Arquivos em linguagem C

Laboratório de Programação

Luciano Antunes

Manipulação de Arquivo em C

- Existem dois tipos possíveis de acesso a arquivos na linguagem C : sequencial (lendo um registro após o outro) e aleatório (posicionando-se diretamente num determinado registro);
- Os arquivos em C são denominados STREAM.

Manipulação de Arquivo em C
Funções em linguagem C

Função	Ação
fopen()	Abre um arquivo
Fclose()	Fecha um arquivo
putc() e fputc()	Escrve um caractere em um arquivo
getc() e fgetc()	Lê um caractere de um arquivo
fseek()	Posiciona em um registro de um arquivo
fprintf()	Efetua a impressão formatada em um arquivo
fscanf()	Elefetua a leitura formatada em um arquivo
feof()	Verifica o final de um arquivo
fwrite()	Escreve tipos maiores que 1 byte em um arquivo
fread()	Lê tipos maiores que 1 byte de um arquivo

Manipulação de Arquivo em C declaração

- Esta definição também está no arquivo stdio.h, e um ponteiro de arquivo pode ser declarado da seguinte maneira:

```
FILE *arquivo;
```

Manipulação de Arquivo em C
abertura de arquivo

- A função que abre um arquivo em C é a função ***fopen()***, que devolve o valor NULL (nulo) ou um ponteiro associado ao arquivo, devendo ser passado para função o nome físico do arquivo e o modo como este arquivo deve ser aberto

```
arquivo=fopen ("texto.txt","w");
```

abertura de arquivo

- De acordo com a instrução anterior, está sendo aberto um arquivo de nome “texto.txt”, habilitado apenas para escrita (w-write)
- Por exemplo, pode-se codificar a instrução de abertura de arquivo da seguinte maneira:

```
if ((Arquivo = fopen("texto.txt","w")) == NULL) {
printf("\n Arquivo TEXTO.TXT não pode ser aberto : TECLE ALGO");
getch();
}
```

Fechando um arquivo

- Para o esvaziamento da memória de um arquivo é utilizada a função ***fclose()***, que associa-se diretamente ao nome lógico do arquivo:

fclose (Arquivo);

Gravação e leitura de dados

- ***putc()*** ou ***fputc()***: Grava um único caracter no arquivo
- ***fprintf()*** : Grava dados formatados no arquivo, de acordo com o tipo de dados (float, int, ...). Similar ao printf, porém ao invés de imprimir na tela, grava em arquivo
- ***fwrite()*** : Grava um conjunto de dados heterogêneos (struct) no arquivo
- ***fscanf()***: retorna a quantidade variáveis lidas com sucesso

Gravação e leitura de dados

- Grava o conteúdo da variável caracter no arquivo
  - putc (caracter, arquivo);
- Grava dados formatados no arquivo, de acordo com o tipo de dados (float, int, ...)
  - fprintf(arquivo, "formatos", var1, var2 ...);
- Grava um conjunto de dados heterogêneos (struct) no arquivo
  - fwrite (buffer, tamanhoembytes, quantidade, ponteirodearquivo);
- Retorna a quantidade variáveis lidas com sucesso
  - fscanf(arquivo, "formatos", &var1, &var2 ...);

Exemplo com fscanf

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include <cstdlib>
#include <iostream>
main( ) {
FILE *arq_cliente;
char var_sexo, var_arquivo_aux, nomecli[50];
int cd_cli, vl_idade, indice = 0;
float vl_limite_credito;
arq_cliente = fopen("CLIENTE.TXT", "r");
if (arq_cliente == NULL) {
        printf("\nArquivo CLIENTE.TXT nao pode ser aberto.");
        getch();
}else {
        var_arquivo_aux = fscanf(arq_cliente, "%d %c %s %d %f",&cd_cli, &var_sexo        ,&nomecli, &vl_idade, &vl_limite_credito);
while (var_arquivo_aux != EOF) {
indice = indice + 1;
printf("\n Dados do %d § cliente : \n ", indice);
printf("\n Codigo do Cliente...: %d Sexo..: %c", cd_cli, var_sexo);
printf("\n Nome do Cliente.....: %s ", nomecli);
printf("\n Idade.....: %d Credito....: %8.2f", vl_idade, vl_limite_credito);
printf("\n---------------------------------------------------------- [Tecle algo] !");
getch();
var_arquivo_aux = fscanf(arq_cliente, "%d %c %s %d %f",&cd_cli, &var_sexo
,&nomecli, &vl_idade, &vl_limite_credito);
}
fclose (arq_cliente);
printf("\n *** FIM : [Tecle algo] !");
getch();
}
}
```

Leitura e gravação de estruturas

- Além da manipulação de arquivos do tipo texto, pode-se ler e escrever estruturas maiores que 1 byte, usando as funções ***fread()*** e ***fwrite()***, conforme as sintaxes a seguir:

fread (**buffer**, tamanhoembytes, quantidade, ponteirodearquivo)

fwrite(**buffer**, tamanhoembytes, quantidade, ponteirodearquivo)

Leitura e gravação de estruturas

- O ***buffer*** é um endereço de memória da estrutura de onde deve ser lido ou onde devem ser escritos os valores (fread() e fwrite(), respectivamente)
- O ***tamanhoembytes*** é um valor numérico que define o número de bytes da estrutura que deve ser lida/escrita
- A ***quantidade*** é o número de estruturas que devem ser lidas ou escritas em cada processo de fread ou fwrite
- O ***ponteirodearquivo*** é o ponteiro do arquivo de onde deve ser lida ou escrita uma estrutura

Leitura e gravação de estruturas

- A função ***sizeof*** retorna a quantidade de bytes de um determinado tipo ou variável
- Tal função é importante para que o programa de manipulação de arquivos possa saber se ainda existem registros para serem lidos
- Por exemplo, enquanto o retorno da instrução abaixo for igual a 1, o programa continua lendo registros:

```
retorno = fread(&Vcli, sizeof(struct Tcliente), 1, cliente);
```

Posição do registro

- Por meio da linguagem C não é possível saber qual é a posição de cada registro no arquivo
- Em outras linguagens, a movimentação em registros é feita por meio de funções que fazem a leitura da linha do registro
- Em C esta posição pode ser calculada pelo tamanho do registro

Posição do registro

- Não é possível, como em outras linguagens, pedir para que se posicione no segundo, terceiro ou último registro
- Para isso, programador em C deve saber o tamanho em bytes de cada registro, e posicionar-se de acordo com este tamanho.
- A função ***seek()***, apresentada logo abaixo movimenta-se de byte em byte

seek(<referência_ao_arquivo>, <n>, <modo>);

Posição do registro

- O parâmetro <n> indica quantos bytes devem ser avançados ou retrocedidos
- O exemplo a seguir posiciona-se no quarto registro do arquivo de cliente
- Observe que é utilizada uma função auxiliar – a função ***sizeof()*** que indica quantos bytes possui o registro a ser inserido (ou a estrutura definida para o registro)

```
fseek(Arquivo_de_Cliente, 4 * sizeof(Cliente), SEEK_SET);
```

Posição do registro

Outros parâmetros usados pela função seek()

- **SEEK_SET** - Parte do início do arquivo e avança <n> bytes
- **SEEK_END** - Parte do final do arquivo e retrocede <n> bytes
- **SEEK_CUR** - Parte do local atual e avança <n> bytes