=0) fclose(arquivo);
```

Onde:

nome = nome do arquivo

modo = tipo do arquivo (ascii ou binário), e objetivo de uso (leitura, escrita, anexação)

Obs: o comando fopen retorna um número inteiro: um número maior que zero significa que a abertura foi feita corretamente, 0(zero) indica erro de abertura do arquivo;

Nunca tente fechar um arquivo que não foi aberto!!!

Modos de Arquivos (Revisão)

Modos de utilização de arquivos	
Modo	**Significado**
r	Abre um arquivo texto para leitura.
w	Cria um arquivo texto para escrita.
a	Anexa a um arquivo-texto.
rb	Abre um arquivo binário para leitura.
wb	Cria um arquivo binário para escrita.
ab	Anexa um arquivo binário.
r+	Abre um arquivo-texto para leitura/escrita.
w+	Cria um arquivo-texto para leitura/escrita.
a+	Anexa ou cria um arquivo-texto para leitura/escrita.
r+b	Abre um arquivo binário para leitura/escrita.
w+b	Cria um arquivo binário para leitura/escrita.
a+b	Anexa a um arquivo binário para leitura/escrita.

Arquivos Binários - Continuação

- Os arquivos binários podem armazenar mais dados do que cabe na memória volátil de um computador;
- Para realizar a leitura/escrita de um registro específico, deve-se posicionar o ponteiro para arquivo de forma adequada;
- Comando para posicionamento do “ponteiro de leitura/escrita” em arquivos binários:

  - **`fseek(FILE *ponteiro_do_arquivo, long int posição_em_bytes, int modo_movimentação)`**

Arquivos Binários - Continuação

- A sintaxe do comando fseek é:

```
fseek(ponteiro_do_arquivo, posicao_em_bytes,
    modo_movimentacao);
```

- O ponteiro_do_arquivo é aquele de quando abrimos o arquivo (Variável FILE *).
- A posição, em bytes, para onde moveremos o ponteiro interno do arquivo é um valor long int, então ao declarar alguma variável para isso devemos usar o tipo long, e ao usar números diretamente é recomendável usar o molde (long), para caso do valor passar o tamanho máximo suportado para uma variável int.

Arquivos Binários - Continuação

- O modo de movimentação controla como queremos mover o ponteiro. São três modos:
- **SEEK_SET (constante de valor 0)** - movimenta para a posição indicada (começando a contar do zero, que representa o primeiro byte do arquivo).
- **SEEK_END (constante de valor 1)**- movimenta para a posição indicada, começando a contar do final do arquivo. Neste caso, o zero representa a posição imediatamente posterior ao último byte do arquivo.
- **SEEK_CUR (constante de valor 2)**- movimenta a partir da posição atual. Neste caso podemos colocar números negativos, que significa que queremos retroceder com o ponteiro do arquivo. Lembrem-se que qualquer operação de leitura ou gravação em um arquivo move o ponteiro interno deste arquivo, o mesmo números de bytes da leitura (ou gravação).
- Quando abrimos um arquivo no modo append (a), o ponteiro começa na posição zero, em relação ao final do arquivo. Seria como fazer fseek usando o SEEK_END.

Arquivos Binários - Continuação

- Exemplos:

```
int result;
FILE *fp;
fp = fopen("arquivo.bin", "r+b");
if(fp!=0)
  { printf("Erro na abertura do arquivo\n");
    exit(fp);}
result = fseek(fp,0,SEEK_SET);
//equivale ao rewind(fp);
if(result!=0)
    printf("Erro no posicionamento!\n");
```

Arquivos Binários - Continuação

- Lendo a posição do ponteiro de leitura/gravação do arquivo :
- O comando ftell informa a posição atual do ponteiro interno do arquivo. Exemplo:

```
long posicao;
posicao=ftell(ponteiro_arq)/sizeof(struct cliente);
printf ("O ponteiro interno do arquivo esta' apontando
  para o %ld° registro.\n", posicao + 1);
```

- Este comando pode ser útil também para ver qual é o tamanho de um arquivo.
- Para isso, posicione o ponteiro do arquivo no final do arquivo (usando SEEK_END), e depois leia a posição.
- O número retornado é o tamanho do arquivo, em bytes.

Arquivos Texto

- Comandos para gravação e leitura de arquivos:
  - **`fgetc();`**
  - **`fputc();`**
  - **`fscanf();`**
  - **`fprintf();`**

Arquivos Texto (ou Arquivos Sequenciais)

- Comandos para gravação e leitura de arquivos texto:
  - `fgetc(FILE *fp);`
  - `fputc(char C, FILE *fp);`
  - `char *fgets(FILE *fp,char *var);`
  - `char *fputs(FILE *fp,char *var);`
  - `fscanf(FILE *fp,“string_formato”,vars);`
  - `fprintf(FILE *fp,“string_formato”,vars);`

Arquivos Texto
(ou Arquivos Sequenciais)

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
FILE *fp; char ch; char* nome;
printf ( "Informe o nome do arquivo a ser lido: \n“ );
gets ( nome );
fp = fopen ( nome, "r" );
if ( NULL == fp)
{
 printf ( "O arquivo não pode ser aberto. \n" );
 system ( "Pause" );
 exit (1);
}
```

Arquivos Texto
(ou Arquivos Sequenciais)

```
ch = fgetc ( fp );
while ( ch != EOF )
{
 putchar( ch ); /* Imprime na tela */
 ch = fgetc ( fp );
}
printf ("\n");
system ("Pause");
return 0;
}
```

Arquivos Texto
(ou Arquivos Sequenciais)

```
   //fputc(char C, FILE *fp);
#include <stdio.h>
#include <stdlib.h>
int main()
{
FILE *fp; char ch;
fp = fopen ( “teste.txt", "w" );
if ( fp == NULL )
{
printf ( "O arquivo não pode ser aberto. \n" );
system ( "Pause" );
exit (1);
}
```

Arquivos Texto
(ou Arquivos Sequenciais)

```
printf ( "Programa para gravar caracteres em um arquivo
   chamado teste.txt." );
printf ( "\n \n" );
printf ( "Digite os caracteres: \n" );
do
{
 ch = getchar();
 fputc ( ch, fp );
} while ( ch != '\n' );
fclose ( fp );
system ( "Pause" );
return 0;
}
```

Arquivos Texto
(ou Arquivos Sequenciais)

- **Leitura com fgets()**
- A função **fgets()** lê uma linha inteira de uma vez.
- Exemplo:

```
result = fgets(Linha, 100, arq);
// o 'fgets' lê até 99 caracteres ou até o '\n`
```

- Se a função for executada com sucesso, **fgets** retorna o endereço da string lida, caso contrário retorna NULL.

Arquivos Texto
(ou Arquivos Sequenciais)

```
#include <stdio.h>

void main(){
   FILE *arq;
   char Linha[100];
   char *result;
   int i;

arq = fopen("ArqTeste.txt", "rt");

if (arq == NULL) // Se houve erro na abertura
   {printf("Problemas na abertura do arquivo\n");
   return;}
```

Arquivos Texto
(ou Arquivos Sequenciais)

```
i = 1;
while (!feof(arq))
   {
   result = fgets(Linha, 100, arq);
   // o 'fgets' lê até 99 caracteres ou até o '\n'

    if (result) // Se foi possível ler
      printf("Linha %d : %s",i,Linha);
    i++;
   }
fclose(arq);
}
```

Arquivos Texto
(ou Arquivos Sequenciais)

- **Leitura com fscanf**
- A função **fscanf()** funciona como a função **scanf()**, porém, ao invés de ler os dados de teclado, estes dados são lidos de arquivo.
- Exemplo:

```
int i, result;
float x;
result = fscanf(arq, "%d%f", &i, &x);
```

- Se result for igual à constante EOF, não há mais dados para serem lidos.

Arquivos Texto
(ou Arquivos Sequenciais)

- **Gravação**

  Para gravação de arquivos texto usa-se as funções **fputs** e **fprintf**

- **Exemplo:**

```
result = fputs(Str, arq);
```

Se a função NÃO for executada com sucesso, **fputs** retorna a constante **EOF**.

Arquivos Texto
(ou Arquivos Sequenciais)

```
char Str[100];

FILE *arq;

arq = fopen("ArqGrav.txt", "wt"); //Cria um arquivo texto
  para gravação

if (arq == NULL) // Se não conseguiu criar
  {printf("Problemas na CRIACAO do arquivo\n");
  return;}

strcpy(Str,"Linha de teste");

result = fputs(Str, arq);

if (result == EOF)
  printf("Erro na Gravacao\n");

fclose(arq);
```

Arquivos Texto
(ou Arquivos Sequenciais)

- **Exemplo de fprintf:**

```
result = fprintf(arq,"Linha %d\n",i);
```

Se a função **fprintf** for executada com sucesso, devolve o número de caracteres gravados. Se a função NÃO for executada com sucesso, retorna a constante **EOF**.

Arquivos Texto
(ou Arquivos Sequenciais)

```
#include <stdio.h>

void main()

{
   FILE *arq; int i; int result;
   arq = fopen("ArqGrav.txt", "wt");

    if (arq == NULL) // Se nao conseguiu criar
     { printf("Problemas na CRIACAO do arquivo\n"); return;}

    for (i = 0; i<10;i++)
     {
     // A funcao 'fprintf' devolve o número de bytes gravados
     // ou EOF se houve erro na gravação
     result = fprintf(arq,"Linha %d\n",i);
     if (result == EOF)
        printf("Erro na Gravacao\n");
     }

fclose(arq);

}
```