Departamento de Ciência da COMPUTAÇÃO

UDESC Joinville

Linguagem C

Estrutura de dados (*structs*) - continuação

André Tavares da Silva

andre.silva@udesc.br

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

```
int main()
{
    Produto prods[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.

Sem definir tipo

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

```
int main()
{
    struct Produto prods[50];
    int i;

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

Exemplo: vetor de produtos

```
// Declaração da struct e das funções

int main(){
   Produto v[10];
   int i;









}
```

Exemplo: vetor de produtos

```
// Declaração da struct e das funções

int main(){
   Produto v[10];
   int i;
   for( i = 0 ; i < 10 ; i++ ){
     printf("Produto %d:\n", i + 1);

   }




}
```

Exemplo: vetor de produtos

```
// Declaração da struct e das funções

int main(){
   Produto v[10];
   int i;
   for( i = 0 ; i < 10 ; i++ ){
     printf("Produto %d:\n", i + 1);
     le_produto( &v[i] ); // ou v + i
   }




}
```

Exemplo: vetor de produtos

```
// Declaração da struct e das funções

int main(){
   Produto v[10];
   int i;
   for( i = 0 ; i < 10 ; i++ ){
     printf("Produto %d:\n", i + 1);
     le_produto( &v[i] ); // ou v + i
   }
   for( i = 0 ; i < 10 ; i++ ){
     printf("Dados do Produto %d:\n", i + 1);

   }
}
```

Exemplo: vetor de produtos

```
// Declaração da struct e das funções

int main(){
   Produto v[10];
   int i;
   for( i = 0 ; i < 10 ; i++ ){
      printf("Produto %d:\n", i + 1);
      le_produto( &v[i] ); // ou v + i
   }
   for( i = 0 ; i < 10 ; i++ ){
      printf("Dados do Produto %d:\n", i + 1);
      mostra_produto( v[i] ); // ou *(v + i)
   }
}
```

Exercício

- Modifique o exemplo anterior da seguinte forma:
  - Faça o vetor com alocação dinâmica (tal como mostrado em aulas passadas);
  - Implemente uma função de busca que recebe o vetor de produtos, a capacidade do vetor e o código do produto (*chave*);
    - Retorna o *índice* do produto no vetor (ou -1, caso não seja encontrado);

```
int busca(Produto *v, int n, int codigo);
```

Estruturas aninhadas

- **Estruturas** podem possuir campos que sejam de tipos estruturados;

Estruturas aninhadas

- **Estruturas** podem possuir campos que sejam de tipos estruturados;

- Por exemplo, para armazenar os dados de uma **pessoa** em uma agenda telefônica, podemos ter os seguintes campos;
  - Nome;
  - Telefone;
  - Data de nascimento (dia, mês, ano);

Estruturas aninhadas

- **Estruturas** podem possuir campos que sejam de tipos estruturados;

- Por exemplo, para armazenar os dados de uma **pessoa** em uma agenda telefônica, podemos ter os seguintes campos;
  - Nome;
  - Telefone;
  - Data de nascimento (dia, mês, ano);

- A **data de nascimento** também é uma estrutura, contendo *dia*, *mês* e *ano*.

Exemplo 2: dados de uma pessoa

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

Exemplo 2: dados de uma pessoa

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

typedef struct pessoa {
    char nome[30];
    char telefone[20];
    Data nascimento;
};
```

Exemplo 2: dados de uma pessoa

```
void le_pessoa( Pessoa *p );
void mostra_pessoa( Pessoa x );

int main() {









    return 0;
}
```

Exemplo 2: dados de uma pessoa

```
void le_pessoa( Pessoa *p );
void mostra_pessoa( Pessoa x );

int main() {
    Pessoa fulano;




    return 0;
}
```

Exemplo 2: dados de uma pessoa

```
void le_pessoa( Pessoa *p );
void mostra_pessoa( Pessoa x );

int main() {
    Pessoa fulano;

    le_pessoa( &fulano );



    return 0;
}
```

Exemplo 2: dados de uma pessoa

```
void le_pessoa( Pessoa *p );
void mostra_pessoa( Pessoa x );

int main() {
    Pessoa fulano;

    le_pessoa( &fulano );

    mostra_pessoa( fulano );

    return 0;
}
```

Exemplo 2: dados de uma pessoa

```
void le_pessoa( Pessoa *p ){




}
```

Exemplo 2: dados de uma pessoa

```
void le_pessoa( Pessoa *p ){
    printf("Digite o nome: ");
    gets(p->nome);









}
```

Exemplo 2: dados de uma pessoa

```
void le_pessoa( Pessoa *p ){
   printf("Digite o nome: ");
   gets(p->nome);
   printf("Digite o numero de telefone: ");
   gets(p->telefone);









}
```

Exemplo 2: dados de uma pessoa

```
void le_pessoa( Pessoa *p ){
   printf("Digite o nome: ");
   gets(p->nome);
   printf("Digite o numero de telefone: ");
   gets(p->telefone);
   printf("Digite a data de nascimento:\n");
   printf("Digite o dia: ");
   scanf("%d", &p->nascimento.dia );




}
```

Exemplo 2: dados de uma pessoa

```
void le_pessoa( Pessoa *p ){
   printf("Digite o nome: ");
   gets(p->nome);
   printf("Digite o numero de telefone: ");
   gets(p->telefone);
   printf("Digite a data de nascimento:\n");
   printf("Digite o dia: ");
   scanf("%d", &p->nascimento.dia );
   printf("Digite o mes: ");
   scanf("%d", &p->nascimento.mes );


}
```

Exemplo 2: dados de uma pessoa

```
void le_pessoa( Pessoa *p ){
   printf("Digite o nome: ");
   gets(p->nome);
   printf("Digite o numero de telefone: ");
   gets(p->telefone);
   printf("Digite a data de nascimento:\n");
   printf("Digite o dia: ");
   scanf("%d", &p->nascimento.dia );
   printf("Digite o mes: ");
   scanf("%d", &p->nascimento.mes );
   printf("Digite o ano: ");
   scanf("%d", &p->nascimento.ano );
}
```

Exemplo 2: dados de uma pessoa

```
void mostra_pessoa( Pessoa x ){




}
```

Exemplo 2: dados de uma pessoa

```
void mostra_pessoa( Pessoa x ){
   printf("Nome: %s\n", x.nome );




}
```

Exemplo 2: dados de uma pessoa

```
void mostra_pessoa( Pessoa x ){
   printf("Nome: %s\n", x.nome );

   printf("Telefone: %s\n", x.telefone );




}
```

Exemplo 2: dados de uma pessoa

```
void mostra_pessoa( Pessoa x ){
   printf("Nome: %s\n", x.nome );

   printf("Telefone: %s\n", x.telefone );

   printf("Data de nascimento: %2d/%2d/%4d\n",
                           x.nascimento.dia,
                           x.nascimento.mes,
                           x.nascimento.ano );
}
```

Considerações

- A manipulação dos campos do tipo `Data` foi feita dentro das funções de manipulação da `Pessoa`;
- Em princípio, não há nada de errado nessa abordagem;
- Porém, caso o tipo data seja usado como campo em outras estruturas, o uso de funções específicas seria uma abordagem mais otimizada.

Considerações

- A ideia aqui é tratar `Data` de maneira independente de outras estruturas;
- Vamos então criar funções de manipulação específicas para manipulação da `Data`;
  - `le_data()` ;
  - `mostra_data()`;
- Tais funções podem ser reutilizadas por outras estruturas, caso tenham campos do tipo `Data`;

Exemplo 3: funções para o tipo data

```
void le_data( Data *p ){
   printf("Digite o dia: ");
   scanf("%d", &p->dia );
   printf("Digite o mes: ");
   scanf("%d", &p->mes );
   printf("Digite o ano: ");
   scanf("%d", &p->ano );
}
```

Exemplo 3: funções para o tipo data

```
void le_data( Data *p ){
   printf("Digite o dia: ");
   scanf("%d", &p->dia );
   printf("Digite o mes: ");
   scanf("%d", &p->mes );
   printf("Digite o ano: ");
   scanf("%d", &p->ano );
}

void mostra_data( Data x ){
   printf("%2d/%2d/%4d\n", x.dia, x.mes, x.ano);
}
```

Exemplo 3: funções para o tipo data

```
void le_pessoa_v2( Pessoa *p ){
    printf("Digite o nome: ");
    gets(p->nome);
    printf("Digite o numero de telefone: ");
    gets(p->telefone);
    printf("Digite a data de nascimento:\n");
    printf("Digite o dia: ");
    scanf("%d", &p->nascimento.dia );
    printf("Digite o mes: ");
    scanf("%d", &p->nascimento.mes );
    printf("Digite o ano: ");
    scanf("%d", &p->nascimento.ano );
}
```

Exemplo 3: funções para o tipo data

```
void le_pessoa_v2( Pessoa *p ){
    printf("Digite o nome: ");
    gets(p->nome);
    printf("Digite o numero de telefone: ");
    gets(p->telefone);
    printf("Digite a data de nascimento:\n");
    printf("Digite o dia: ");
    scanf("%d", &p->nascimento.dia );
    printf("Digite o mes: ");
    scanf("%d", &p->nascimento.mes );
    printf("Digite o ano: ");
    scanf("%d", &p->nascimento.ano );
}
```

Exemplo 3: funções para o tipo data

```
void le_pessoa_v2( Pessoa *p ){
    printf("Digite o nome: ");
    gets(p->nome);
    printf("Digite o numero de telefone: ");
    gets(p->telefone);
    printf("Digite a data de nascimento:\n");
    le_data( &p->nascimento );
}
```

Exemplo 3: funções para o tipo data

```
void mostra_pessoa_v2( Pessoa x ){
   printf("Nome: %s\n", x.nome );

   printf("Telefone: %s\n", x.telefone );

   printf("Data de nascimento: %2d/%2d/%4d\n",
                           x.nascimento.dia,
                           x.nascimento.mes,
                           x.nascimento.ano );
}
```

Exemplo 3: funções para o tipo data

```
void mostra_pessoa_v2( Pessoa x ){
   printf("Nome: %s\n", x.nome );

   printf("Telefone: %s\n", x.telefone );

   printf("Data de nascimento: %2d/%2d/%4d\n",
                              x.nascimento.dia,
                              x.nascimento.mes,
                              x.nascimento.ano );
}
```

Exemplo 3: funções para o tipo data

```
void mostra_pessoa_v2( Pessoa x ){
    printf("Nome: %s\n", x.nome );

    printf("Telefone: %s\n", x.telefone );

    printf("Data de nascimento:");
    mostra_data( x.nascimento );
}
```

Considerações

- Embora o exemplo apresentado seja muito simples, a reutilização de estruturas e suas funções associadas é fundamental para a construção de sistemas maiores;

- Suponha que tenhamos que criar outras entidades para o nosso sistema que utilizam campos do **tipo data**, tais como:
  - agenda de eventos;
  - pedidos de compras;

- Essas entidades poderiam reutilizar as funções já criadas, organizando melhor o código.

Exercício

- Modifique o exemplo 3 da seguinte forma:
  - Faça o vetor de pessoas com alocação dinâmica;
  - Ordene o vetor baseado no nome da pessoa;
  - Dada uma letra, mostre na tela os dados das pessoas cujo nome começem com essa letra;
  - Dado um ano, mostre o dados das pessoas que nasceram a partir daquele ano.

Revisitando Exercício (pg 9)

- Modifique o **primeiro exemplo** da seguinte forma:
  - Faça o vetor com alocação dinâmica (tal como mostrado em aulas passadas);
  - Implemente uma função de busca que recebe o vetor de produtos, a capacidade do vetor e o código do produto (*chave*);
    - Retorna o *índice* do produto no vetor (ou -1, caso não seja encontrado);

```
int busca(Produto *v, int n, int codigo);
```