Registros

C3

Universidade Federal de Rio Grande
Centro de Ciências Computacionais
ICC - Introdução a Ciência da Computação

8 de outubro de 2015

Objetivo

Conceituar tipos de dados definidos pelo usuário. Utilizar `struct` e `typedef` para criar registros.

Motivação

Dados que são relacionados podem ser salvos em uma única estrutura ou registro.

Registros

- Registros permitem a criação de novos **tipos** de variáveis.
- Registros são tipos de variáveis que agrupam dados geralmente desiguais, porém relacionados.
  - Vetores são tipos de variáveis que agrupam dados similares.
- Os itens de dados de um registro são chamados **campos**.
- Um registro pode ter um ou mais campos de diferentes tipos.

Declaração

- Para se criar um registro, utiliza-se o comando `struct`. Sua forma geral é:

```
struct nome_do_registro {
  tipo_1 campo_1;
  tipo_2 campo_2;
  tipo_3 campo_3;
  ...
  tipo_n campo_n;
};
```

Exemplo

- Exemplo para se criar um registro para cadastro de uma pessoa.

```
/* Declaracao do registro */
/* Criacao de um novo tipo */
struct pessoa {
   char nome[20];
   int idade;
   int RG;
};

/* Declaracao de variaveis do tipo criado */
struct pessoa p1;
struct pessoa nova_pessoa;
```

Para acessar

Para se acessar um campo de um registro, digita-se o nome da variável que é do tipo `registro`, por exemplo, `p1` seguida de um ponto, seguido do nome do campo. Exemplo:

```
struct pessoa p1;
struct pessoa p2;

strcpy(p1.nome,"Joao");
p1.idade = 50;
p1.RG = 567891011;
printf(" Nome: %s\n Idade: %d\n RG: %d\n",
                p1.nome,p1.idade, p1.RG);

strcpy(p2.nome,"Maria");
p2.idade = 15;
p2.RG = 32133122;
printf(" Nome: %s\n Idade: %d\n RG: %d\n",
                p2.nome,p2.idade, p2.RG);
```

Para acessar

Os comandos anteriores irão imprimir:

```
Nome: Joao
Idade: 50
RG: 567891011

Nome: Maria
Idade: 15
RG: 32133122
```

typedef

Determina um novo nome para um tipo. Esse tipo pode ser uma `struct` ou mesmo um tipo simples em C (e.g. `int`).
Exemplo:

```
typedef struct pessoa {
  char nome[20];
  int idade;
  int RG;
} tp_pessoa;
```

Onde `tp_pessoa` agora é o novo tipo. Portanto as variáveis declaradas vão ser do tipo `tp_pessoa`. Ex.:

```
tp_pessoa p1, p2;
```

typedef - Exemplo

```c
#include <stdio.h>

typedef struct _venda {
  int cod_peca;
  float preco;
} tp_venda ;

int main (){
  tp_venda A, B;

  A.cod_peca = 1;
  A.preco = 99.99;

  printf("Codigo Peca: %d \nPreco: %.2f reais\n", A.cod_peca, A.preco);

  B.cod_peca = 2;
  B.preco= 49.50;

  printf("Codigo Peca: %d \nPreco: %.2f reais\n", B.cod_peca, B.preco);

  return 0;
}
```

Atribuição Direta

Em C, pode-se fazer atribuição diretas de registros. Exemplo:

```c
#include <stdio.h>

typedef struct _venda {
  int cod_peca;
  float preco;
} tp_venda ;

int main (){
  tp_venda A, B;

  A.cod_peca = 1;
  A.preco = 99.99;

  printf("Codigo Peca: %d \nPreco: %.2f reais\n", A.cod_peca, A.preco);

  /**** Atribuicao direta ****/
  B = A;

  printf("Codigo Peca: %d \nPreco: %.2f reais\n", B.cod_peca, B.preco);

  return 0;
}
```

Funções e Registros

Registros podem ser passados como parâmetros e também podem ser retornados através do comando `return`. Exemplo

```
#include <stdio.h>

typedef struct _venda {
  int cod_peca;
  float preco;
} tp_venda ;

tp_venda novavenda( int codigo, float preco){
  tp_venda A;
  A.cod_peca = codigo;
  A.preco = preco;
  return A;
}

void imprimeVenda(tp_venda C) {
  printf("Codigo da Peca: %d\nPreco: %.2f reais\n\n", C.cod_peca, C.preco);
}

int main (){
  tp_venda V1, V2;

  V1 = novavenda(5,100);
  V2 = novavenda(7,200);

  imprimeVenda(V1);
  imprimeVenda(V2);

  return 0;
}
```

Vetor de Registros e Registros Aninhados

```c
#include <stdio.h>
#define TAM 1000

typedef struct _endereco {
  int numero;
  char rua[20];
  int CEP;
} tp_endereco;

typedef struct _pessoa {
  char nome[20];
  tp_endereco endereco;
} tp_pessoa;

int main(){
  tp_pessoa pessoas[TAM];
  int i;
  for (i=0;i<TAM;i++){
    printf("\n Nome: ");
    scanf("%s",pessoas[i].nome);
    printf("    Rua: ");
    scanf("%s",pessoas[i].endereco.rua);
    printf("    Numero: ");
    scanf("%d",&pessoas[i].endereco.numero);
    printf("    CEP: ");
    scanf("%d",&pessoas[i].endereco.CEP);
  }

  /* Mostrar todas as Pessoas que moram
     em casas impares */

  for (i=0;i<TAM;i++)
    if (pessoas[i].endereco.numero%2){
      printf("\n\n Nome: %s",
             pessoas[i].nome);
      printf("\n    Rua: %s",
             pessoas[i].endereco.rua);
      printf("\n    Numero: %d",
             pessoas[i].endereco.numero);
      printf("\n    CEP: %d",
             pessoas[i].endereco.CEP);
    }
  return 0;
}
```

Vetor de registros e registros aninhados

Mesmo exemplo usando funções.

```c
#include <stdio.h>
#define TAM 1000

typedef struct _endereco {
  int numero;
  char rua[30];
  int CEP;
} tp_endereco;

typedef struct _pessoa {
  char nome[50];
  tp_endereco endereco;
} tp_pessoa;

tp_pessoa lePessoa(){
  tp_pessoa p;

  printf("\n Nome: ");
  scanf("%s",p.nome);
  printf("   Rua: ");
  scanf("%s",p.endereco.rua);
  printf("    Numero: ");
  scanf("%d",&p.endereco.numero);
  printf("    CEP: ");
  scanf("%d",&p.endereco.CEP);

  return p;
}

void mostraPessoa(tp_pessoa p){
  printf("\n\n Nome: %s",p.nome);
  printf("\n    Rua: %s",p.endereco.rua);
  printf("\n    Numero: %d",p.endereco.numero);
  printf("\n    CEP: %d",p.endereco.CEP);
}

int main(){
  tp_pessoa pessoas[TAM];
  int i;

  for (i=0;i<TAM;i++){
    pessoas[i]=lePessoa();
  }

  /* Mostrar todas as Pessoas
     que moram em casas impares */
  for (i=0;i<TAM;i++)
    if (pessoas[i].endereco.numero % 2 == 1){
      mostraPessoa(pessoas[i]);
    }

  return 0;
}
```

Exercício 01

Faça um programa que leia nome, peso, altura de 1000 pessoas, e depois mostre o nome das pessoas que estão acima do peso, ou seja, que possuem $IMC > 30$. Onde $IMC = \frac{peso}{altura^2}$. Use um registro para associar o peso e a altura de cada pessoa.

Resposta Exercício 01

```c
#include<stdio.h>
#define TAM 3

typedef struct _pessoa {
  char nome[20];
  float peso;
  float altura;
} tp_pessoa;

int IMC(tp_pessoa A) {
  return (float)(A.peso/(A.altura*A.altura));
}

int main(){
  tp_pessoa pessoas[TAM];
  int i;
  for (i=0;i<TAM;i++){
    printf("\n Nome: ");
    scanf("%s",pessoas[i].nome);
    printf("    Peso: ");
    scanf("%f",&pessoas[i].peso);
    printf("    Altura: ");
    scanf("%f",&pessoas[i].altura);
  }

  /* Mostra todas as Pessoas que estao acima do peso */
  printf("\n\n     ***** Pessoas acima do peso *****\n");
  for (i=0;i<TAM;i++)
    if (IMC(pessoas[i])>30)
      printf("             Nome: %s\n",pessoas[i].nome);
  return 1;
}
```

Referências

- ASCENCIO, Ana F. G.; CAMPOS, Edilene A. V. Fundamentos da programa de computadores. Pearson Prentice Hall, 2007.
- Schildt, Herbert. C completo e total. Pearson Makron Books, 1997.
- HUSS, E. The C Library Reference Guide. Disponível em `http://www.acm.uiuc.edu/webmonkeys/book/c_guide`
- Notas de aula do Prof. Flavio Keidi Miyazawa.
- Notas de aula do Prof. Emanuel Estrada.
- Notas de aula do Prof. Alessandro Bicho.