

LPG0002 – Linguagem de Programação

Tipos Estruturados (parte 1)

Prof^a Luciana Rita Guedes
Departamento de Ciência da Computação
UDESC / Joinville

Material elaborado por: Prof. Rui Jorge Tramontin Junior

Introdução

- Uma **estrutura** permite armazenar um conjunto de variáveis dentro de uma única entidade, referenciada pelo seu identificador;

Introdução

- Uma **estrutura** permite armazenar um conjunto de variáveis dentro de uma única entidade, referenciada pelo seu identificador;
- Um exemplo de estrutura pode ser o cadastro de um **produto** e seus respectivos atributos:

Introdução

- Uma **estrutura** permite armazenar um conjunto de variáveis dentro de uma única entidade, referenciada pelo seu identificador;
- Um exemplo de estrutura pode ser o cadastro de um **produto** e seus respectivos atributos:
 - **código** (*inteiro*), **descrição** (*string*) e **preço** (*float*);

Introdução

- Uma **estrutura** permite armazenar um conjunto de variáveis dentro de uma única entidade, referenciada pelo seu identificador;
- Um exemplo de estrutura pode ser o cadastro de um **produto** e seus respectivos atributos:
 - **código** (*inteiro*), **descrição** (*string*) e **preço** (*float*);
- Uma estrutura permite armazenar valores de tipos diferentes → **heterogênea**

Introdução

- Uma estrutura é tratada como um novo tipo da linguagem C;

Introdução

- Uma estrutura é tratada como um novo tipo da linguagem C;
- É possível criar variáveis do tipo **Produto**;

Introdução

- Uma estrutura é tratada como um novo tipo da linguagem C;
- É possível criar variáveis do tipo **Produto**;
- Toda a infraestrutura da linguagem pode ser usada em um tipo estruturado:

Introdução

- Uma estrutura é tratada como um novo tipo da linguagem C;
- É possível criar variáveis do tipo **Produto**;
- Toda a infraestrutura da linguagem pode ser usada em um tipo estruturado:
 - Vetores, ponteiros, *sizeof()*, alocação dinâmica...

Sintaxe da Declaração de uma Estrutura

```
struct identificador {  
    declaração da variável membro 1  
    declaração da variável membro 2  
    ...  
};
```

Exemplo: produto

```
struct Produto {  
    int codigo;  
    char descricao[20];  
    float preco;  
};
```

Exemplo: produto

```
struct Produto {  
    int codigo;  
    char descricao[20];  
    float preco;  
};
```

- As variáveis membros da estrutura são denominadas **campos**;

Exemplo: produto

```
struct Produto {  
    int codigo;  
    char descricao[20];  
    float preco;  
};
```

- As variáveis membros da estrutura são denominadas **campos**;
- Para que a estrutura tenha escopo global (visível no programa todo), a declaração deve ser feita fora da *main()*;

Usando o tipo estruturado

- A declaração de uma variável de tipo estruturado deve ser feita com a palavra reservada ***struct***;

Usando o tipo estruturado

- A declaração de uma variável de tipo estruturado deve ser feita com a palavra reservada ***struct***;

```
struct Produto x;
```

Usando o tipo estruturado

- A declaração de uma variável de tipo estruturado deve ser feita com a palavra reservada ***struct***;

```
struct Produto x;
```

- O acesso aos campos da estrutura é feito pelo operador *ponto* (**.**);

Usando o tipo estruturado

- A declaração de uma variável de tipo estruturado deve ser feita com a palavra reservada ***struct***;

```
struct Produto x;
```

- O acesso aos campos da estrutura é feito pelo operador *ponto* (**.**);

```
x.codigo = 123;
```

Exemplo 1: produto

```
struct Produto {  
    int codigo;  
    char descricao[20];  
    float preco;  
};
```

```
int main() {
```

```
}
```

Exemplo 1: produto

```
struct Produto {  
    int codigo;  
    char descricao[20];  
    float preco;  
};
```

```
int main() {  
    struct Produto x;  
  
}
```

Exemplo 1: produto

```
struct Produto {  
    int codigo;  
    char descricao[20];  
    float preco;  
};
```

```
int main() {  
    struct Produto x;  
    x.codigo = 123;  
  
}
```

Exemplo 1: produto

```
struct Produto {  
    int codigo;  
    char descricao[20];  
    float preco;  
};  
  
int main() {  
    struct Produto x;  
    x.codigo = 123;  
    strcpy(x.descricao, "Caderno");  
  
}
```

Exemplo 1: produto

```
struct Produto {  
    int codigo;  
    char descricao[20];  
    float preco;  
};  
  
int main() {  
    struct Produto x;  
    x.codigo = 123;  
    strcpy(x.descricao, "Caderno");  
    x.preco = 10.0;  
  
}
```

Exemplo 1: produto

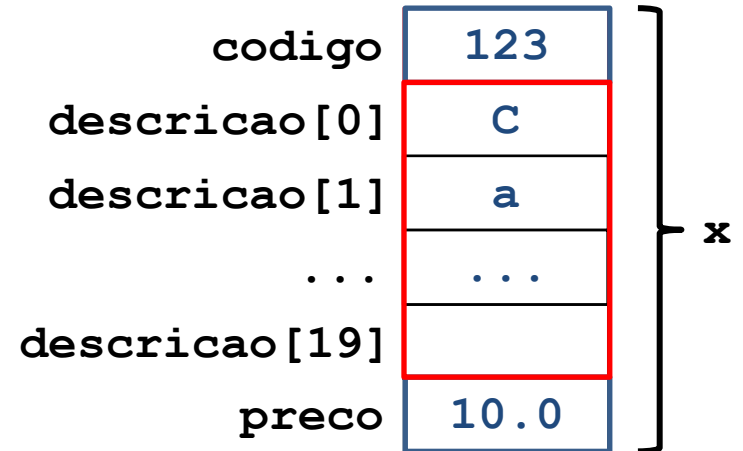
```
struct Produto {  
    int codigo;  
    char descricao[20];  
    float preco;  
};  
  
int main() {  
    struct Produto x;  
    x.codigo = 123;  
    strcpy(x.descricao, "Caderno");  
    x.preco = 10.0;  
    printf("%s (codigo %d) custa R$%.2f\n",  
           x.descricao, x.codigo, x.preco);  
}
```

Exemplo 1: produto

```
struct Produto {  
    int codigo;  
    char descricao[20];  
    float preco;  
};
```

```
int main() {  
    struct Produto x;  
    x.codigo = 123;  
    strcpy(x.descricao, "Caderno");  
    x.preco = 10.0;  
    printf("%s (codigo %d) custa R$%.2f\n",  
           x.descricao, x.codigo, x.preco);  
}
```

Modelo da Memória



Manipulação dos valores

- Cada campo é equivalente a uma variável simples de seu tipo;

Manipulação dos valores

- Cada campo é equivalente a uma variável simples de seu tipo;
- Portanto, uma variável do tipo *int* pode receber o valor do campo `codigo`;

Manipulação dos valores

- Cada campo é equivalente a uma variável simples de seu tipo;
- Portanto, uma variável do tipo *int* pode receber o valor do campo `codigo`;

```
int n = x.codigo;
```

Manipulação dos valores

- Cada campo é equivalente a uma variável simples de seu tipo;
- Portanto, uma variável do tipo *int* pode receber o valor do campo `codigo`;

```
int n = x.codigo;
```

- A entrada de dados também deve ser feita para cada campo;

Manipulação dos valores

- Cada campo é equivalente a uma variável simples de seu tipo;
- Portanto, uma variável do tipo *int* pode receber o valor do campo `codigo`;

```
int n = x.codigo;
```

- A entrada de dados também deve ser feita para cada campo;

```
scanf ("%d", &x.codigo) ;
```

Atribuição entre estruturas

- A linguagem permite a atribuição direta entre variáveis de tipo estruturado;

Atribuição entre estruturas

- A linguagem permite a atribuição direta entre variáveis de tipo estruturado;
- Continuando o exemplo anterior:

```
struct Produto y;
```

Atribuição entre estruturas

- A linguagem permite a atribuição direta entre variáveis de tipo estruturado;
- Continuando o exemplo anterior:

```
struct Produto y;  
y = x;
```


Atribuição entre estruturas

- A linguagem permite a atribuição direta entre variáveis de tipo estruturado;
- Continuando o exemplo anterior:

```
struct Produto y;  
y = x;
```

```
printf("%s (codigo %d) custa R$%.2f\n",  
       y.descricao, y.codigo, y.preco);
```

Vetor de estruturas

- A declaração é semelhante a de um vetor de tipo simples:

Vetor de estruturas

- A declaração é semelhante a de um vetor de tipo simples:

```
struct Produto v[10];
```

Vetor de estruturas

- A declaração é semelhante a de um vetor de tipo simples:

```
struct Produto v[10];
```

- O acesso é feito da mesma forma, combinando-se os colchetes (para acessar uma *posição*) e o ponto (para acessar um *campo*);

Vetor de estruturas

- A declaração é semelhante a de um vetor de tipo simples:

```
struct Produto v[10];
```

- O acesso é feito da mesma forma, combinando-se os colchetes (para acessar uma *posição*) e o ponto (para acessar um *campo*);
- Ex.: acesso ao campo *codigo* do produto na posição *i*:

Vetor de estruturas

- A declaração é semelhante a de um vetor de tipo simples:

```
struct Produto v[10];
```

- O acesso é feito da mesma forma, combinando-se os colchetes (para acessar uma *posição*) e o ponto (para acessar um *campo*);
- Ex.: acesso ao campo *codigo* do produto na posição *i*:

```
v[i].codigo = 123;
```

Exemplo 2: vetor de produtos

```
struct Produto {  
    int codigo;  
    char descricao[20];  
    float preco;  
};  
  
int main() {  
    struct Produto v[10];  
    int i;  
  
    // Continua...
```

Exemplo 2: vetor de produtos

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

```
}
```

```
return 0;
```

```
}
```


Exemplo 2: vetor de produtos

```
for( i = 0 ; i < 10 ; i++ ){  
    printf("Produto %d:\n", i + 1);  
  
}
```

```
return 0;  
}
```

Exemplo 2: vetor de produtos

```
for( i = 0 ; i < 10 ; i++ ){  
    printf("Produto %d:\n", i + 1);  
    scanf("%d", &v[i].codigo);  
    scanf("%s", v[i].descricao);  
    scanf("%f", &v[i].preco);  
}
```

```
return 0;  
}
```

Exemplo 2: vetor de produtos

```
for( i = 0 ; i < 10 ; i++ ){  
    printf("Produto %d:\n", i + 1);  
    scanf("%d", &v[i].codigo);  
    scanf("%s", v[i].descricao);  
    scanf("%f", &v[i].preco);  
}  
for( i = 0 ; i < 10 ; i++ ){
```

```
}  
return 0;  
}
```

Exemplo 2: vetor de produtos

```
for( i = 0 ; i < 10 ; i++ ){
    printf("Produto %d:\n", i + 1);
    scanf("%d", &v[i].codigo);
    scanf("%s", v[i].descricao);
    scanf("%f", &v[i].preco);
}
for( i = 0 ; i < 10 ; i++ ){
    printf("Dados do Produto %d:\n", i + 1);

}
return 0;
}
```

Exemplo 2: vetor de produtos

```
for( i = 0 ; i < 10 ; i++ ){
    printf("Produto %d:\n", i + 1);
    scanf("%d", &v[i].codigo);
    scanf("%s", v[i].descricao);
    scanf("%f", &v[i].preco);
}
for( i = 0 ; i < 10 ; i++ ){
    printf("Dados do Produto %d:\n", i + 1);
    printf("Código: %d\n", v[i].codigo);
    printf("Descrição: %s\n", v[i].descricao);
    printf("Preço: R$%.2f\n\n", v[i].preco);
}
return 0;
}
```

Exemplo prático

- Vetor de produtos alocado dinamicamente.

Exercício

- Modifique o exemplo apresentado da seguinte forma:
 - Faça com que o usuário entre com um **valor**;
 - Mostre na tela as **descrições** e os **preços** dos *produtos* cujo preço é **menor** que o **valor** informado pelo usuário.