Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Linguagem C

Estrutura de dados (*structs*)

André Tavares da Silva

andre.silva@udesc.br

Estrutura de Dados

- As valores armazenados em vetores ou matrizes são todos do mesmo tipo, e por isso são chamados de variáveis compostas homogêneas.
- Porém, há casos em que precisamos agrupar variáveis de **diferentes tipos**. Para estes casos a linguagem C nos fornece as estruturas (*structs*).

Estrutura de Dados

- A vantagem em se usar estruturas de dados é que podemos agrupar de forma organizada vários tipos de dados diferentes.
- Por exemplo, dentro de uma estrutura de dados usada para armazenar dados de alunos podemos ter nome, número de matrícula, data de nascimento, notas, etc.

Sintaxe e exemplo

```
struct <identificador>
{
   <tipo> <identificador do campo 1>;
   <tipo> <identificador do campo 2>;
   . . .
   <tipo> <identificador do campo n>;
} <variaveis>;
```

```
struct Aluno
{
   int matricula;
   char nome[50];
   float coeficiente_rendimento;
} aluno1, aluno2;
```

Tipos homogêneos?

- Nem sempre os tipos de uma estrutura precisam ser diferentes. Em alguns casos, podemos usar elas para definir dados homogêneos, como um ponto no espaço.

```
struct Ponto3D
{
   float x, y, z;
} p1, p2;
```

Tipos de dados - typedef

- Ao se definir uma estrutura de dados (*struct*), está se definindo um novo tipo de dado. A linguagem C oferece uma outra forma de se definir um novo tipo de dado. Trata-se da declaração *typedef* que pode ser utilizado para se definir a estrutura Aluno da seguinte forma:

```
typedef struct
{
   int matricula;
   char nome[50];
   float coeficiente_rendimento;
} Aluno;
```

Declaração de variáveis

- Após criarmos uma estrutura de dados com *struct*, poderemos utilizá-la como um tipo de dado comum, ou seja, da mesma forma que declarar *float*, *int*, ou *char*.
- Para declarar variáveis do tipo Pessoa você tem duas opções.
  - A primeira consiste em declarar as variáveis juntamente com a declaração da estrutura.
  - A forma mais prática é criar um tipo com *typedef* e declarar a variável no local (função) onde ela será utilizada.

Declaração

- Juntamente com a declaração da estrutura:

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

Declaração

- No local onde ela é utilizada:

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

int main()
{
   Data dt_nascimento;
   int i;
. . .
```

Acesso aos membros da estrutura

- Para acessar os membros de uma estrutura de dados utilizamos o nome da variável declarada mais um ponto (.) e o nome do membro:

```
int main()
{
   Data hoje;
   int i;
. . .
   hoje.dia = 22;
   hoje.mes = 4;
   hoje.ano = 2013;
. . .
```

Acesso aos membros da estrutura

- Para acessar os membros de uma estrutura de dados através de um ponteiro, utilizamos o nome da variável declarada mais os sinais de menos e maior que (->) seguidos do nome do membro:

```
int main()
{
   Data hoje;
   Data *ptr_dia = &hoje;
. . .
   ptr_dia->dia = 22;
   ptr_dia->mes = 4;
   ptr_dia->ano = 2013;
. . .
```

Estruturas e vetores

- As estruturas normalmente são também chamadas de registros, armazenando dados como em arquivos como será visto mais adiante no curso.
- Algumas vezes precisamos utilizar uma grande quantidade de informações de um tipo heterogêneo, como por exemplo todos os alunos de uma turma.

```
typedef struct
{
   int matricula;
   char nome[50];
   float coeficiente_rendimento;
} Aluno;
```

```
int main()
{
   Aluno turma[50];
   int i;

   for(i=0; i<50; i++) {
. . .
```

Estruturas e vetores

- As estruturas normalmente são também chamadas de registros, armazenando dados como em arquivos como será visto mais adiante no curso.
- Algumas vezes precisamos utilizar uma grande quantidade de informações de um tipo heterogêneo, como por exemplo todos os alunos de uma turma.

Outra forma de declaração (moderna)

```
struct Aluno
{
   int matricula;
   char nome[50];
   float coeficiente_rendimento;
};
```

```
int main()
{
   Aluno turma[50];
   int i;

   for(i=0; i<50; i++) {
. . .
```

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()*;

Exemplo: produto

- 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;


}
```

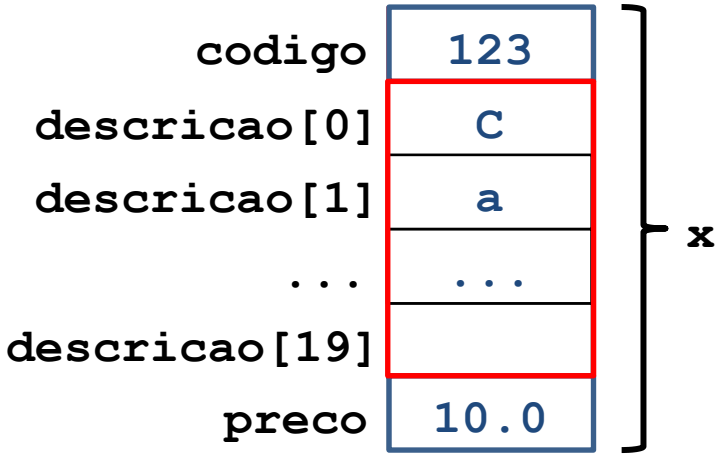
Exemplo 1: produto

```
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;
}
```

Exercício

- Crie uma estrutura de dados para uma pesquisa, contendo dados coletados como: idade, salário, número de filhos e sexo.
- Faça um programa para ler os dados de uma pesquisa com 20 pessoas e fornecer a média salarial, a média das idades e o número de mulheres cujo salário é maior que R$ 500,00.