Profa. Michele Fúlvia Angelo

PGCA 028 – Tópicos Especiais em Tecnologia Computacional I – Introdução à Programação de Computadores
Aula 10

Universidade Estadual de Feira de Santana

Sumário

- Registros
- Arquivos

Registros

Vetores e Matrizes

- Variáveis Compostas Homogêneas (mesmo tipo)

lista										
posição	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
valor										

Todos os elementos são *int*

matriz					
índices	0	1	2	3	4
0					
1					
2					
3					

Todos os elementos são *int*

Registros

- Variáveis Compostas Heterogêneas
- Podem agrupar variáveis de tipos diferentes
- Variáveis (campos) com nomes diferentes mas relacionadas e referenciadas por um nome comum
- Chamadas de registros ou estruturas

Pessoa			
Nome (string)	**Estado** (string)	**Idade** (int)	**Sexo** (char)

Registros

```
typedef struct {
  char nome[50];
  char estado[20];
  int idade;
  char sexo;
} tipopessoa;
...
tipopessoa pessoa;
```

Pessoa			
Nome (string)	**Estado** (string)	**Idade** (int)	**Sexo** (char)

Registros

```
tipopessoa pessoa;
...
strcpy(pessoa.nome, “Fulano”);
strcpy(pessoa.estado, “Paraná”);
pessoa.idade = 37;
pessoa.sexo = ‘M’;
```

Pessoa			
Nome (string)	**Estado** (string)	**Idade** (int)	**Sexo** (char)
Fulano	***Paraná***	***37***	***M***

Registros

- Como representar uma estrutura como essa?

```
tipopessoa pessoa1,pessoa2,pessoa3;
...
strcpy(pessoa1.nome, ”Fulana da Silva”);
strcpy(pessoa2.nome, ”Cicrano de Tal”);
pessoa1.idade = 30;
pessoa2.idade = pessoa1.idade * 2;
pessoa3 = pessoa1;
scanf(“%c”, &pessoa1.estado);
printf(“%c”, pessoa1.estado);
```

Registros

Nome (string)	**Estado** (string)	**Idade** (int)	**Sexo** (char)
Fulano	***Paraná***	***37***	***M***
Cicrano	***Rio de Janeiro***	***15***	***M***
Beltrano	***Acre***	***22***	***M***
Fulana	***Pará***	***25***	***F***
Beltrana	***Ceará***	***76***	***F***

Registros

Um vetor de registros!

tipopessoa pessoas[5];

Pessoas	
posição	**valor**
0	(registro tipopessoa)
1	(registro tipopessoa)
2	(registro tipopessoa)
3	(registro tipopessoa)
4	(registro tipopessoa)

Pessoa[0]			
Nome (string)	**Estado** (string)	**Idade** (int)	**Sexo** (char)
Fulano	***Paraná***	***37***	***M***

Registros

```
typedef struct{
  char nome[50];
  char estado[20];
  int idade;
  char sexo;
} tipopessoa;

int main(){
 tipopessoa pessoas[5]; /*tabela de registros*/

 strcpy(pessoas[0].nome,“Fulano”);
 strcpy(pessoas[0].estado,“Paraná”);
 pessoas[0].idade=37;
 pessoas[0].sexo='M';
 strcpy(pessoas[1].nome,”Cicrano”);
 strcpy(pessoas[1].estado,”Rio de Janeiro”);
 pessoas[1].idade=15;
 pessoas[1].sexo='M';
}
```

Arquivos

Arquivos

- Estruturas de dados armazenadas fora da memória principal do computador;
- Memória secundária (por exemplo:HD);
- Maior quantidade de informação;
- Uso futuro (persistência);
- Nome e caminho para identificação;

Arquivos

- Arquivos Texto
  Composto por uma sequência de caracteres organizados por linhas (ex. código-fonte na linguagem C).

- Arquivos Binários
  Composto por uma sequência de bits, da mesma forma como fica na memória principal (ex. programa executável).

Arquivos

- Quando você envia caracteres para serem gravados em um arquivo, estes caracteres são armazenados temporariamente em uma área de memória (o *buffer* ou *stream*) ao invés de serem escritos em disco imediatamente.

- Quando o *buffer* estiver cheio, seu conteúdo é escrito no disco de uma vez.

- A razão para se fazer isto tem a ver com a eficiência nas leituras e gravações de arquivos. As gravações só serão efetuadas quando houver um volume razoável de informações a serem gravadas ou quando o arquivo for fechado.

Arquivos de Texto

- Arquivos de Texto no C (stdio.h)
  - Tipo FILE para representar um arquivo
    FILE *arquivo;
  - Mas qual arquivo será usado?
  - Função fopen recebe o nome e o caminho e abre o arquivo

- arquivo = fopen(“meutexto.txt”,”w”);

- arquivo = fopen(“c:\\doc\\arq.c”,”w”);

Abertura e Fechamento de um Arquivo de Texto

- ponteiroarq = fopen(nomeexterno, modo);
  - Modos de Abertura:
    - r : Abre o arquivo somente para leitura. Se o arquivo não existir, dá erro.
    - w : Abre o arquivo somente para escrita. SEMPRE cria um novo arquivo. Portanto, se o arquivo já existir ele é apagado.
    - a : Abre o arquivo para leitura/escrita. Escreve apenas no final do arquivo. Caso o arquivo já exista, ele abre, caso contrário ele cria o arquivo.
    - r+ : Abre o arquivo para leitura/escrita. Se o arquivo não existir, dá erro.
    - w+ : Abre o arquivo para leitura/escrita. SEMPRE cria um novo arquivo. Portanto, se o arquivo já existir ele é apagado.
- fclose (ponteiroarquivo);

Comandos de Leitura e Escrita

- Comandos de leitura:
  - fscanf (ponteiroarquivo, …);

  igual ao scanf, acrescentando o ponteiro para arquivo no começo.

  Obs.: somente para arquivos abertos com leitura

- Comandos de escrita:
  - fprintf (ponteiroarquivo, …);

  igual ao printf, acrescentando o ponteiro para arquivo no começo.

  Obs.: somente para arquivos abertos com escrita

Comandos de Leitura e Escrita

```
#include <stdio.h>
int main ( ){
  FILE *arq ;

  /* abrir arquivo para escrita*/
  arq = fopen ( ”teste.txt”, ”w”) ;
// OU
//arq= fopen("C:\\Users\\Michele\\Documents\\teste.txt", “w");

  /* escrever um texto no arquivo */
  fprintf( arq, “Testando escrita.”);

  /* fechar arquivo */
  fclose( arq ) ;
}
```

fscanf e fprintf

```
#include <stdio.h>
int main(){
  char texto[100];
  FILE *leitura, *escrita;
  leitura = fopen(“C:\\Borland\\teste.txt”, “r”); //Se teste.txt não existir dá erro
  if( leitura ==NULL ){
    printf(“Nao abriu o teste.txt”);
  } else {
    escrita = fopen(“C:\\Borland\\arq2.txt”, “w”);
    if( escrita ==NULL ){
      printf(“Nao abriu o arq2.txt”);
    } else {
      fscanf(leitura, “%s”, &texto); //Retorna a string até encontrar ‘ ’
      fprintf(escrita, “%s”, texto);
      fclose(leitura);
      fclose(escrita);
}}}
```

Exemplos

```
int main(){
 int i, n, vet[100],num;
 FILE *arq ;

 printf ("Digite a quantidade: ");
 scanf("%d", &n);
 printf ("Digite os elementos: ");
 for ( i=0; i<n; i++){
   scanf ("%d" , &vet[i] );
 }
 arq = fopen ("vetor1.txt" , "w+") ;
 fprintf (arq,"%d ", n) ;
 for ( i=0; i<n; i++){
   fprintf ( arq, "%d ", vet[i]);
 }
 rewind(arq);  //Reposiciona o ponteiro no início do arquivo
 printf("\n Os elementos do arquivo sao:");
 for (i=0; i<=n; i++){
   fscanf(arq, "%d", &num);
   printf("%d\n",num);
 }

 fclose(arq) ;
```

fgets e fputs

- char * fgets (char *str, int n, FILE *f);

Lê n caracteres de f e guarda em str, retorna nulo ou str.

- int fputs (const char *str, FILE * f);

Escreve str em f, retorna EOF ou valor não-negativo.

```
int main(){
  char texto[100],*c;
  FILE *leitura;
  leitura = fopen("teste.txt", "r"); //Se teste.txt não existir dá erro
  c=fgets(texto, 50, leitura); //'fgets' lê até 49 caracteres ou até o '\n'
  printf("%s", c);
  fclose(leitura);
}
```

fgets e fputs

```
int main(){
  char texto[100],*linha;
  FILE *leitura;
  int i=1;
  leitura = fopen("teste.txt", "r");
  while (!feof(leitura)){
      linha=fgets(texto, 50, leitura);
      if (linha) {
        printf("Linha %d : %s",i,linha);
        i++;
      }
  }
  fclose(leitura);
}
```

fgets e fputs

```
int main(){
  char texto[100],*linha;
  FILE *leitura, *escrita;
  int i=1;
  leitura = fopen("teste.txt", "r");
  escrita = fopen("saida.txt", "w");
  while (!feof(leitura)){
      linha=fgets(texto, 50, leitura);
      if (linha) {
        fputs(linha,escrita);
      }
  }
  fclose(leitura);
}
```

FEOF

```
#include <stdio.h>
int main(){
 FILE *arq;
 char str[80],c;
 printf(“Digite o nome do arquivo:\n");
 gets(str);
  printf("Este arquivo chama-se:\n%s\n",str);
 arq = fopen(str,"r");
 fscanf(arq,"%c",&c);
 while(feof(arq)==0) { //OU (! feof(arq))
   printf("%c",c);
   fscanf(arq,"%c",&c);
 }
 fclose(arq);
}
```

feof(ponteiroarquivo) é uma função que retorna 0 se ainda não foi atingido o final do arquivo, e um número diferente de 0, caso contrário.

Arquivos Binários

- Orientado a bytes e não a caracteres
- Abertura de um arquivo
  ponteiroarq = fopen(nomeexterno, modo);
- Modos de Abertura:
  b : Indica arquivo binário.
  Combinando: rb, wb, ab, rb+, wb+
- Fechamento:
  fclose

Comando de Escrita e Leitura

- size_t fwrite(const void *ptr, size_t size, size_t count, FILE *f);
- size_t fread(void *ptr, size_t size, size_t count, FILE * f);
  - **Ponteiro:** Um ponteiro para estrutura (ou vetor) que será lido ou escrito no arquivo.
  - **Tamanho:**Inteiro que indica o tamanho da estrutura, utilizamos **sizeof** para descobrir o tamanho em bytes da variável ou do tipo de dados.
  - **Quantidade:** Número de estruturas (maior que 1 quando for vetor).
  - **Arquivo:** Ponteiro para o arquivo.

Exemplos:

```
intval;
fread(&val,sizeof(int),1, leitura);
fwrite(&val,sizeof(int),1, escrita);

intvetor[100];
fread(vetor,sizeof(int),100, leitura);
fwrite(vetor,sizeof(int),100, escrita);
```

Exemplos

```
int main(){
FILE *arq ;
  int vet[] = {10,20,30}, vet2[3];
  int i;

  arq = fopen ("dados.dat", "wb");
  fwrite (vet, sizeof(vet), 1, arq);
  fclose (arq);

  arq = fopen ("dados.dat", "rb" );
  fread (vet2, sizeof(vet2), 1, arq);
  fclose( arq ) ;
  for ( i=0; i<3; i++)
    printf( "%d ", vet2[i]);
}
```

Ponteiro do Arquivo

- Para se fazer procuras e acessos randômicos em arquivos usa-se a função fseek().

- Esta função move a posição corrente de leitura ou escrita no arquivo de um valor especificado, a partir de um ponto especificado.

  int fseek(FILE *f, long int numbytes, int origin);
  - Coloca o indicador de posição de f na posição definida pela adição de numbytes a posição de referência indicada por origin.

Ponteiro do Arquivo

- O parâmetro *origin* determina a partir de onde os *numbytes* de movimentação serão contados.

- Os valores possíveis são definidos em stdio.h:

Nome	Valor	Significado
SEEK_SET	0	Início do arquivo
SEEK_CUR	1	Ponto corrente no arquivo
SEEK_END	2	Fim do arquivo

Exemplos

```
int main(){
 int numero[3],num1,num2,i;
 FILE *classe;
 classe = fopen("alunos.dat", "ab+");
 for(i=0;i<3;i++){
     scanf("%d",&numero[i]);
 }
 fwrite(numero,sizeof(numero),1, classe);
 printf("Digite o numero do aluno");
 scanf("%d",&num1);
 rewind(classe);
 fread(&num2,sizeof(num2),1,classe);
 while ((!feof(classe)) && (num1 != num2))
      fread(&num2,sizeof(num2),1,classe);
 if (feof(classe))
      printf("O aluno %d nao esta cadastrado.",num1);
 else
      printf("O aluno %d esta cadastrado.",num1);
 fclose(classe);
 }
```

Exemplos:
Gravando um Registro no Arquivo

```
int main(){
 struct aluno{
    int numero;
    char nome[20];
    float nota;
    };

  FILE *classe;
  aluno a, b;
  classe = fopen("alunos.dat", "ab+");

  printf("\nDigite o numero do aluno a ser incluido ");
  scanf("%d",&a.numero);
  printf("\nDigite o nome do aluno a ser incluido ");
  scanf("%s",a.nome);
  printf("\nDigite a nota do aluno ");
  scanf("%f",&a.nota);

  fread(&b, sizeof(aluno), 1, classe);
  while ((!feof(classe)) && (a.numero != b.numero))
       fread(&b, sizeof(aluno), 1, classe);

     if (feof(classe)){
        fwrite(&a, sizeof(aluno), 1, classe);
          printf("\nAluno incluido com sucesso!");
     }
     else
          printf("\nAluno jah cadastrado!");
  fclose(classe);
}
```

Arquivos Binários x Arquivos Texto

- **Arquivos de Texto:** Os dados são gravados como caracteres de 8 bits. Ex.: Um número inteiro de 32 bits com 8 dígitos ocupará 64 bits no arquivo.
- **Arquivos Binários:** Os dados são gravados na forma binária (do mesmo modo que estão na memória). Ex.: Um número inteiro de 32 bits com 8 dígitos ocupará 32 bits no arquivo.
  - Acesso não sequencial: função fseek();
  - Geralmente têm tempos de leitura e escrita mais rápidos que os arquivos de texto, já que uma imagem binária do registro é armazenada diretamente da memória para o disco (ou vice-versa).