Introdução a Programação
Registros

Hebert Coelho

Instituto de Informática
Universidade Federal de Goiás

Roteiro

- Redefinição de tipos
- Registros
- Exercícios

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.
- Isso é util quando desenvolvemos programas grandes onde a alteração do tipo de uma determinada variável para outra acarretaria na alteração de muitas variáveis.
- Por exemplo, em um programa onde manipulamos médias de alunos, todas as variáveis que trabalhassem com nota tivessem o tipo nota, e não int ou float.

O comando typedef

- A forma de se fazer isso é utilizando o comando typedef, seguindo a estrutura 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, cujas variáveis desse tipo serão pontos flutuantes.

Exemplo de uso do typedef

```
#include <stdio.h>
typedef float nota; {
main () {
    nota P1;
    printf(“Digite a nota 1”);
    scanf (“%f”, &P1);
    printf(“A nota 1 foi %f”, P1);
}
```

Registro

Definição

Um registro é uma estrutura que contém diversas variáveis (chamadas de campos), usualmente de tipos diferentes, mas que dentro de um determinado contexto, fazem sentido se agrupadas.

- Podemos comparar um registro com uma ficha que possui todos os dados sobre uma determinada entidade, por exemplo: Registro de alunos (nome, RA, médias de provas, médias de labs, etc...)
- Registro de produtos (Nome, código, descrição, etc...)

Declarando o formato do registro

Para criar um registro é preciso declarar seu formato/estrutura. 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;
};
```

Declarando o formato do registro

- O registro pode ser declarado dentro ou fora da função **main**. Normalmente é feita fora da **main**, conforme exemplo.
- A próxima etapa é declarar uma variável do tipo `struct nome do tipo da estrutura`.
- Podemos acessar individualmente os campos de um determinado registro como se fossem variáveis normais.

```
#include <stdio.h>
struct aluno {
    int mat; //matricula
    float media;
};
int main () {
    struct aluno j;
    j.mat = 10;
    j.media = 8.5;
    printf("Matricula" %d, media %f.", j.mat, j.media);
}
```

Utilizando os campos de um Registro

- Para o registro declarado anteriomente, utilizaríamos

```
j.mat
```

para acessar o campo mat do registro j (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 a matricula do aluno:  ”);
scanf (“%d”, &j.mat);
printf (“Digite a média do aluno:  ”);
scanf (“%f”, &j.media);
```

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;
```

Vetor de registros

Registros podem ser declarados como um vetor.

```
#include <stdio.h>
struct aluno {
    int mat; //matricula
    float media;
};
int main () {
    struct aluno turma[40];
    turma[0].mat = 10;
    turma[0].media = 8.5;
    ...
}
```

Registros aninhados

Pode-se também declarar um registro como uma das variáveis de um registro, quantas vezes isso for necessário.

```
#include <stdio.h>
struct medias {
    float p1;
    float p2;
    float p3;
};
typedef struct medias Medias;
struct ficha {
    int matricula;
    Medias provas;
};
```

Registros

Resolução de exercícios.