Lógica de programação

Arquivos

Walisson Pereira

walisson_pereira@uvanet.br
Universidade Estadual Vale do Acaraú

Roteiro

Antes de começar

Antes de começar

Antes de começar, vamos conhecer a função **sizeof**.

Exemplo de uso:

```
1 int tamanho = sizeof(int);
```

A função retorna o tamanho em bytes de uma variável ou um tipo.

Antes de começar

Exemplo:

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct {
  float largura, altura;
} t_retangulo;

int main () {
  int tamanho_inteiro = sizeof(int);
  printf ("int = %d bytes\n", tamanho_inteiro);
  int tamanho = sizeof(t_retangulo);
  printf ("t_retangulo = %d bytes\n", tamanho);
}
```

Antes de começar

Trabalharemos com arquivos já considerando que ele **já foi criado**.

Para isto, basta abrir um editor de texto (bloco de notas, gedit, etc.) e salvar um arquivo vazio.

No linux, é possível criar um arquivo vazio com o seguinte **comando no terminal**:

```
1 touch binario.dat
```

Antes de começar

A função **exit** é usada para encerrar o programa a qualquer momento. Para utilizá-la, é necessário incluir a biblioteca **stdlib.h**.

```
1 exit(0); // finaliza o programa indicando que deu tudo certo
2 exit(1); // finaliza o programa indicando que algo deu errado
```

Você pode colocar qualquer valor diferente de zero para indicar que o programa foi encerrado com algum problema. Você pode passar qualquer número no parâmetro. Cada número pode significar alguma mensagem.

Introdução

Introdução

Estruturas de dados manipuladas fora do ambiente do programa são conhecidas como **arquivos**.

Considera-se como ambiente do programa a memória principal, onde nem sempre é possível ou conveniente manter certas estruturas de dados.

Um **arquivo**, que é armazenado em um dispositivo de memória secundária, como discos, por exemplo, pode ser lido ou escrito por um programa e é formatado por uma coleção de caracteres (arquivo de texto) ou bytes (arquivo binário).

Arquivos

Arquivos

Um **arquivo** é uma área em disco na qual podemos ler e gravar informação.

Esta área é **gerenciada pelo sistema operacional** do computador, ou seja, não precisamos nos preocupar em como esse espaço é organizado em disco.

Do ponto de vista de programas, um arquivo é acessado por nome e é onde podemos ler e escrever linhas de texto ou dados em geral.

Arquivos

Para acessar um arquivo, precisamos **abri-lo**.

Durante a abertura, informamos o nome do arquivo, com o nome do diretório em que ele se encontra (se necessário) e que operações queremos realizar: leitura e/ou escrita.

Após o uso do arquivo, você deve sempre **fechá-lo.**

Arquivos

Na linguagem C, o tipo **FILE**, definido na biblioteca **stdio.h**, é usado para tratar arquivos. Um apontador para **FILE** deve ser declarado sempre que uma variável arquivo for utilizada no programa.

Exemplo de declaração de uma variável do tipo apontador para arquivo denominada **arq**.

```
1 FILE *arq;
```

Arquivos

Os apontadores para **FILE** não fazem referência ao arquivo físico em si, mas a uma região da memória principal que possui as informações necessárias para leitura e escrita no arquivo.

Isso implica que não é possível manipular o arquivo diretamente, por exemplo, abrir um arquivo e tentar armazenar o conteúdo do apontador **FILE*** em outro arquivo.

O programador deve associar um arquivo físico a variável **FILE***, e depois usá-la como argumento para as funções que operam sobre arquivos.

Operações sobre arquivos

Operações sobre arquivos

Arquivos demandam algumas operações, como **abertura** e **fechamento**.

Os valores já armazenados nos arquivos podem ser lidos e novos valores podem ser escritos.

As operações que realizam essas tarefas são fornecidas pela linguagem de programação em vários formatos.

Abertura

Após a declaração de uma variável do tipo arquivo, ela **deve ser associada a um nome de arquivo**, o qual identifica um arquivo de fato, com correspondência na memória secundária.

Se o arquivo com o nome especificado já existir, o programador terá acesso ao seu conteúdo e poderá lê-lo e alterá-lo.

A medida que uma leitura ou escrita é realizada, um apontador chamado **indicador de posição de arquivo** é incrementado automaticamente pelo sistema operacional.

Abertura

Formato da função para abertura de arquivos:

```
1 FILE* fopen (char nome_arquivo[], char modo[]);
```

Exemplo:

```
1 FILE *arq1, *arq2;
2 arq1 = fopen ("arquivo.dat", "ab");
3 arq2 = fopen ("arquivo.dat", "rb");
4 arq3 = fopen ("arquivo.dat", "rb+");
```

Abertura

A função **fopen** utiliza os parâmetros nome e modo. O nome é o nome do arquivo em si. O modo indica as operações que vamos realizar.

Modo	**Operações**
ab	Anexa novos dados a um arquivo binário.
rb	Abre um arquivo binário onde poderão ser realizadas apenas operações de leitura.
rb+	Abre um arquivo binário onde poderão ser realizadas operações de leitura e de escrita.

Existem outros modos, mas vamos trabalhar só com estes.

Abertura

Um ponto importante a ser ressaltado sobre a função **fopen** é a avaliação de seu valor de retorno.

Caso ocorra erros, a função **fopen** retorna **NULL** para indicar que houve falha na abertura de uma arquivo.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main () {
4    FILE* arq;
5    arq = fopen ("alunos.dat", "rb+");
6    if (arq == NULL) {
7      printf ("Erro na abertura do arquivo!\n");
8      exit(1);
9    }
10   /* resto do codigo */
11 }
```

Fechamento

Quando um arquivo aberto não for mais utilizado em um programa, **ele deve ser fechado**. Esse fechamento desassocia o arquivo físico, em disco, com a variável do tipo arquivo usada para abri-lo.

```
1 fclose (FILE *arquivo);
```

É muito importante lembrar de fechar um arquivo. Só assim, você garante que qualquer alteração no arquivo seja persistida no disco.

Escrita

A função **fwrite** pode ser usada para escrever em um arquivo binário.

Exemplo de uso:

```
1 fwrite (&registro, sizeof(tipo_registro), 1, arquivo);
```

Escrita

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct {
  float largura, altura;
} t_retangulo;
int main () {
  FILE *arquivo = fopen ("binario.dat", "ab");
  if (arquivo == NULL) {
    printf ("Erro na abertura do arquivo.\n");
    exit(1);
  }
  t_retangulo retangulo;
  retangulo.largura = 3.0;
  retangulo.altura = 2.0;
  fwrite (&retangulo, sizeof(t_retangulo), 1, arquivo);
  fclose (arquivo);
}
```

Final de arquivo

Uma função essencial usada para trabalhar com arquivos é o **feof**.

Seu formato é:

```
1 int feof (FILE *arquivo);
```

A função retornará 0 se o arquivo passado como argumento ainda não chegou ao final.

Leitura

A função **fread** pode ser usada para ler um arquivo binário.

O formato da função é:

```
1 fread (&registro, sizeof(tipo_registro), 1, arquivo);
```

Leitura

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct {
  float largura, altura;
} t_retangulo;
int main () {
  FILE *arquivo = fopen ("binario.dat", "rb");
  /* verifica se a abertura ocorreu */
  t_retangulo retangulo;
  while (1) {
    fread (&retangulo, sizeof(t_retangulo), 1, arquivo);
    if (feof(arquivo))
      break;
    printf("Largura= %.1f e Altura= %.1f\n", retangulo.largura,
    retangulo.altura);
  }
  fclose (arquivo);
}
```

Mudar posição no arquivo

A função **fseek** é usada para posicionar o **indicador de posição de arquivo**.

O formato da função é:

```
1 fseek(FILE *arquivo, posicao * sizeof (tipo_registro), SEEK_SET);
```

Observe que o segundo parâmetro é a quantidade de bytes o **indicador de posição de arquivos** deve se posicionar desde o começo do arquivo.

Exemplo

Exemplo

A seguir, um exemplo de uso de um arquivo será implementado.

O arquivo conterá um registro de pessoas com duas informações:

- nome
- idade

O primeiro passo é criar um arquivo vazio no terminal:

```
1 touch dados.bin
```

Exemplo

Seguiremos criando funcionalidades para:

- adicionar um registro;
- listar os registros;
- localizar um registro;
- apagar um registro;

As funcionalidades podem ser implementadas no programa principal, em vários programas ou numa biblioteca.

Será necessário incluir as seguintes bibliotecas:

```
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
```

Definido o registro

O registro é definido pelos campos nome, idade e validade.

```
1 typedef struct {
2   char nome[10];
3   int idade;
4   int valido;
5 } t_amigo;
```

Adicionando registro no final

É importante informar o modo de operação será o de adicionar registro no final (”**ab**”).

```
1  FILE* arquivo;
2  arquivo = fopen ("dados.bin", "ab");
3  t_amigo amigo;
4  printf("NOME: ");
5  scanf(" %s", amigo.nome);
6  printf("IDADE: ");
7  scanf(" %d", &amigo.idade);
8  amigo.valido = 1;
9  fwrite (&amigo, sizeof (t_amigo), 1, arquivo);
10 fclose (arquivo);
```

Listando todos os registros

É importante informar o modo de operação será apenas de leitura (”**rb**”).

```
1  FILE* arquivo;
2  arquivo = fopen ("dados.bin", "rb");
3  t_amigo amigo;
4  while (1) {
5    fread (&amigo, sizeof(t_amigo), 1, arquivo);
6    if (feof (arquivo))
7      break;
8    if (amigo.valido)
9      printf("%s tem %d anos\n", amigo.nome, amigo.idade);
10 }
11 fclose(arquivo);
```

Procurando um registro

```
1  int procurar (char nome[]) {
2    FILE* arquivo = fopen ("dados.bin", "rb");
3    int cont = 0;
4    t_amigo amigo;
5    while (1) {
6      fread(&amigo, sizeof(t_amigo), 1, arquivo);
7      if (feof(arquivo)) {
8        cont = -1;
9        break;
10     }
11     if (amigo.valido && !strcmp(nome, amigo.nome)) {
12       break;
13     }
14     cont++;
15   }
16   fclose (arquivo);
17   return cont;
18 }
```

Recuperando informações de um registro

```
1  char nome[10];
2  printf("Nome: ");
3  scanf ("%s", nome);
4  int posicao = procurar (nome);
5  if (posicao == -1)
6    printf("No encontrado\n");
7  else {
8    FILE* arquivo = fopen ("dados.bin", "rb");
9    t_amigo amigo;
10   fseek (arquivo, posicao * sizeof(t_amigo), SEEK_SET);
11   fread(&amigo, sizeof(t_amigo), 1, arquivo);
12   printf("%s tem %d anos\n", amigo.nome, amigo.idade);
13   fclose (arquivo);
14 }
```

Alterando um registro

É importante informar o modo de operação será de leitura e escrita.

```
1  char nome[10];
2  printf("Nome: ");
3  scanf ("%s", nome);
4  int posicao = procurar (nome);
5  if (posicao == -1)
6    printf("Nao encontrado\n");
7  else {
8    FILE* arquivo = fopen ("dados.bin", "rb+");
9    t_amigo amigo;
10   fseek (arquivo, posicao * sizeof(t_amigo), SEEK_SET);
11   fread(&amigo, sizeof(t_amigo), 1, arquivo);
12   printf("Digite nova idade: ");
13   scanf("%d", &amigo.idade);
14   fseek (arquivo, posicao * sizeof(t_amigo), SEEK_SET);
15   fwrite(&amigo, sizeof(t_amigo), 1, arquivo);
16   fclose (arquivo);
17 }
```

Removendo (invalidando) um registro

É importante informar o modo de operação será de leitura e escrita.

```
1  char nome[10];
2  printf("Nome: ");
3  scanf ("%s", nome);
4  int posicao = procurar (nome);
5  if (posicao == -1)
6    printf("No encontrado\n");
7  else {
8    FILE* arquivo = fopen ("dados.bin", "rb+");
9    t_amigo amigo;
10   fseek (arquivo, posicao * sizeof(t_amigo), SEEK_SET);
11   fread(&amigo, sizeof(t_amigo), 1, arquivo);
12   amigo.valido = 0;
13   fseek (arquivo, posicao * sizeof(t_amigo), SEEK_SET);
14   fwrite(&amigo, sizeof(t_amigo), 1, arquivo);
15   fclose (arquivo);
16 }
```

Recapitulando

É importante sempre informar o modo de operação do arquivo:

- **"ab"** para adicionar no final do arquivo;
- **"rb"** para acessá-lo em modo apenas leitura;
- **"rb+"** para acessá-lo em modo leitura e escrita.

Os passos sempre são os mesmo:

- **Adicionar:** abra no modo adicionar no final de arquivo e escreva um novo registro;
- **Listar:** abra no modo somente leitura e leia os registros;
- **Procurar:** abra no modo somente leitura, leia os registro e compare. Anote a posição do registro encontrado;
- **Alterar/Apagar:** abra no modo leitura e escrita, procure o registro, posicione o **indicador de posição de arquivo** (IPA), leia o registro, altere os valores, reposicione o IPA, grave o registro.

Exercícios

Exercícios

1. Faça um programa de uma agenda telefônica. Para cada contato, ela deve guardar o nome, telefone e a idade. Os dados devem ser guardados em um arquivo.
   Faça com que o programa tenha um menu com as seguintes opções:
   - Cadastrar um contato.
   - Localizar um contato (informe o nome e ele retorna o telefone e a idade).
   - Alterar um contato (informe o nome e pergunta se deseja alterar os valores deste contato: nome, telefone e idade).
   - Apagar um contato (informe o nome e ele apagar o primeiro contato que este mesmo nome).
   - Listar todos os contatos.
   - Calcula a média de idade dos seus contatos.

Referências

Referências

1 ASCENCIO, A.F.G.; CAMPOS, E.A.V. de. Fundamentos da Programção de Computadores. Algoritmos, Pascal e C/C++. São Paulo: Pearson Prentice Hall, 2012.

2 VAREJÃO, F. M. V. Introdução à programação: uma nova abordagem usando C. Rio de Janeiro: Elsevier, 2015.

3 BACKES, A. Linguagem C: completa e descomplicada. Rio de Janeiro: Elsevier, 2013.