MC-102 — Aula 21
Registros (Continuação)

Instituto de Computação – Unicamp

Segundo Semestre de 2011

Roteiro

Registro

O que é

Registro é uma coleção de variáveis relacionadas de vários tipos, organizadas em uma unica estrutura e referenciadas por um nome comum.

Características

- Cada variável é chamada de membro do registro
- Cada membro é acessado por um nome na estrutura
- Cada estrutura define um novo tipo, com as mesmas características de um tipo padrão da linguagem

Declarando o formato do registro

A primeira parte da criação de um registro é declarar seu formato. Isso é feito utilizando a palavra chave `struct`, da seguinte forma:

```
struct nome_do_tipo_do_registro {
  tipo_1 nome_1;
  tipo_2 nome_2;
  tipo_3 nome_3;
  ...
  tipo_n nome_n;
};
```

Exemplos de Estrutura

Ficha de dados cadastrais de um aluno

```
struct ficha_aluno {
int ra;
int telefone;
char nome[30];
char endereco[100];
int dia_nascimento;
int mes_nascimento;
int ano_nascimento;
};
```

Exemplos de Estrutura

data

```
struct data {
int dia;
int mes;
int ano;
};
```

Ficha de dados cadastrais de um aluno

```
struct ficha_aluno {
int ra;
int telefone;
char nome[30];
char endereco[100];
struct data nascimento
};
```

Declarando um registro

A próxima etapa é declarar uma variável do tipo `struct nome_do_tipo_da_estrutura`, que será usada dentro de seu programa, como no exemplo abaixo:

```
#include <stdio.h>

struct ficha { int ra; float media; };

main () {
  struct ficha f;
}
```

Utilizando os campos de um registro

Podemos acessar individualmente os campos de um determinado registro como se fossem variáveis normais, utilizando a seguinte estrutura

```
nome_do_registro.nome_do_campo
```

Utilizando os campos de um registro

- Para o registro declarado anteriomente, utilizaríamos

  ```
  f.ra
  ```

  para acessar o campo ra do registro f (note que usamos o nome da variável e não o nome dado ao formato do registro).
- Podemos colocar o campo de um registro em qualquer lugar onde colocaríamos uma variável.

Lendo os campos de um registro

A leitura dos campos de um registro a partir do teclado deve ser feita campo a campo, como se fossem variáveis independentes.

```
printf ("Digite o ra do aluno: ");
scanf ("%d", &f.ra);

printf ("Digite a média do aluno: ");
scanf ("%f", &f.media);
```

Escrevendo os campos de um registro

A escrita na tela do valor dos campos de um registro deve ser feita campo a campo, como se fossem variáveis independentes.

```
printf ("O aluno %d tirou média %f\n",
        f.ra, f.media);
```

Veja o exemplo em `leitura_escrita.c`

Copiando registros

A cópia de um registro pode ser feita como se fosse a cópia de uma variável normal, ou seja

```
registro_1 = registro_2
```

Veja o exemplo em `copia.c`

Vetor de registros

Pode ser declarado quando necessitarmos de diversas cópias de um registro (por exemplo, para cadastrar todos os alunos de uma mesma turma).

- Para declarar: `struct ficha f[5];`
- Para usar: `f[indice].campo;`

Veja o exemplo em `vetor.c`

Registros em funções

Registros podem ser usados como qualquer outro tipo.

- Por padrão, são passados por valor em parâmetros
- Podem ser usadodo como retorno de uma função

Ver os algortmos parametros-valor.c , parametros-ref1.c, parametros-ref2.c

Tipos enumerados

- Para criar uma variável para armazenar um determinado mês de um ano (de janeiro a dezembro), uma das soluções possíveis é criar um inteiro e armazenar um número associado àquele mês. Assim, janeiro seria o mês número 1, fevereiro o mês número 2, e assim sucessivamente.
- Mas, seria mais simples se pudéssemos escrever no código simplesmente

```
mes = janeiro
```

O comando `enum`

- O comando `enum` cria um tipo enumerado, ou seja, um tipo que funciona como um inteiro, mas para o qual estão associadas constantes numeradas que podem ser utilizadas como constantes inteiras.
- Sua sintaxe é

`enum` $< \textit{nome} >$ { $< \textit{constante}_1 >$, $< \textit{constante}_2 >$, $\cdots$, $< \textit{constante}_N >$, }

O comando enum

- O compilador associa o número zero para o primeiro item e associa para o item i o número $i - 1$. Ex:

```
enum mes { Jan, Fev, Mar, Abr, Mai, Jun,
           Jul, Ago, Set, Out, Nov, Dez};
```

Aqui, janeiro corresponde a 0, fevereiro a 1 e sucessivamente, até dezembro que corresponde ao número 11.

Usando um tipo enumerado

- Declara-se uma variável do tipo enumerado utilizando o nome do tipo que você escolheu. Ex:

```
enum mes mes_aniversario;
```

- Você pode usar o tipo enumerado em qualquer lugar em que você utilizaria um inteiro. Ex:

```
printf (``%d'', mes_aniversario);
```

Veja o exemplo em `enum.c`

Atribuindo valores a um tipo enumerado

- Você pode atribuir um valor inicial para qualquer um dos elementos do tipo enumerado, bastando substituir a $< constante_N >$ por $< constante_N >= valor_n$. Ex:

```
enum mes { Jan = 1, Fev, Mar, Abr, Mai, Jun,
           Jul, Ago, Set, Out, Nov, Dez};
```

Aqui, janeiro corresponde a 1, fevereiro a 2 e sucessivamente, até dezembro que corresponde ao número 12.