Linguagem C: Tipos Enumerativos e Estruturas

Prof. Paulo R. S. L. Coelho
paulo@facom.ufu.br

Faculdade de Computação
Universidade Federal de Uberlândia

Organização

Organização

Introdução

- Até então, os valores manipulados pelos programas foram números inteiros, números reais e caracteres.
- Além disso, usando a diretiva `#define`, constantes simbólicas puderam ser definidas.
- A linguagem C (e outras) oferecem artifícios para trabalhar com valores diferentes daqueles vistos até então, sem precisar de diretivas.
- Esses artifícios são os **tipos enumerativos**.

Declaração - I

- A criação de um tipo enumerativo consiste em declarar 2 coisas: um novo tipo de variável escalar e valores a serem atribuídos a variáveis desse tipo.
- Para declarar tipos enumerativos em C, utilizamos a palavra reservada `enum`.
- Exemplo:

```
enum diasemana {dom, seg, ter, qua, qui, sex, sab};
```

- Definido o tipo, varáveis podem ser declaradas como sendo desses tipos. Por exemplo:

```
...
enum diasemana hoje, ontem, amanha;
...
if (ontem == seg) {
  hoje = ter;
  amanha = qua;
}
```

Declaração - II

- Os identificadores usados para os valores da declaração de um tipo enumerativo são constantes inteiras.
- A palavra reservada `enum` é obrigatória em tais declarações, mas pode-se dispensá-la mediante declarações de tipo, usando a diretiva `typedef`.
- Por exemplo:

```
typedef enum diasemana diasem;
enum diasemana {dom, seg, ter, qua, qui, sex, sab};

diasem hoje, ontem, amanha;

if (ontem == 1) { //seg
  hoje = ter;
  amanha = qua;
}
```

Exemplo

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef enum diasemana diasem;
enum diasemana {dom, seg, ter, qua, qui, sex, sab};

int main() {
  diasem hj, ontem, amanha;
  int i;
  char hojeStr[4];
  char nomedia[7][4] = {"dom","seg","ter","qua","qui","sex","sab"};
  printf("entre dia de hoje: "); scanf("%s", hojeStr);
  hj = -1;
  for (i = 0; i <=6; i++)        //calculo do dia de hj
    if (strcmp(hojeStr, nomedia[i]) == 0)
      hj = i;

  if (hj != -1) {
    ontem = (hj + 6) % 7;
    amanha = (hj + 1) % 7;
    printf("\n\nHoje eh %s; ontem foi %s; amanha serah %s.\n\n",
         nomedia[hj], nomedia[ontem], nomedia[amanha]);
  } else
    printf("\n\nDia da semana inválido!\n\n");
  return 0;
}
```

Organização

Introdução I

- Em muitas aplicações, é útil a capacidade de se tratar todas as informações de uma entidade (uma pessoa, por exemplo), como sendo uma única unidade de armazenamento, ou seja, uma única variável.
- Além disso, pode ser importante a capacidade de tratar cada informação em separado.
- Por exemplo, seja o cadastramento de empregados de uma empresa, em que cada um tem as seguintes informações: nome, endereço, telefone, sexo, estado civil, cargo, setor, salário;
- Algumas dessas informações ainda englobam outros conjuntos de informações.

Introdução II

- Por exemplo, o endereço é composto de: logradouro, número, complemento, cidade, estado, país.
- Observa-se que, diferentemente de variáveis indexadas (vetores e matrizes), não há homogeneidade quanto ao tipo de dados tratado.
- Por isso, deve haver um mecanismo para trabalhar com variáveis estruturadas heterogêneas.

Declaração - I

- Na linguagem C, a declaração `struct` é destinada à declaração de variáveis não-homogêneas.
- As informações a serem armazenadas numa variável desse tipo são especificadas entre as chaves, dentro das quais são declarados os **campos** da estrutura.
- Por exemplo:

```
struct endereco {
  char logradouro[15];
  int numero;
  char complemento[6], bairro[10];
  char cidade[10], estado[3], pais[10];
};
...
struct endereco end1, end2, end3;
```

Declaração - II

- A palavra reservada `struct` é obrigatória em tais declarações, mas pode-se dispensá-la mediante declarações de tipo, usando a diretiva `typedef`.
- Por exemplo:

```
typedef struct endereco end;
struct endereco {
  char logradouro[15];
  int numero;
  char complemento[6], bairro[10];
  char cidade[10], estado[3], pais[10];
};
...
end end1, end2, end3;
```

Acesso aos Campos de Uma Estrutura

- Para acessar um dos campos de uma estrutura, usa-se o operador '.' (ponto).
- Diferentemente de variáveis indexadas, variáveis do tipo estrutura podem receber o valor de outra do mesmo tipo.
- Por exemplo:

```
typedef struct endereco end;
struct endereco {
  char logradouro[30];
  int numero;
  char bairro[10];
};
...
end loja = {"R. dos Ipes", 1200, "Centro"}, minhaCasa, lojaCopia;
strcpy(minhaCasa.logradouro, "Av. Tiradentes");
end.numero = 999;
strcpy(end.bairro, "Centro");
lojaCopia = loja;
...
```

Exemplo

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

typedef struct compl complexo;
struct compl {
  float real, imag;
};

int main() {
  int i, j;
  complexo A[3][3] = {
                       { {1.0, -0.1}, {2.0, -0.2}, {2.0, -0.2} },
                       { {4.0, -3.4}, {5.0, 4.1}, {6.0, -2.6} }
                     };
  for ( i= 0; i < 3; i++) {
    for (j = 0; j < 3; j++)
      printf("(%5.1f) + i(%5.1f)    ", A[i][j].real, A[i][j],imag);
    printf("\n");
  }
  return 0;
}
```

Organização

Exercícios I

1. Faça um programa que leia informações sobre 15 pessoas. Essa informação deve ficar em um vetor de variáveis do tipo estruturado `pessoa`, o qual deve conter as seguintes informações:
   - Nome: string de tamanho 30;
   - Sexo: tipo enumerativo com os valores `masc`, `fem`;
   - Idade: valor inteiro;
   - Estado Civil: tipo enumerativo com os valores `solteiro`, `casado`, `separado`, `viúvo`.
   - Salário: valor real.

   Em seguida, imprima o número de homens, número de mulheres e informações da pessoa com maior salário.

Exercícios II

2. Faça um programa que leia o nome, duas notas e número de faltas de 10 alunos. As informações desses alunos devem ser armazenadas em um vetor de variáveis do tipo estruturado `aluno`, o qual deve conter as seguintes informações de cada aluno:
   - Nome: string de tamanho 30;
   - Média: número real resultado da média das duas notas lidas;
   - Situação: caractere representando situação, isto é, 'A' (Aprovado), se média maior ou igual a 6 e número de faltas menor que 10, e 'R' (Reprovado), caso contrário.
   - Faltas: número de faltas (valor inteiro).

   Por fim, devem ser impressas as informações de cada aluno.

Respostas I

1.
```
#include <stdlib.h>
#include <stdio.h>

typedef enum sex tSexo;
enum sex {masc, fem};

typedef enum estCiv estado;
enum estCiv {solteiro, casado, separado, viuvo};

typedef struct pes pessoa;
struct pes {
  char nome[30];
  tSexo sexo;
  int idade;
  estado estadoCivil;
  float salario;
};

int main() {
  pessoa P[15], maisRica;
  int i, nH, nM;
  float maiorSal = 0;
  char sexoString[2][5] = {"masc", "fem"};
```

Respostas II

```
for (i = 0; i < 15; i++) {
  printf("\nentre nome: ");
  scanf("%s", P[i].nome);
  printf("entre sexo (0: Masc, 1: Fem): ");
  scanf("%d", &P[i].sexo);
  printf("entre idade: ");
  scanf("%d", &P[i].idade);
  printf("entre estado civil (0: solteiro, 1: casado,
                              2: separado, 3: viuvo): ");
  scanf("%d", &P[i].estadoCivil);
  printf("entre salario: ");
  scanf("%f", &P[i].salario);
  if (P[i].sexo == masc) nH++;
  if (P[i].sexo == fem) nM++;
  if (P[i].salario > maiorSal) {
    maiorSal = P[i].salario;
    maisRica = P[i];
  }
}

printf("\n\nNumero de homens: %d", nH);
printf("\n\nNumero de mulheres: %d", nM);
```

Respostas III

```
  printf("\n\nPessoa mais rica: \n\tNome: %s", maisRica.nome);
  printf("\n\tSalario: %8.2f", maisRica.salario);
  printf("\n\tSexo: %s\n",sexoString[maisRica.sexo]);

  return 0;
}
```

2.
```
#include <stdio.h>
#include <stdlib.h>

typedef struct aluno aluno;
struct aluno {
  char nome[30];
  float media;
  char situacao;
  int faltas;
};

int main() {
  aluno A[10];
  int i;
  float nota1, nota2;

  for (i = 0; i < 10; i++) {
```

Respostas IV

```
    printf("\nentre nome do aluno: ");
    scanf("%s", A[i].nome);
    printf("entre primeira nota: ");
    scanf("%f", &nota1);
    printf("entre segunda nota: ");
    scanf("%f", &nota2);
    printf("entre numero de faltas: ");
    scanf("%d", &A[i].faltas);
    A[i].media = (nota1 + nota2) / 2;
    if (A[i].media >= 6.0 && A[i].faltas < 10)
      A[i].situacao = 'A';
    else
      A[i].situacao = 'R';
  }
  printf("\n\nInformações dos alunos:");
  for (i = 0; i < 10; i++) {
    printf("\n\nNome: %s\n", A[i].nome);
    printf("Media: %3.1f\n", A[i].media);
    printf("Faltas: %3d\n", A[i].faltas);
    printf("Situacao: %c\n", A[i].situacao);
  }
  return 0;
}
```