

# Estruturas de dados

Estruturas (structs)

# Struct

- É uma coleção de uma ou mais variáveis (de **mesmo tipo** ou de **tipos diferentes**) agrupadas sobre um único nome.
- Cada elemento dessa coleção é chamado de **campo da estrutura**.

## Definição da estrutura

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

OBS: Funciona como se fosse um novo tipo de dado definido pelo<sub>2</sub> programador

# Struct

## Forma geral da definição de uma estrutura

```
struct nome_da_estrutura {  
    tipo nome_do_elemento_1;  
    tipo nome_do_elemento_2;  
    tipo nome_do_elemento_3;  
    ...  
} variáveis_da_estrutura;
```

Onde `nome_da_estrutura` e as `variáveis_da_estrutura` podem ser omitidos, mas não simultaneamente.

# Struct

Como declarar uma variável do tipo struct?

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

Definição da estrutura

```
struct Data dtaNasc;
```

Declaração da variável

dtaNasc

dia	mes	ano
7	9	1822

# Struct

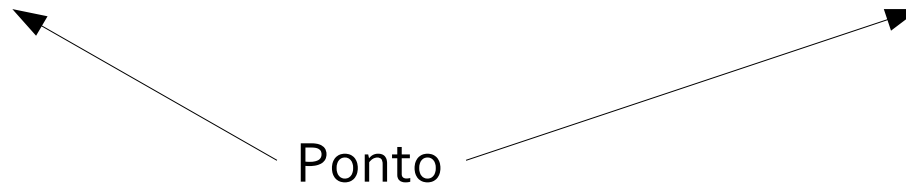
Como acessar um campo da struct?

**dtaNasc**

dia	mes	ano
7	9	1822

**dtaNasc.dia**

**dtaNasc.mes**



# Struct

## PROBLEMA:

Definir uma estrutura chamada Data com os campos dia, mes e ano. Escrever um programa para ler uma data e imprimir se nela comemora-se ou não o dia da independência. Imprimir a data lida no formato `dd/mm/aaaa`

# Struct

```
#include <stdio.h>

struct Data {
    int dia,
        mes,
        ano;
};

int main()
{
    struct Data dt;

    printf("Informe o dia:");
    scanf("%d", &dt.dia);

    printf("Informe o mês:");
    scanf("%d", &dt.mes);

    printf("Informe o ano:");
    scanf("%d", &dt.ano);
```

```
    if (dt.dia == 7 && dt.mes==9) {
        printf("Dia da independência\n");
    }
    else
        printf("Sem comemoração\n");

    printf("Data digitada: %d/%d/%d\n",
        dt.dia, dt.mes, dt.ano);

    return 0;
}
```

# Estruturas aninhadas

Uma estrutura dentro da outra.

codigo	salario	dtaNasc		
		dia	mes	ano
2354	1000.00	15	9	1995

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

```
struct Funcionario {  
    int codigo;  
    float salario;  
    struct Data dtaNasc;  
};
```



# Estruturas aninhadas

```
struct Data {  
    int dia,mes,ano;  
};  
  
struct Funcionario {  
    int codigo;  
    float salario;  
    struct Data dtaNasc;  
};  
  
int main()  
{  
    struct Funcionario func;  
  
    scanf ("%d", &func.codigo) ;  
    scanf ("%f", &func.salario) ;  
    scanf ("%d", &func.dtaNasc.dia) ;  
    scanf ("%d", &func.dtaNasc.mes) ;  
    scanf ("%d", &func.dtaNasc.ano) ;  
    . . .
```

# Vetor dentro de uma estrutura

matricula	rg	notas		
		[0]	[1]	[2]
2936	1234	8.0	9.0	8.5

```
struct Aluno {  
    int matricula;  
    int rg;  
    float notas[3];  
};
```

# Vetor dentro de uma estrutura

```
struct Aluno {  
    int matricula;  
    int rg;  
    float notas[3];  
};  
  
int main()  
{  
    int i;  
    struct Aluno alu;  
  
    scanf("%d", &alu.matricula);  
    scanf("%d", &alu.rg);  
    for (i=0; i<=2; i++)  
        scanf("%f", &alu.notas[i]);  
    ...  
}
```

# Vetor de estruturas

	matricula	rg	dtaNasc			notas		
			dia	mes	ano	[0]	[1]	[2]
0	3011	1345	15	6	1996	9.0	10.0	9.5
1	4123	9342	17	8	1997	7.0	8.0	7.5
...								
4	2936	2766	22	5	1995	6.0	8.0	7.0

```
struct Aluno {  
    int matricula;  
    int rg;  
    Data dtaNasc;  
    float notas[3];  
};
```

# Vetor de estruturas

```
...  
int main()  
{  
    int i,j;  
    struct Aluno vet[5];  
  
    for (i=0; i<=4; i++) {  
        scanf("%d",&vet[i].matricula);  
        scanf("%d",&vet[i].rg);  
        scanf("%d",&vet[i].dtaNasc.dia);  
        scanf("%d",&vet[i].dtaNasc.mes);  
        scanf("%d",&vet[i].dtaNasc.ano);  
        for (j=0; j<=2; j++)  
            scanf("%f",&vet[i].notas[j]);  
    }  
}
```

...

# Atribuição de estruturas

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

```
int main()  
{  
    struct Data dta,dtb;
```

```
    scanf ("%d", &dta.dia) ;  
    scanf ("%d", &dta.mes) ;  
    scanf ("%d", &dta.ano) ;
```

```
    dtb = dta;
```

```
    printf ("%d/%d/%d\n", dtb.dia, dtb.mes, dtb.ano) ;  
    return 0;  
}
```

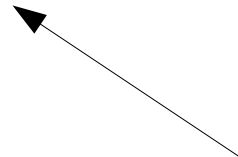
# typedef

Usado para definir um novo tipo.

Exemplo:

```
typedef int INTEIRO;
```

```
int main()  
{  
    INTEIRO i;  
  
    ...  
}
```



Novo tipo

# typedef

Pode ser usado com structs

Exemplo:

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

```
typedef struct data Data;
```

```
int main()  
{  
    Data dtaNasc;
```

```
...
```



Novo tipo



# typedef

Exemplos:

Estrutura sem nome

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

```
int main()  
{  
    Data dtaNasc;
```

```
...
```

Novo tipo

# Structs e funções

Na linguagem C é possível:

- Passar estruturas como argumento de uma função
- Retornar uma estrutura

```
void escreveData (Data dta) ;  
Data leData(void) ;
```

# Structs e funções

...

```
Data leData() {  
    Data d;  
  
    scanf("%d %d %d", &d.dia, &d.mes, &d.ano);  
    return d;  
}  
  
void escreveData(Data dta) {  
    printf("%d/%d/%d\n",  
           dta.dia, dta.mes, dta.ano);  
}  
  
int main() {  
    Data dt;  
  
    dt = leData();  
    escreveData(dt);  
    return 0;  
}
```

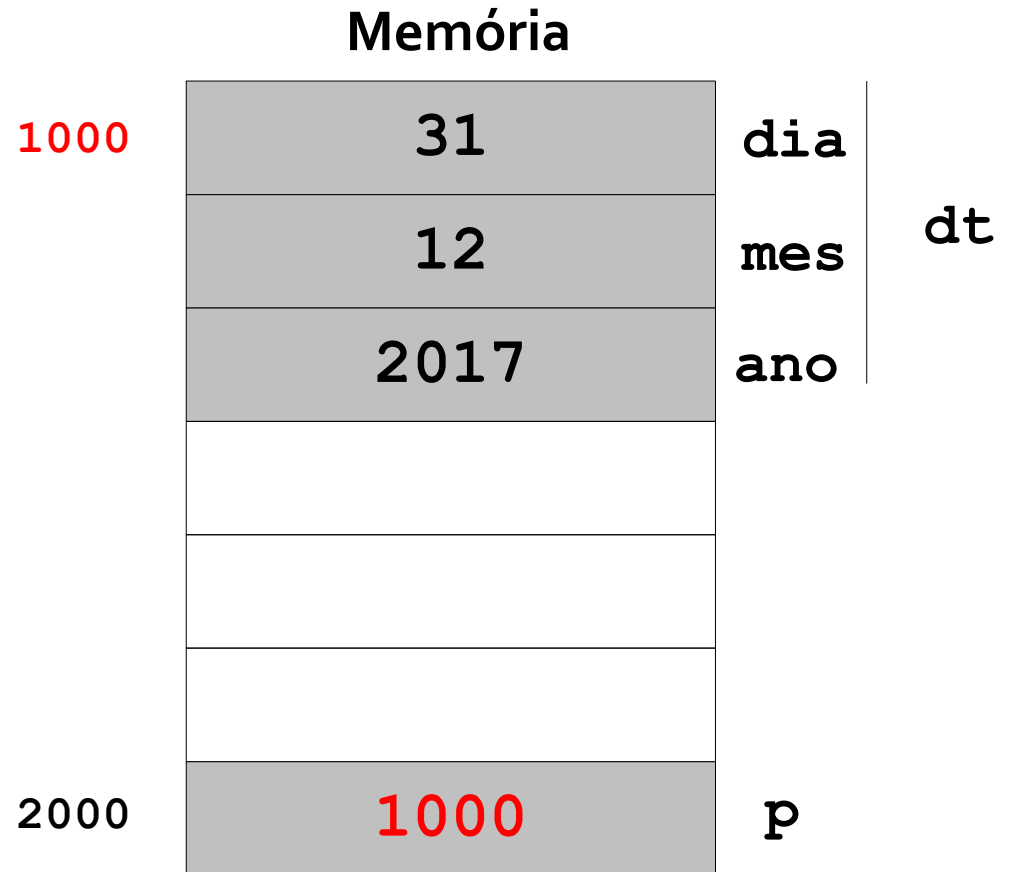
# Ponteiros e structs

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

```
int main() {  
    Data dt, *p;  
  
    dt.dia = 31;  
    dt.mes = 12;  
    dt.ano = 2017;
```

```
p = &dt;
```

```
printf("%d/%d/%d\n", p->dia, p->mes, p->ano);  
return 0;  
}
```



Referencia um campo da estrutura a partir de um ponteiro

# Passando estruturas por referência

Função: **leData**

Saída: uma data

Descrição: Faz a leitura de uma data

```
...  
void leData(Data *pdt) {  
    scanf("%d", &pdt->dia);  
    scanf("%d", &pdt->mes);  
    scanf("%d", &pdt->ano);  
}  
  
int main() {  
    Data dt;  
  
    leData(&dt);  
    printf("%d/%d/%d\n", dt.dia, dt.mes, dt.ano);  
    return 0;  
}
```