# LPG0002 – Linguagem de Programação

## Tipos Estruturados
## (parte 3)

Profª Luciana Rita Guedes
Departamento de Ciência da Computação
UDESC / Joinville

Material elaborado por: Prof. Rui Jorge Tramontin Junior

# Introdução

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

# Introdução

- **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;

# Introdução

- **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;

# Introdução

- **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;

# Introdução

- **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);

# Introdução

- **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 1: dados de uma pessoa

```
struct data{
    int dia;
    int mes;
    int ano;
};
```

# Exemplo 1: dados de uma pessoa

```
struct data{
    int dia;
    int mes;
    int ano;
};

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

# Exemplo 1: dados de uma pessoa

```
void le_pessoa( struct pessoa *p );
void mostra_pessoa( struct pessoa x );

int main() {




    return 0;
}
```

# Exemplo 1: dados de uma pessoa

```
void le_pessoa( struct pessoa *p );
void mostra_pessoa( struct pessoa x );

int main() {
    struct pessoa fulano;




    return 0;
}
```

# Exemplo 1: dados de uma pessoa

```
void le_pessoa( struct pessoa *p );
void mostra_pessoa( struct pessoa x );

int main() {
    struct pessoa fulano;

    le_pessoa( &fulano );



    return 0;
}
```

# Exemplo 1: dados de uma pessoa

```
void le_pessoa( struct pessoa *p );
void mostra_pessoa( struct pessoa x );

int main() {
    struct pessoa fulano;

    le_pessoa( &fulano );

    mostra_pessoa( fulano );

    return 0;
}
```

# Exemplo 1: leitura dos dados

```
void le_pessoa( struct pessoa *p ){




}
```

# Exemplo 1: leitura dos dados

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









}
```

# Exemplo 1: leitura dos dados

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









}
```

# Exemplo 1: leitura dos dados

```
void le_pessoa( struct 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 1: leitura dos dados

```
void le_pessoa( struct 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 1: leitura dos dados

```
void le_pessoa( struct 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 1: impressão dos dados

```
void mostra_pessoa( struct pessoa x ){




}
```

# Exemplo 1: impressão dos dados

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




}
```

# Exemplo 1: impressão dos dados

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

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




}
```

# Exemplo 1: impressão dos dados

```
void mostra_pessoa( struct 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 **`struct data`** foi feita dentro das funções de manipulação da **`struct pessoa`**;

# Considerações

- A manipulação dos campos do tipo **`struct data`** foi feita dentro das funções de manipulação da **`struct pessoa`**;

- Em princípio, não há nada de errado nessa abordagem;

# Considerações

- A manipulação dos campos do tipo **`struct data`** foi feita dentro das funções de manipulação da **`struct 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 **`struct data`** de maneira independente de outras estruturas;

# Considerações

- A ideia aqui é tratar **`struct data`** de maneira independente de outras estruturas;

- Vamos então criar funções de manipulação específicas para manipulação da **`struct data`**;

# Considerações

- A ideia aqui é tratar **`struct data`** de maneira independente de outras estruturas;

- Vamos então criar funções de manipulação específicas para manipulação da **`struct data`**;
  - **`le_data()`**;
  - **`mostra_data()`**;

# Considerações

- A ideia aqui é tratar **`struct data`** de maneira independente de outras estruturas;

- Vamos então criar funções de manipulação específicas para manipulação da **`struct data`**;
  - **`le_data()`** ;
  - **`mostra_data()`**;

- Tais funções podem ser reutilizadas por outras estruturas, caso tenham campos do tipo **`struct data`**;

# Exemplo 2: funções para o tipo data

```
void le_data( struct 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 2: funções para o tipo data

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

# Exemplo 3: leitura dos dados da pessoa (versão 2)

```
void le_pessoa_v2( struct 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: leitura dos dados da pessoa (versão 2)

```
void le_pessoa_v2( struct 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: leitura dos dados da pessoa (versão 2)

```
void le_pessoa_v2( struct 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: impressão dos dados da pessoa (versão 2)

```
void mostra_pessoa_v2( struct 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: impressão dos dados da pessoa (versão 2)

```
void mostra_pessoa_v2( struct 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: impressão dos dados da pessoa (versão 2)

```
void mostra_pessoa_v2( struct 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;

# 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:

# 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;

# 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;

# 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.