

MC-102 — Aula 19

Registros e enumerados

Instituto de Computação – Unicamp

29 de Abril de 2014

Roteiro

- 1 Registros
- 2 Redefinição de tipos
- 3 Tipos Enumerados

Registros

- Um registro é um mecanismo para agrupar várias variáveis, de tipos diferentes ou não, e que dentro de um contexto fazem sentido estarem juntas.
- Exemplos de uso de registros:
 - ▶ Registro de alunos para guardar os dados: (nome, RA, médias de provas, médias de labs, etc...)
 - ▶ Registro de pacientes para guardar os dados: (Nome, endereço, histórico de doenças, etc...)

Declarando um novo tipo de registro

- Para criarmos um novo tipo de registro em C usamos 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;  
};
```

- Cada *nome_i*, é um identificador que será do tipo *tipo_i* (são declarações de variáveis simples).

Exemplo:

```
struct Aluno{  
    char nome[45];  
    int idade;  
    char sexo;  
}; //estamos criando um novo tipo "struct Aluno"
```

Declarando um novo tipo de registro

- A declaração do registro pode ser feita dentro de uma função (como `main`) ou fora dela. Usualmente, ela é feita fora de qualquer função, como no exemplo abaixo:

```
#include <stdio.h>

/* Declare tipos registro aqui */

int main () {
    /* Construa seu programa aqui */
}
```

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 Aluno{
    char nome[45];
    int idade;
    char sexo;
};

int main(){
    struct Aluno a, b; //variáveis a, b são do tipo "struct Aluno"
    .....
}
```

Utilizando os campos de um registro

- Podemos acessar individualmente os campos de uma determinada variável registro como se fossem variáveis normais. A sintaxe é:

`variável_registro.nome_do_campo`

- Os campos individuais de um variável registro tem o mesmo comportamento de qualquer variável do tipo do campo.
 - ▶ Isto significa que todas operações válidas para variáveis de um tipo são válidas para um campo do mesmo tipo.

Utilizando os campos de um registro

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

struct Aluno{
    char nome[45];
    int idade;
    char sexo;
};

int main(){
    struct Aluno a, b;

    strcpy(a.nome, "Helen");
    a.idade = 18;
    a.sexo = 'F';

    strcpy(b.nome, "Dilbert");
    b.idade = 34;
    b.sexo = 'M';

    printf("a.nome = %s, a.idade = %d, a.sexo = %c\n", a.nome, a.idade, a.sexo);
    printf("b.nome = %s, b.idade = %d, b.sexo = %c\n", b.nome, b.idade, b.sexo);
}
```

Lendo e Escrevendo Registros

- A leitura dos campos de um registro a partir do teclado deve ser feita campo a campo, como se fossem variáveis independentes.
- A mesma coisa vale para a escrita, que deve ser feita campo a campo.

```
int main(){
    struct Aluno a, b;

    printf("Digite o nome:");
    scanf("%[^\n]", a.nome);
    printf("Digite a idade:");
    scanf("%d", &a.idade);
    printf("Digite o sexo:");
    getchar(); //usado só para limpar o buffer de entrada
    scanf("%c", &a.sexo);

    printf("a.nome = %s, a.idade = %d, a.sexo = %c\n", a.nome, a.idade, a.sexo);
}
```

Atribuição de registros

- Podemos atribuir um registro a outro diretamente:

```
var1_registro = var2_registro;
```

- É feita uma cópia de cada campo de var2 para var1.

Exemplo:

```
int main(){
    struct Aluno a, b;

    printf("Digite o nome:");
    scanf("%[^\n]", a.nome);
    printf("Digite a idade:");
    scanf("%d", &a.idade);
    printf("Digite o sexo:");
    getchar(); //usado só para limpar o buffer de entrada
    scanf("%c", &a.sexo);

    b = a;
    printf("b.nome = %s, b.idade = %d, b.sexo = %c\n", b.nome, b.idade, b.sexo);
}
```

Vetor de registros

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

- Para declarar: `struct Aluno turma[5];`
- Para usar: `turma[indice].campo;`

```

#include <stdio.h>

struct Aluno {
    int ra;
    double nota;
};

int main (){
    struct Aluno turma[10];
    int i;
    double media;

    for (i = 0; i < 10; i++) {
        printf ("Digite o RA do %dº aluno: ", i);
        scanf ("%d", &turma[i].ra);
        printf ("Digite a média do %dº aluno: ", i);
        scanf ("%lf", &turma[i].nota);
    }

    //calcula a media da turma
    media = 0.0;
    for (i = 0; i < 10; i++) {
        media = media + turma[i].nota;
    }
    media = media/10.0;
    printf("\nA media da turma é: %lf\n",media);
}

```

Redefinido um tipo

- Às vezes, por questão de organização, gostaríamos de criar um tipo próprio nosso, que faz exatamente a mesma coisa que um outro tipo já existente.
- Por exemplo, em um programa onde manipulamos médias de alunos, todas as variáveis que trabalhassem com nota tivessem o tipo `nota`, e não `double`.

O comando typedef

- A forma de se fazer isso é utilizando o comando typedef, seguindo a sintaxe abaixo:

```
typedef <tipo_ja_existente> <tipo_novo>;
```

- Usualmente, fazemos essa declaração fora da função main(), embora seja permitido fazer dentro da função também.
- Ex: `typedef float nota;`
Cria um novo tipo, chamado nota. Variáveis desse tipo serão pontos flutuantes.

Exemplo de uso do typedef

```
#include <stdio.h>

typedef double nota;

int main(){
    nota p1;
    printf("Digite a nota:");
    scanf("%lf",&p1);
    printf("A nota digitada foi: %lf",p1);
}
```

Exemplo de uso do typedef

- Mas o uso mais comum para o comando **typedef** é para a redefinição de tipos registro.
- No nosso exemplo de **struct Aluno**, poderíamos redefinir este tipo para algo mais simples como simplesmente **Aluno**:
 - ▶ **typedef struct Aluno Aluno;**

```

#include <stdio.h>

struct Aluno {
    int ra;
    double nota;
};

typedef struct Aluno Aluno; //redefinimos tipo struct Aluno como Aluno

int main (){
    Aluno turma[10];
    int i;    double media;

    for (i = 0; i < 10; i++) {
        printf ("Digite o RA do %dº aluno: ", i);
        scanf ("%d", &turma[i].ra);
        printf ("Digite a média do %dº aluno: ", i);
        scanf ("%lf", &turma[i].nota);
    }

    //calcula a media da turma
    media = 0.0;
    for (i = 0; i < 10; i++) {
        media = media + turma[i].nota;
    }
    media = media/10.0;
    printf("\nA media da turma é: %lf\n",media);
}

```

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, o código seria mais claro se pudéssemos escrever algo como:

```
mes = janeiro;
```

Tipos Enumerados: (enum)

- O comando `enum` cria um tipo enumerado: podemos usar nomes/identificadores para um conjunto finito de valores inteiros.
- Sua sintaxe é:

```
enum nomeDoTipo { identificador1 , identificador2 , ... ,  
                  identificadorN , };
```

- Exemplo:

```
//criamos um novo tipo chamado "enum meses"  
enum meses {jan, fev, mar, abr, mai, jun, jul,  
            ago, set, out, nov, dez};
```

Tipos Enumerados: `enum`

- O compilador associa o número 0 para o primeiro identificador, 1 para o segundo, etc.
- Variáveis do novo tipo criado são na realidade variáveis inteiras.
- Tipos enumerados são usados para deixar o código mais legível.

```
#include <stdio.h>
//aqui criamos um novo tipo enumerado
//que pode ser usado por qualquer função
enum meses {jan, fev, mar, abr, mai, jun, jul, ago, set,
            out, nov, dec};

int main(){
    enum meses a,b; //aqui criamos 2 variáveis do tipo "enum meses"

    a = jan;
    b = jun;
    if(a != b){
        printf("%d é um mes diferente de %d", a, b);
        //será impresso "0 é um mes diferente de 5"
    }
}
```

Usando um tipo enumerado

- Note que o primeiro identificador recebeu o valor zero, e demais identificadores receberam valores em sequência.
- Podemos alterar o valor inicial dos identificadores.

```
#include <stdio.h>
//aqui criamos um novo tipo enumerado
//que pode ser usado por qualquer função
enum meses {jan = 1, fev, mar, abr, mai, jun, jul, ago, set,
            out, nov, dec};

int main(){
    enum meses a,b; //aqui criamos 2 variáveis do tipo "enum meses"

    a = jan;
    b = jun;
    if(a != b){
        printf("%d é um mes diferente de %d", a, b);
        //será impresso "1 é um mes diferente de 6"
    }
}
```

Tipos Enumerados: Resumindo

- Um tipo enumerado pode ser criado para deixar o código mais legível.
- Variáveis de um tipo enumerado criado são, na realidade, variáveis inteiras, mas temos a versatilidade de atribuir os identificadores do tipo enumerado para tais variáveis.

Exercício

- Crie um programa que pode conter um cadastro de até 100 alunos utilizando structs. As informações a serem armazenadas para cada aluno são o nome do aluno, o RA do aluno e o CPF do aluno. Faça duas funções: uma para adicionar alunos e outra para imprimir todos os alunos cadastrados. Crie um menu de opções para o usuário escolher a função que será executada, e que também permita sair do programa. Para este programa, utilize uma variável global (vetor de structs) para armazenar as informações sobre os alunos.