Linguagem C

Armazenamento de Dados em Arquivos

Drª Alana Morais

Arquivos

- Arquivos são importantes em computação pois são a forma de se perpetuar dados em dispositivos de armazenamento, ao contrário da memória de trabalho, que é volátil.
- Toda a organização dos computadores se baseia no conceito de arquivos, inclusive as metáforas de uso de dados e de sua hierarquia de armazenamento (pastas, diretórios).
- É fundamental para um programador saber registrar dados no formato de arquivos, bem como conhecer as operações de manipulação destes.

Arquivos

- Você deverá ser capaz ao final da aula:
  - Entender o armazenamento de dados na forma de arquivos;
  - Usar os arquivos como forma de registro de dados;
  - Realizar as operações de manipulação de arquivos.

Arquivos

- Um arquivo é uma fonte de dados para o programa, podendo servir como entrada de dados (input) ou é um destino dos dados gerados pelo programa, ou a saída de dados (output).
- A Linguagem C processa os bytes por meio de *streams* (conjunto sequencial de caracteres, ou de bytes, sem qualquer estrutura interna).
  - As streams, portanto, independem do dispositivo utilizado e representam um conceito universal de fluxo de dados.
  - Cada *stream* está ligada a um arquivo, que pode não corresponder fisicamente a uma estrutura existente no disco, como é o caso do teclado ou da tela do computador.

Arquivos

- Todo arquivo, quando aberto em Linguagem C, vai se encaixar em uma das seguintes possibilidades de padrão de stream:
  - `stdin – a entrada-padrão, em geral o teclado;`
  - `stdout – a saída-padrão, em geral a tela;`
  - `stderr – a saída-padrão das mensagens de erro, em geral a tela;`
  - `stdaux – a porta de comunicação;`
  - `stdprn – a saída para a impressora.`

Arquivos

- Todo arquivo, quando aberto em Linguagem C, vai se encaixar e
- As principais tarefas que podem ser realizadas sobre arquivos são:
  - Abertura: associamos uma variável ao arquivo com o qual se pretende trabalhar;
  - Utilização: depois de aberto, podemos manipular o conteúdo do arquivo: ler dados, escrever dados, posicionar ponteiro em dado trecho do arquivo, entre outras tarefas vistas neste capítulo;
  - Fechamento: devemos fechar os arquivos depois do uso para evitar perdas de dados e do próprio arquivo.

Arquivos

- Em Linguagem C, todas essas operações com arquivos são precedidas pela letra f (de file): fopen, fclose.
- As funções específicas para se trabalhar com arquivos são:
  - `fopen( ) – abre um arquivo para trabalho;`
  - `fclose( ) – fecha o arquivo;`
  - `putc( ) – escreve um caractere no arquivo aberto;`
  - `fputc( ) – mesma função de putc( );`
  - `getc( ) – lê um caractere do arquivo de trabalho;`
  - `fgetc( ) – mesma função de getc( );`
  - `fseek( ) – posiciona o ponteiro de arquivo em um byte específico;`
  - `rewind( ) – posiciona o ponteiro de arquivo no início deste;`
  - `fprintf( ) – idem ao printf na saída-padrão;`
  - `fscanf( ) – idem ao scanf na entrada-padrão.`

Arquivos - Abrindo arquivos

- Para se abrir um arquivo, declaramos uma variável do tipo FILE (essa é, na verdade, um ponteiro para FILE – que está definido em stdio.h e não é do tipo primitivo):

- Comandos para gravação e leitura de arquivos:

```
FILE *arquivo;
arquivo = fopen(“nome”, “modo”);
fclose (arquivo);
```

Arquivos - Abrindo arquivos

- 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!!!

Arquivos

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 - Escrevendo e Apagando

- Essa operação sobrescreve o conteúdo já existente no arquivo.
- Comandos para gravação e leitura de arquivos:
  - `FILE *arquivo;`
  - `arquivo = fopen(“nome”, “w”);`
  - `fprintf(arquivo, "%s \n", mensagem);`
  - `fclose (arquivo);`

Arquivos - Escrevendo e Mantendo

- É a operação de anexação (append).
- Em primeiro lugar, precisamos abrir o arquivo em modo que permita adicionar conteúdo, e depois gravar o texto.
- Comandos para gravação e leitura de arquivos:
  - `FILE *arquivo;`
  - `arquivo = fopen(“nome”, “a”);`
  - `fprintf(arquivo, "%s \n", mensagem);`
  - `fclose (arquivo);`

Arquivos - Escrevendo Strings

- Esse programa acumula cadeias de caracteres em várias linhas, até que o caractere de nova linha seja encontrada.
- Comandos para gravação e leitura de arquivos:
  - `FILE *arquivo;`
  - `arquivo = fopen(“nome”, “modo”);`
  - `fputs(mensagem, arquivo);`
  - `fclose (arquivo);`

Arquivos - Lendo o conteúdo

- A função fgets( ) fará a leitura a partir de um buffer de bytes, ou seja, um bloco de dados de determinado tamanho.

- Comandos para gravação e leitura de arquivos:
  - `FILE *arquivo;`
  - `arquivo = fopen("nome", "modo");`
  - `fputs(mensagem, arquivo);`
  - `fgets(mensagem,tam,arquivo);`
  - `feof(arquivo)`
  - `fclose (arquivo);`

Arquivos - Lendo o conteúdo II

- Comando para leitura de dados de arquivos:

```
fread (&variavel, sizeof ( tipo_var), t,arquivo);
```

Onde:

- variavel : variável a ser lida do arquivo (tipos básicos ou compostos, porém apenas variáveis, e não vetores);

-tipo_var: o tipo da variável a ser lida do arquivo;

-t é a quantidade de dados a ser lida(1 para uma só variável, mais para leitura de vetores);

-arquivo: variável de arquivo

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

Arquivos - Escrevendo II

- Comando para escrita de dados em arquivos:

```
fwrite (&variavel, sizeof ( tipo_var), t,arquivo);
```

Onde:

-variavel : variável a ser escrita no arquivo (tipos básicos ou compostos, porém apenas variáveis, e não vetores);

-tipo_var: o tipo da variável a ser escrita no arquivo;

-t é a quantidade de dados a ser escrita(1 para uma só variável, mais para leitura de vetores);

-arquivo: variável de arquivo

Obs: o comando fwrite retorna um número inteiro: um número maior que zero significa que a escrita foi feita corretamente, 0(zero) indica erro de escrita no arquivo;

Arquivos

- Comando para re-abertura de arquivos:
  - `rewind(arquivo);`

  Reinicia o ponto de leitura/escrita do arquivo. Ao abrir o arquivo, a cada leitura/escrita, o arquivo vai para a "próxima" variável. O comando rewind reinicia a leitura/escrita a partir da primeira posição do arquivo.

- Comando para verificar o final do arquivo:
  - `feof(arquivo);`

  Obs: o comando feof retorna um número inteiro: um número maior que zero significa que o arquivo está no final, e não há mais dados no arquivo, 0(zero) indica que ainda existem dados;

Exemplo

Programa para Lista Postal
Pseudocódigo

```
/* Um programa de lista postal muito simples */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define TAM 2

struct Elemento
{
       char nome [100];
       char rua [100];
       char cidade [100];
       char estado [2];
       char cep [10];
} Lista [TAM];

char menu ();
void inicia_lista ();
void cadastra ();
void mostra ();
void salva ();
void carrega ();
```

```
int main()
{
char escolha;
 inicia_lista();
 for ( ;; )
 {escolha = menu();
  switch (escolha)
  {case 'c':
   case 'C': { cadastra(); } break;
   case 'm':
   case 'M': { mostra(); } break;
   case 's':
   case 'S': { salva(); } break;
   case 'a':
   case 'A': { carrega(); } break;
   case 't':
   case 'T': { exit (0 ); } break;
   default : { printf ( "Opcao invalida. \n" ); }
  }
 printf ( "\n \n \n" );
}
system ( "Pause" );
return 0;
}
```

```
char menu()
{
char opcao;

 printf ("\n \n \n");
 printf ( " (C)adastrar. \n" );
 printf ( " (M)ostrar. \n" );
 printf ( " C(A)arregar. \n" );
 printf ( " (S)alvar. \n" );
 printf ( " (T)erminar. \n" );
 fflush(stdin);
 scanf ( "%c", &opcao );
 return opcao;
}
```

```
void mostra()
{
int t;
  printf ("\n \n \n");
  for( t = 0; t < TAM; t++ )
  {
  if (!(strcmp(Lista[t].nome, "")==0) )
  {
   printf ( "%s \n", Lista[t].nome);
   printf ( "%s \n", Lista[t].rua);
   printf ( "%s \n", Lista[t].cidade);
   printf ( "%s \n", Lista[t].estado);
   printf ( "%s \n", Lista[t].cep);
  }
  printf ("\n");
  }
}
```

```
void inicia_lista()
{
int t;
for (t = 0; t < TAM; t++)
 {
 strcpy(Lista[t].nome , "");
 }
}
```

```
void cadastra ()
{
int i;
 printf ("\n \n \n");
 for (i = 0; i < TAM; i++ )
 {
    printf ( "Nome: \n" );fflush (stdin);
    gets ( Lista[i].nome );
    printf ( " Rua: \n" );fflush (stdin);
    gets ( Lista[i].rua );
    printf ( "Cidade: \n" );fflush(stdin);
    gets ( Lista[i].cidade );
    printf ( "Estado: \n" );fflush(stdin);
    gets ( Lista[i].estado );
    printf ( "CEP: \n" ); fflush (stdin);
    gets ( Lista[i].cep );
  }
}
```

```
void salva ()
{
FILE *fp;
int i, result;
  printf ("\n \n \n");
  fp = fopen ("cadastro", "wb");
  if ( fp == NULL )
  {
     printf ( "O arquivo nao pode ser aberto. \n" );
     return;
  }
  for (i = 0; i < TAM; i++ )
  {
      if ( !(strcmp(Lista[i].nome, "")==0) )
      {
       result = fwrite ( &Lista[i], sizeof ( struct Elemento ), 1, fp );
       if ( result != 1 )
       {
          printf ( "Erro de escrita no arquivo. \n" );
       }
      }
  }
  fclose (fp);
}
```

```
void carrega ()
{
FILE *fp;
int i, resp;
    printf ("\n \n \n");
    fp = fopen ( "cadastro", "rb" );
    if ( fp == NULL )
    {
       printf ( "O arquivo nao pode ser aberto. \n" );
       return;
    }
    inicia_lista ();
    for (i = 0; i < TAM; i++ )
    {
        resp = fread ( &Lista[i], sizeof (struct Elemento), 1, fp );
        if ( resp != 1 )
        {
         if ( feof (fp) )
         {
            break;
         }
        printf ( " Erro de leitura no arquivo. \n" );
        }
    }
    fclose ( fp );
}
```

Exercício

1. Implemente um estrutura chamada alunos, com nome, endereço e telefone.
2. Crie uma lista de 5 alunos.
3. Ao final da interação, o sistema deve salvar esses 5 alunos no arquivo.

Dúvidas?

alanamm.prof@gmail.com

Dúvidas?

alanamm.prof@gmail.com