



Introdução à Programação

Eduardo Silva Lira

XLVIII Programa de Verão do IME-USP

São Paulo - SP, Jan 2019



INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO



Problema: Agrupar uma coleção de dados de diferentes tipos, mas que são logicamente relacionados a uma única entidade?

Array?

Problema: Agrupar uma coleção de dados de diferentes tipos, mas que são logicamente relacionados a uma única entidade?

Array? NÃO!

Solução: **estruturas (struct)**

Estruturas

- Uma estrutura é uma coleção de variáveis de tipos diferentes sob um mesmo “nome”;
- As variáveis são chamadas membros da estrutura;
- Uma estrutura também é chamada de “tipo definido pelo usuário”.

endereço

rua

número

CEP

bairro

Definindo Estruturas

Sintaxe:

```
struct identificador_struct
{
    <tipo1> <identificador1>;
    <tipo2> <identificador2>;
    ...
    <tipoN> <identificadorN>;
};
```

Sintaxe de declaração da variável do “tipo” definido:

```
struct identificador_struct identificador_variavel;
```

Nota: os membros de uma estrutura não ocupam espaço na memória até que eles estejam associados a uma variável do tipo da struct.

Exemplo no código: struct.c

Introdução

Manipulação de arquivos em C

Créditos do Material:

Estrutura de Dados II

Prof Jairo Francisco de Souza

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
- Um STREAM é associado a um arquivo por uma operação de abertura do arquivo e, a partir da associação, todas as demais operações de escrita e leitura podem ser realizadas

Manipulação de Arquivo em C

- A tabela abaixo apresenta as principais funções da linguagem C para manipulação de arquivos

| Função | Ação |
|--|--|
| <code>fopen()</code> | Abre um arquivo |
| <code>fclose()</code> | Fecha um arquivo |
| <code>putc()</code> e <code>fputc()</code> | Escreve um caractere em um arquivo |
| <code>getc()</code> e <code>fgetc()</code> | Lê um caractere de um arquivo |
| <code>fseek()</code> | Posiciona em um registro de um arquivo |
| <code>fprintf()</code> | Efetua impressão formatada em um arquivo |
| <code>fscanf()</code> | Efetua leitura formatada em um arquivo |
| <code>feof()</code> | Verifica o final de um arquivo |
| <code>fwrite()</code> | Escreve tipos maiores que 1 byte em um arquivo |
| <code>fread()</code> | Lê tipos maiores que 1 byte de um arquivo |

Manipulação de Arquivo em C

- O sistema de entrada e saída do ANSI C, sendo composto por uma série de funções, cujos protótipos estão reunidos em `stdio.h`
- Todas as funções relacionadas anteriormente trabalham com o conceito de ponteiro de arquivo

um
ponteiro de arquivo pode ser declarado da seguinte
maneira:

```
FILE *Arquivo;
```

Manipulação de Arquivo em C

- Pela declaração do ponteiro anterior, passa a existir uma variável de nome `Arquivo`, que é ponteiro para um arquivo a ser manipulado
- O ponteiro de arquivo une o sistema de E/S a um buffer e não aponta diretamente para o arquivo em disco, contendo informações sobre o arquivo, incluindo nome, status (aberto, fechado e outros) e posição atual sobre o arquivo

Abrindo um 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") ;
```

Abrindo um Arquivo

- Além do modo de escrita, a linguagem C permite o uso de alguns valores padronizados para o modo de manipulação de arquivos, conforme mostra a tabela abaixo:

| Modo | Significado |
|-------|---|
| "r" | Abre um arquivo texto para leitura. O arquivo deve existir antes de ser aberto. |
| "w" | Abrir um arquivo texto para gravação. Se o arquivo não existir, ele será criado. Se já existir, o conteúdo anterior será destruído. |
| "a" | Abrir um arquivo texto para gravação. Os dados serão adicionados no fim do arquivo ("append"), se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente. |
| "rb" | Abre um arquivo binário para leitura. Igual ao modo "r" anterior, só que o arquivo é binário. |
| "wb" | Cria um arquivo binário para escrita, como no modo "w" anterior, só que o arquivo é binário. |
| "ab" | Acrescenta dados binários no fim do arquivo, como no modo "a" anterior, só que o arquivo é binário. |
| "r+" | Abre um arquivo texto para leitura e gravação. O arquivo deve existir e pode ser modificado. |
| "w+" | Cria um arquivo texto para leitura e gravação. Se o arquivo existir, o conteúdo anterior será destruído. Se não existir, será criado. |
| "a+" | Abre um arquivo texto para gravação e leitura. Os dados serão adicionados no fim do arquivo se ele já existir, ou um novo arquivo será criado, no caso de arquivo não existente anteriormente. |
| "r+b" | Abre um arquivo binário para leitura e escrita. O mesmo que "r+" acima, só que o arquivo é binário. |
| "w+b" | Cria um arquivo binário para leitura e escrita. O mesmo que "w+" acima, só que o arquivo é binário. |
| "a+b" | Acrescenta dados ou cria um arquivo binário para leitura e escrita. O mesmo que "a+" acima, só que o arquivo é binário. |

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 (STREAM):

```
fclose (Arquivo) ;
```

Gravando e lendo Dados em Arquivos

- Existem várias funções em C para a operação de gravação e leitura de dados em arquivos. Abaixo seguem algumas:
 - ***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

Sintaxe das funções para gravação

- Grava o conteúdo da variável caracter no arquivo

```
putc (character, 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 ...);
```

Lendo e Gravando 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)
```

Lendo e Gravando 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

Lendo e Gravando Estruturas

- Normalmente é necessário manipular arquivos por meio de estruturas de dados ou arquivos de estruturas (struct)
- Podemos por exemplo falar num arquivo de **CLIENTES**, onde cada cliente possui **NOME, RG, ENDERECO E TELEFONE**

Lendo e Gravando 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);
```

Posicionando em um 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

Posicionando em um 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>);
```

Posicionando em um 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);
```


Posicionando em um 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

Dúvidas?