Registros

Aula 15

Fábio Henrique Viduani Martinez

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul

Algoritmos e Programação I, 2012

Conteúdo da aula

Motivação

- vimos variáveis composta homogêneas (vetores e matrizes)
- existe outra forma para agrupamento de dados, chamada variável composta heterogênea, registro ou estrutura
- em um registro, podemos armazenar sob uma mesma entidade lógica valores de tipos diferentes
- ao invés de índices ou endereços, especificamos o nome de um campo para selecionar um campo particular do registro

Motivação

- vimos variáveis composta homogêneas (vetores e matrizes)
- existe outra forma para agrupamento de dados, chamada variável composta heterogênea, registro ou estrutura
- em um registro, podemos armazenar sob uma mesma entidade lógica valores de tipos diferentes
- ao invés de índices ou endereços, especificamos o nome de um campo para selecionar um campo particular do registro

Motivação

- vimos variáveis composta homogêneas (vetores e matrizes)
- existe outra forma para agrupamento de dados, chamada variável composta heterogênea, registro ou estrutura
- em um registro, podemos armazenar sob uma mesma entidade lógica valores de tipos diferentes
- ao invés de índices ou endereços, especificamos o nome de um campo para selecionar um campo particular do registro

Motivação

- vimos variáveis composta homogêneas (vetores e matrizes)
- existe outra forma para agrupamento de dados, chamada variável composta heterogênea, registro ou estrutura
- em um registro, podemos armazenar sob uma mesma entidade lógica valores de tipos diferentes
- ao invés de índices ou endereços, especificamos o nome de um campo para selecionar um campo particular do registro

Definição e declaração

- 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**, uu simplesmente **campo**
- um registro é composto pelo seu identificador e pelos seus campos

Definição e declaração

- 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**, uu simplesmente **campo**
- um registro é composto pelo seu identificador e pelos seus campos

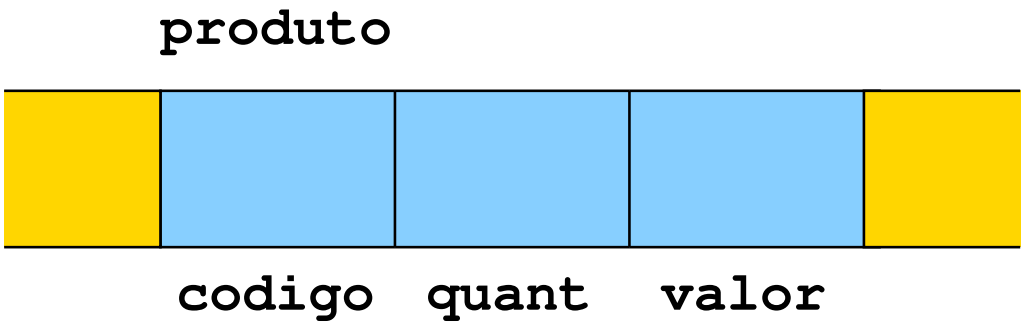
Definição e declaração

- 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**, uu simplesmente **campo**
- um registro é composto pelo seu identificador e pelos seus campos

Definição e declaração

- exemplo:

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



Definição e declaração

- formato geral de declaração de um registro:

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

Definição e declaração

- exemplos:

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

Definição e declaração

- exemplos:

```
struct {
   char sala, turma;
   int horas_inicio, minutos_inicio, horas_fim, minutos_fim;
   float largura, comprimento;
} aula;
```

```
struct {
   char sala;
   char turma;
   int horas_inicio;
   int minutos_inicio;
   int horas_fim;
   int minutos_fim;
   float largura;
   float comprimento;
} aula;
```

Definição e declaração

- a atribuição de um valor a um campo de uma variável do tipo registro é realizada através do acesso a esse campo, especificando o identificador do registro, um ponto e o identificador do campo:

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

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

- quando referenciamos um campo de uma variável do tipo registro, não são permitidos espaços entre o identificador do registro, o operador 'ponto' e o identificador do campo

Definição e declaração

- a atribuição de um valor a um campo de uma variável do tipo registro é realizada através do acesso a esse campo, especificando o identificador do registro, um ponto e o identificador do campo:

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

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

- quando referenciamos um campo de uma variável do tipo registro, não são permitidos espaços entre o identificador do registro, o operador 'ponto' e o identificador do campo

Definição e declaração

- a atribuição de um valor a um campo de uma variável do tipo registro é realizada através do acesso a esse campo, especificando o identificador do registro, um ponto e o identificador do campo:

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

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

- quando referenciamos um campo de uma variável do tipo registro, não são permitidos espaços entre o identificador do registro, o operador ‘ponto’ e o identificador do campo

Definição e declaração

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

Definição e declaração

- 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 = 'O';
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};
```

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

Declaração e inicialização simultâneas

- as regras são idênticas às dos vetores:

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

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

Operações sobre registros

- é incorreto tentar fazer uma atribuição como abaixo:

```
A = B;
```

  para copiar os valores do vetor `B` no vetor `A` (o compilador da linguagem C deve acusar um erro como esse)

- o correto é fazer a cópia elemento a elemento de uma variável para outra; suponha que n é a dimensão desses vetores:

```
for (i = 0; i < n; i++)
  A[i] = B[i];
```

Operações sobre registros

- é incorreto tentar fazer uma atribuição como abaixo:

```
A = B;
```

  para copiar os valores do vetor `B` no vetor `A` (o compilador da linguagem C deve acusar um erro como esse)
- o correto é fazer a cópia elemento a elemento de uma variável para outra; suponha que n é a dimensão desses vetores:

```
for (i = 0; i < n; i++)
   A[i] = B[i];
```

Operações sobre registros

- quando tratamos de 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

```
#include <stdio.h>
int main(void)
{
   struct {
      int hh;
      int mm;
      int ss;
   } agora, prox;
   scanf("%d:%d:%d", &agora.hh, &agora.mm, &agora.ss);
   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 `01/07/2012`.
   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.

Exercícios

```
#include <stdio.h>
int main(void)
{
   struct {
      int dia;
      int mes;
      int ano;
   } data, prox;

   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`**.
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.

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`**.
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.

Exercícios

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:

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

onde

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

e

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

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

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

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