

Estrutura de Dados

Aula 15



Este material é parte integrante da disciplina oferecida pela UNINOVE.

O acesso às atividades, conteúdos multimídia e interativo, encontros virtuais, fóruns de discussão e a comunicação com o professor devem ser feitos diretamente no ambiente virtual de aprendizagem UNINOVE.

Uso consciente do papel.

Cause boa impressão, imprima menos.

Aula 15: Arquivos

Objetivo: Estudar os arquivos, os quais permitem o armazenamento de dados de forma não volátil em discos.

Arquivos

Caro aluno, até agora nós estudamos estruturas que nos permitem armazenar dados em variáveis e estruturas que ficam armazenadas na memória do computador. Este tipo de armazenamento é volátil, ou seja, só existe enquanto o programa está sendo executado e enquanto o computador tiver energia. Se o programa finaliza ou ocorre um pique de energia, perdemos tudo. Isso nos motiva a estudar os arquivos, que vão nos permitir armazenar dados no disco do computador. Assim, quando quisermos usar os dados novamente, basta recuperarmos o arquivo e obter os dados lá armazenados (de forma não volátil).

Mas antes de começarmos a trabalhar com arquivos, vamos entender um importante conceito da linguagem C: *stream*.

O conceito de *stream*

Para fornecer ao programador uma interface independente de dispositivo, o sistema de E/S da linguagem C trabalha com um conceito abstrato denominado ***stream***, o qual consiste em um dispositivo lógico associado ao dispositivo físico, denominado “**arquivo**”. Este, por sua vez, pode ser desde um arquivo propriamente dito, até a tela do computador ou uma impressora.

Declaração de arquivos

Existem 2 nomes que são associados a um arquivo de dados: o **nome físico**, que é o nome com o qual o arquivo é gravado no disco, e o **nome lógico**, que é o nome que é utilizado no programa que manipula o arquivo.

Na linguagem C o nome lógico é chamado de ponteiro de arquivo.

O ponteiro de arquivo

A maioria das funções da linguagem C tem como argumento um ponteiro de arquivo, usado pela **stream** associada para direcionar as operações das funções de E/S.

Para declararmos um arquivo em C, é necessário que usemos uma variável ponteiro do tipo **FILE** (que é definido na biblioteca <stdio.h>).

Exemplo:

```
FILE *arq;    // Declara uma variável ponteiro de arquivo
```

```
// arq corresponde ao nome lógico do arquivo
```

A função fopen()

Para usarmos um arquivo é necessário que ele seja criado ou aberto, caso exista. Além disso, precisamos associar seu nome lógico declarado como ponteiro de arquivo com seu nome físico. A função fopen() é responsável por estas tarefas. Vejamos as características dessa função:

- Arquivo de cabeçalho a ser incluído: <stdio.h>.
- Abre uma **stream** e associa um arquivo a ela.
- Recebe como parâmetros o nome do arquivo e o modo em que ele deve ser aberto (veja a tabela 1 que traz os modos de abertura possíveis).
- Retorna um ponteiro para arquivo ou, se ocorrer alguma falha, retorna um **NULL** (definido em <stdio.h> como '\0').

Exemplo:

```
FILE *arq;
```

```
arq = fopen( "teste.txt", "r");  
if (arq == NULL)  
{ printf("\nErro na abertura do arquivo teste.txt!\n");  
  getch();  
}
```

```
exit(1);
}
```

O que este trecho de código faz:

- 1) Declara um ponteiro de arquivo chamado `arq` (nome lógico do arquivo).
- 2) Associa `arq` com o nome físico "teste.txt".
- 3) Abre o arquivo no modo de leitura (indicado por "r").
- 4) Verifica se ocorreu algum erro na abertura do arquivo (`arq == NULL`) e, caso positivo, imprime uma mensagem de erro e finaliza (`exit(1)`).

Tabela 1

Valores para o parâmetro modo de abertura de arquivo da função `fopen()`

Modo	Significado
r	Abre um arquivo texto para leitura.
w	Cria um arquivo texto para escrita.
a	Anexa a um arquivo texto.
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.
rb	Abre um arquivo binário para leitura.
wb	Cria um arquivo binário para escrita.
ab	Anexa a um arquivo binário.
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.

Tipos de arquivos da linguagem C

Na linguagem C podemos trabalhar com dois tipos de arquivo: arquivos texto e arquivos binários. Na tabela 1 podemos identificar seis modos de abertura para cada um dos dois tipos de arquivo.

Os arquivos do tipo texto são editáveis. Você pode criar o arquivo usando um editor de textos sem formatação e pode também abrir o arquivo num editor de textos e ver/alterar os dados.

Os arquivos do tipo binário não são editáveis. Você pode criar um arquivo binário somente por meio de um programa C e também pode ler/gravar/ dados neste arquivo apenas por um programa C.

Por estas definições fica claro que os arquivos texto são fáceis de serem criados e manipulados, mas podem ser facilmente corrompidos. Já os arquivos binários, como são protegidos por programa, são mais seguros, mas dependem de um programa para serem manipulados.

A função `fclose()`

Após finalizarmos o uso de um arquivo, é necessário que ele seja fechado. A função `fclose()` é responsável por esta tarefa. Vejamos as características dessa função:

- Arquivo de cabeçalho a ser incluído: **<stdio.h>**
- Fecha uma **stream** que foi aberta por uma chamada a `fopen()`.
- Recebe como parâmetro um ponteiro de arquivo.
- Se bem sucedida, retorna o valor zero.
- Um valor de retorno diferente de zero indica ocorrência de erro, podendo ser utilizada a função `ferror()` para se determinar o erro ocorrido.

Exemplo:

```
FILE *arq;

arq = fopen( "teste.txt", "w");
if (arq == NULL)
{ printf("\nErro na abertura do arquivo teste.txt!\n");
  getch();
```

```
        exit(1);
    }

    ....

    fclose(arq);
```

Observação: Em geral, o sistema operacional impõe um limite ao número de arquivos os quais podem ser simultaneamente abertos, por isso deve-se fechar um arquivo ao se encerrar o seu uso.

Exemplo de programa que cria, grava e lê dados num arquivo texto

/* file1.c - Ilustra a criação de arquivos do tipo texto */

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
FILE *arq;
char nome[30], nomearq[30];
int idade, idadearq;
float altura, alturaarq;

int main()
{ arq = fopen("teste.txt", "w");
  // abre arquivo para escrita, ou seja, cria o arquivo
  if (arq == NULL)
  { printf("\nErro na criação do arquivo teste.txt");
    getch();
    exit(1);
  }

  puts("\nQual eh o seu nome? ");
  gets(nome);
  puts("\nQual eh o sua idade? ");
  scanf("%i",&idade);
  puts("\nQual eh a sua altura? ");
  scanf("%f",&altura);

  puts("\nGravando seus dados no arquivo teste.txt\n");
  fprintf(arq,"%s %i %.2f \n",nome,idade,altura);
  fclose(arq);
```

```
printf("\nAgora vamos abrir o arquivo e ler os dados gravados");
arq = fopen("teste.txt", "r");
if (arq==NULL)
{ printf("\nErro na abertura do arquivo teste.txt");
  getch();
  exit(1);
}
fscanf(arq,"%s %i %f \n",nomearq,&idadearq,&alturaarq);
printf("\n\nDados do arquivo: Nome=%s Idade=%i Altura=%.2f \n",
      nomearq,idadearq,alturaarq);
fclose(arq);
printf("\n\nFechando o arquivo. Fim do programa\n");
puts("\nHasta la vista, baby!\n");
getch();
return 0;
}
```

Caro aluno, um exercício interessante agora é implementar este programa, para, em seguida, ir no diretório corrente e verificar se o arquivo realmente está lá com os dados que você forneceu.

Agora, vamos praticar resolvendo os exercícios propostos. Leia a lista, resolva os exercícios e verifique seu conhecimento. Caso fique alguma dúvida, leve a questão ao Fórum e divida com seus colegas e professor.

REFERÊNCIAS

SCHILDT, H. *C Completo e total*. São Paulo: Makron Books, 1997.

TENEMBAUM, Aaron M., et al. *Estruturas de dados usando C*. São Paulo: Pearson Makron Books, 1995.