

# Registros

Profa. Graziela Santos de Araújo

Faculdade de Computação  
Universidade Federal de Mato Grosso do Sul

Algoritmos e Programação II



# Conteúdo da aula

- 1 Introdução
- 2 Definição, declaração e uso
- 3 Declaração e inicialização simultâneas
- 4 Operações sobre registros
- 5 Exemplo
- 6 Exercícios



# Introdução

- ▶ aprendemos a utilizar variáveis compostas homogêneas, agrupando valores de um mesmo tipo em uma única entidade lógica
- ▶ a linguagem C dispõe também de uma outra forma para agrupamento de dados, chamada **variável composta heterogênea, registro** ou **estrutura**
- ▶ nos registros, podemos armazenar sob uma mesma entidade lógica valores de tipos diferentes



# Definição, declaração e uso

- ▶ uma **variável composta heterogênea** ou **registro** é uma estrutura onde podemos armazenar valores de tipos diferentes sob uma mesma entidade lógica
- ▶ cada um desses possíveis valores é armazenado em um compartimento do registro denominado **campo do registro**, ou apenas **campo**
- ▶ um registro é composto pelo seu identificador e pelos seus campos

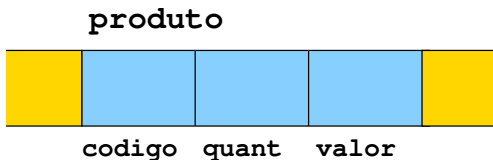


# Definição, declaração e uso

- suponha que queremos trabalhar com um agrupamento de valores que representam uma determinada mercadoria de uma loja.

```
struct {  
    int codigo;  
    int quant;  
    float valor;  
} produto;
```

- a figura abaixo mostra a disposição do registro **produto** na memória do computador.



# Definição, declaração e uso

- ▶ formato geral de declaração de um registro:

```
struct {  
    :  
    bloco de declarações  
    :  
} identificador;
```

- ▶ podemos declarar outras variáveis do tipo registro com os mesmos campos da variável **produto**:

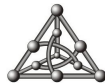
```
struct {  
    int codigo;  
    int quant;  
    float valor;  
} produto, estoque, baixa;
```



# Definição, declaração e uso

- ▶ diferentemente da atribuição de um valor a uma variável ou a um compartimento de uma variável composta homogênea, a atribuição de um valor a um campo de um registro é realizada através do acesso a esse campo, especificando:
  1. o identificador do registro
  2. um ponto
  3. o identificador do campo
- ▶ por exemplo:

```
produto.codigo = 12;  
produto.quant = 5;  
produto.valor = 34.5;
```



# Definição, declaração e uso

- ▶ também podemos usar o valor de um campo de um registro em quaisquer expressões:

```
if (produto.valor < 150.0)
    printf("Comprar produto\n");
else
    printf("Acima do preço de mercado!\n");
```





# Definição, declaração e uso

- Declarações de registros diferentes podem conter campos com mesmo identificador:

```
struct {  
    char tipo;  
    char fatorRH;  
    int idade;  
    float altura;  
} coleta;  
  
struct {  
    char codigo;  
    int tipo;  
    int idade;  
} certidao;
```

```
coleta.tipo = '0';  
certidao.tipo = 0;  
coleta.idade = 29;  
certidao.idade = coleta.idade + 2;
```



# Declaração e inicialização simultâneas

- ▶ as regras são idênticas às dos vetores:

```
struct {  
    int codigo;  
    int quant;  
    float valor;  
} produto = {1, 5, 34.5};
```

- ▶ se uma inicialização não contém valores suficientes para todos os campos, então o restante dos campos é inicializado com 0 (zero)

```
struct {  
    int codigo;  
    int quant;  
    float valor;  
} produto = {0};
```



# Operações sobre registros

- ▶ quando queremos copiar o conteúdo completo de todos os compartimentos de uma variável composta **homogênea** para outra, é necessário realizar a cópia elemento a elemento.
- ▶ então, se **A** e **B** são, por exemplo, vetores de um mesmo tipo de dados e mesma dimensão, é **errado** tentar fazer uma atribuição como abaixo:

**A = B;**



# Operações sobre registros

- ▶ com registros, podemos fazer uma atribuição direta e realizar a cópia de todos os seus campos nessa única atribuição:

```
⋮  
struct {  
    char tipo;  
    int  codigo;  
    int  quant;  
    float valor;  
} mercadoria1, mercadoria2;  
⋮  
mercadoria1.tipo = 'A';  
mercadoria1.codigo = 10029;  
mercadoria1.quant = 62;  
mercadoria1.valor = 10.32 * TAXA + 0.53;  
⋮  
mercadoria2 = mercadoria1;  
⋮
```



# Exemplo 1

```
#include <stdio.h>

/* Recebe um horário no formato hh:mm:ss e o atualiza
   em 1 segundo, mostrando o novo horário na saída */
int main()
{
    struct {
        int hh;
        int mm;
        int ss;
    } agora, prox;

    printf("Informe o horário atual (hh:mm:ss): ");
    scanf("%d:%d:%d", &agora.hh, &agora.mm, &agora.ss);
```



# Exemplo 1

```
prox = agora;

prox.ss = prox.ss + 1;
if (prox.ss == 60) {
    prox.ss = 0;
    prox.mm = prox.mm + 1;
    if (prox.mm == 60) {
        prox.mm = 0;
        prox.hh = prox.hh + 1;
        if (prox.hh == 24)
            prox.hh = 0;
    }
}

printf("Próximo horário é %d:%d:%d\n", prox.hh, prox.mm, prox.ss);

return 0;
}
```



## Exemplo 2

### FORMA COMO VAMOS USAR!

```
#include <stdio.h>

struct tipoHora{
    int hh;
    int mm;
    int ss;
};

/* Recebe um horário no formato hh:mm:ss e o atualiza
   em 1 segundo, mostrando o novo horário na saída */
int main()
{
    tipoHora agora, prox;

    printf("Informe o horário atual (hh:mm:ss): ");
    scanf("%d:%d:%d", &agora.hh, &agora.mm, &agora.ss);
```



## Exemplo 2

```
prox = agora;

prox.ss = prox.ss + 1;
if (prox.ss == 60) {
    prox.ss = 0;
    prox.mm = prox.mm + 1;
    if (prox.mm == 60) {
        prox.mm = 0;
        prox.hh = prox.hh + 1;
        if (prox.hh == 24)
            prox.hh = 0;
    }
}

printf("Próximo horário é %d:%d:%d\n", prox.hh, prox.mm, prox.ss);

return 0;
}
```





# Exercícios

1. Dada uma data no formato **dd/mm/aaaa**, escreva um programa que mostre a próxima data, isto é, a data que representa o dia seguinte à data fornecida.

Exemplo:

*Se a data fornecida é **30/06/2011** a saída deve ser **01/07/2011**.*

*Importante!* Não esqueça dos anos bissextos. Lembre-se que um ano é bissexto se é divisível por 400 ou, em caso negativo, se é divisível por 4 mas não por 100.



## Veja arquivo .cpp anexado!

```
#include <stdio.h>

/* Recebe uma data no formato dd:mm:aaaa e a atualiza em 1 dia, mostrando a nova data na saída */
int main()
{
    struct {
        int dia;
        int mes;
        int ano;
    } data, prox;

    printf("Informe uma data (dd/mm/aa): ");
    scanf("%d/%d/%d", &data.dia, &data.mes, &data.ano);

    prox = data;
    prox.dia++;
```



# Exercícios

```
if (prox.dia > 31 ||
    (prox.dia == 31 && (prox.mes == 4 || prox.mes == 6 ||
                        prox.mes == 9 || prox.mes == 11)) ||
    (prox.dia == 30 && prox.mes == 2) ||
    (prox.dia == 29 && prox.mes == 2
     && (prox.ano % 400 != 0 &&
          (prox.ano % 100 == 0 ||
           prox.ano % 4 != 0)))) {

    prox.dia = 1;
    prox.mes++;
    if (prox.mes > 12) {
        prox.mes = 1;
        prox.ano++;
    }
}

printf("%02d/%02d/%02d\n", prox.dia, prox.mes, prox.ano);

return 0;
}
```



# Exercícios

2. Dados dois horários de um mesmo dia, expressos no formato **hh:mm:ss**, calcule o tempo decorrido entre estes dois horários, apresentando o resultado no mesmo formato **hh:mm:ss**.

Exemplo:

*Se os dois horários fornecidos são **07:13:22** e **13:05:56**, a resposta tem de ser **05:52:34**.*



# Exercícios

3. Dadas duas datas no formato **dd/mm/aaaa**, calcule o número de dias decorridos entre estas duas datas.

Exemplo:

*Se as duas datas fornecidas são **01/03/2007** e **23/09/2001**, a resposta deve ser 1985.*

Uma maneira provavelmente mais simples de computar essa diferença é usar a fórmula 1 para calcular um número de dias  $N$  baseado em uma data:



# Exercícios

3.

$$N = \left\lfloor \frac{1461 \times f(\text{ano}, \text{mês})}{4} \right\rfloor + \left\lfloor \frac{153 \times g(\text{mês})}{5} \right\rfloor + \text{dia} \quad (1)$$

onde

$$f(\text{ano}, \text{mês}) = \begin{cases} \text{ano} - 1, & \text{se } \text{mês} \leq 2, \\ \text{ano}, & \text{caso contrário} \end{cases}$$

e

$$g(\text{mês}) = \begin{cases} \text{mês} + 13, & \text{se } \text{mês} \leq 2, \\ \text{mês} + 1, & \text{caso contrário.} \end{cases}$$

Lembre-se ainda que o **piso de um número real**  $x$ , denotado por  $\lfloor x \rfloor$ , é o maior número inteiro menor ou igual a  $x$ . Ou seja,  $\lfloor 5.3 \rfloor = 5$ ,  $\lfloor 12.999 \rfloor = 12$  e  $\lfloor 2 \rfloor = 2$ .

Podemos calcular o valor  $N_1$  para a primeira data informada, o valor  $N_2$  para a segunda data informada e a diferença  $|N_2 - N_1|$  é o número de dias decorridos entre estas duas datas informadas.



## Exercícios

4. Seja  $N$  computado como na equação 1. Então, o valor

$$D = (N - 621049) \bmod 7$$

é um número entre 0 e 6 que representa os dias da semana, de domingo a sábado. Por exemplo, para a data de 21/06/2007 temos

$$\begin{aligned} N &= \left\lfloor \frac{1461 \times f(2007, 6)}{4} \right\rfloor + \left\lfloor \frac{153 \times g(6)}{5} \right\rfloor + 21 \\ &= \left\lfloor \frac{1461 \times 2007}{4} \right\rfloor + \left\lfloor \frac{153 \times 7}{5} \right\rfloor + 21 \\ &= 733056 + 214 + 21 \\ &= 733291 \end{aligned}$$

e então

$$\begin{aligned} D &= (733291 - 621049) \bmod 7 \\ &= 112242 \bmod 7 \\ &= 4. \end{aligned}$$



# Exercícios

4. Dada uma data fornecida pelo usuário no formato **dd/mm/aaaa**, determine o dia da semana para esta data.

