Técnicas de Programação

Luiz Fernando Carvalho

luizfcarvalhoo@gmail.com

Variáveis heterogêneas

- Em cada parte de um programa geralmente há várias variáveis associadas à realização de uma tarefa específica;
- Por causa disso, é conveniente ter um modo de agrupar um conjunto de variáveis relacionadas;
- Vetores e matrizes agrupam uma série de variáveis do **MESMO TIPO**, cada uma identificada por índices;
- Se, por outro lado, quisermos um tipo de agrupamento que englobe variáveis de **TIPOS DIFERENTES**, ou no qual cada variável possa ser identificada por um nome específico usamos **STRUCTS**.

Struct / registro

- Usamos um tipo de estrutura chamado de **registro** (mais conhecido por seu nome em inglês, ***struct***, uma abreviação de *structure*, 'estrutura');
- Esse recurso da linguagem C permite que o usuário "defina" **seus próprios tipos de dados** a partir dos tipos primitivos da linguagem (*int*, *float*, *char*, etc.);
- **Struct** contém um conjunto de variáveis, que têm tipos fixados e são identificadas por nomes (como as variáveis comuns);

Definição de Struct

- Por exemplo, uma **struct** que representa uma pessoa pode conter as seguintes variáveis:

```
char nome[50];
int idade;
float peso;
float altura;
```

- Essas variáveis são denominadas de membros da struct.
- Cada **struct** é em si uma variável!!!

Definição de Struct II

- Por exemplo, uma **struct** que representa um produto numa compra pode conter as seguintes variáveis:

```
char descricao[30];
int quantidade;
float preco_unitario;
float desconto;
float preco_total;
```

- Essas variáveis são denominadas de membros da struct.
- Cada **struct** é em si uma variável!!!

Declaração de Struct

- Uma **struct/registro** é declarada usando a palavra chave **struct** seguida de um bloco (delimitado por chaves) contendo as declarações dos membros, como se fossem declaração de variáveis comuns.

```
struct Produto{
    char descricao[30];
    int quantidade;
    float preco_unitario;
    float desconto;
    float preco_total;
};
```

Definição da struct

Ponto e vírgula
(definição da struct é um comando)

```
struct Produto produto_1, produto_2, produto_3;
```

Declaração de 3 variáveis do tipo Produto

Acesso aos membros

- Para acessar campos de um registro, usamos o operador . (um ponto), colocando à esquerda dele o nome da variável que contém o registro, e à direita o nome do campo.
- No exemplo anterior:

```
struct Produto{
    char descricao[30];
    int quantidade;
    float preco_unitario;
    float desconto;
    float preco_total;
};
```

```
struct Produto produto_1, produto_2, produto_3;
```

- Pode-se acessar o preço do produto_1 usando a expressão:

```
produto_1.preco_unitario
```

Resumo

- Declaração de um tipo de registro

```
struct nome_do_tipo{
    /* declarações dos membros */
};
```

- Declaração de um registro

```
nome_da_struct nome_da_variável;
```

- Acesso de membros de um registro

```
variavel.nome_do_membro;
```

Exemplo

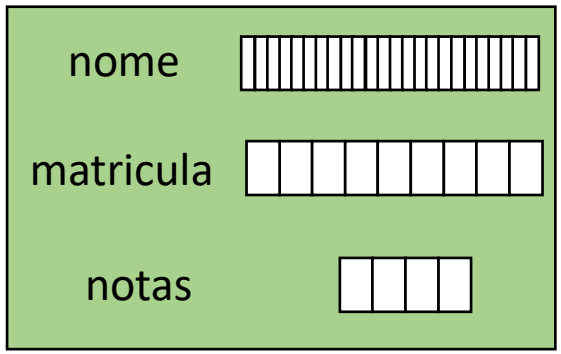
```c
#include<stdio.h>
#include<stdlib.h>

struct Aluno {
    char nome[30];
    char matricula[10];
    float notas[4];
};

int main()
{
    struct Aluno a;
    int i;

    printf("Qual o nome do aluno? ");
    fgets(a.nome, 30, stdin);

    printf("Qual a matricula? ");
    fgets(a.matricula, 10, stdin);

    for(i=0;i<4;i++)
    {
        printf("Digite a nota do %d bimestre: ", i+1);
        scanf("%f", &a.notas[i]);
    }

    return 0;
}
```

Nenhuma variável foi criada até esse momento!!!

Definição da **struct**

Declarando/criando uma variável **a** do "tipo" `Aluno`

a



Inicializando da Struct

- Pode-se inicializar a **`struct`** de forma estática, como ocorre com variáveis comuns, vetores e matrizes;

```
struct Produto{
   char descricao[30];
   int quantidade;
   float preco_unitario;
   float desconto;
   float preco_total;
};

struct Produto agua_sanitaria = {“Qboa”, 1, 7.99, 0, 7.99};
```

```
agua_sanitaria.descricao = “Qboa”;
agua_sanitaria.quantidade = 1;
agua_sanitaria.preco_unitario = 7.99;
agua_sanitaria.desconto = 0;
agua_sanitaria.preco_total = 7.99;
```

Inicializando da Struct

- Pode-se inicializar a *struct* de forma estática, como ocorre com variáveis comuns, vetores e matrizes;

```
struct Aluno{
    char nome[30];
    char matricula[10];
    float notas[4];
};

struct Aluno a = {“Joao”, “123456789”, 7.8, 5.4, 6.1, 9.3};
```

```
a.nome = “Joao”;
a.matricula = “123456789”;
a.notas[0] = 7.8;
a.notas[1] = 5.4;
a.notas[2] = 6.1;
a.notas[3] = 9.3;
```

Atribuição entre structs

- Uma variável estrutura pode ser atribuída a outra do mesmo tipo por meio de uma atribuição simples

```
struct Produto{
   char descricao[30];
   int quantidade;
   float preco_unitario;
   float desconto;
   float preco_total;
};

struct Produto feijao = {“redondo”, 1, 20.0, 0, 20.0};
struct Produto feijao_carioca;

feijao_carioca = feijao;
```

Atribuição só pode ser feita com **structs** do mesmo tipo

Comando Typedef

- É possível nomear um tipo baseado em uma estrutura
  - Para isso utiliza-se `typedef` na declaração

```
typedef struct{
   char descricao[30];
   int quantidade;
   float preco_unitario;
   float desconto;
   float preco_total;
}Mercadoria;

int main(){
   Mercadoria feijao = {"redondo", 1, 20.0, 0, 20.0};
   Mercadoria feijao_carioca;

   feijao_carioca = feijao;
```

Mercadoria é o nome do tipo

Agora não existe mais a necessidade de:

struct Mercadoria feijao;

Comando Typedef

- É possível nomear um tipo baseado em uma estrutura
  - Para isso utiliza-se `typedef` na declaração

```
typedef struct{
   int dia, mes, ano;
}Data;


int main()
{
   Data atual;

   return 0;
}
```

Comando Typedef

```
typedef struct{
   int dia, mes, ano;
}Data;


int main()
{
   Data atual;

   return 0;
}
```

São equivalentes

```
struct Data{
   int dia, mes, ano;
};


int main()
{
   struct Data atual;

   return 0;
}
```

Exercícios

1. Escreva um trecho de código para fazer a criação dos novos tipos de dados conforme solicitado abaixo:
   - **Horário:** composto de hora, minutos e segundos.
   - **Data:** composto de dia, mês e ano.
   - **Compromisso:** local, horário e texto que descreve o compromisso.
2. Construa uma estrutura aluno com nome, curso e 4 notas, média e situação. Leia as informações nome, curso e notas do teclado, calcule a média e armazene a situação do aluno.
   - media ≥ 7 → Aprovado;
   - 3 ≤ media < 7 → Exame;
   - media < 3 → reprovado;

Exercícios

3. Considerando a estrutura

```
typedef struct {
    float x;
    float y;
    float z;
}Vetor;
```

Para representar um vetor em $\mathbb{R}^3$, implemente um programa que calcule a soma de dois vetores.

$$\vec{u} = (-2, 2, -4)$$
$$\vec{v} = (0, 4, -5)$$
$$\vec{u} + \vec{v} = (-2, 6, -9)$$

Exercícios

4. Faça um programa que controla o consumo de energia dos eletrodomésticos de uma casa e:
   - Crie e leia 5 eletrodomésticos que contém nome (máximo 15 letras), potência (real, em kW) e tempo ativo por dia (real, em horas).
   - Leia um tempo t (em dias), calcule e mostre o consumo total na casa e o consumo relativo de cada eletrodoméstico (consumo/consumo total) nesse período de tempo. Apresente este último dado em porcentagem.

Exercícios

5. Faça um programa que realize operações simples de números complexos:
   - leia dois números complexos z e w, compostos por parte real e parte imaginaria.
   - apresente a soma, subtração e produto entre z e w, nessa ordem, bem como o módulo de ambos.