



PROGRAMAÇÃO ESTRUTURADA

Estruturas

ESTRUTURAS - STRUCT

- Uma estrutura é uma coleção de uma ou mais variáveis colocadas juntas em um único nome
- As variáveis da estrutura podem possuir tipos diferentes
- Permitem agrupar dados para uma manipulação mais organizada e conveniente

ESTRUTURAS - STRUCT

- Exemplos:

- Aluno

- Número de matrícula
 - CR
 - Nome

- Ponto

- x
 - y

ESTRUTURAS - STRUCT

○ Declaração:

```
struct nome_da_estrutura {  
    tipo comp1;  
    tipo comp2;  
    ...  
};
```

```
struct tipoPonto {  
    int x;  
    int y;  
};
```

```
struct tipoAluno {  
    int numMat;  
    float CR;  
    char nome[40];  
};
```

ESTRUTURAS - STRUCT

- Uma declaração `struct` pode ser seguida por uma lista de variáveis

```
struct tipoPonto {  
    int x;  
    int y;  
} p1, p2, p3;
```

- Se a declaração não é seguida por uma lista de variáveis, não é reservado espaço de memória. Apenas é descrito o formato da estrutura.

ESTRUTURAS - STRUCT

- O programador também pode criar um tipo da sua definição de estrutura.
- Exemplo:

```
struct tipoAluno {  
    int numMat;  
    float CR;  
    char nome[40];  
};  
typedef struct tipoAluno tAluno;  
...  
tAluno aluno1, aluno2;
```

ESTRUTURAS - STRUCT

- Tendo criado as variáveis `aluno1` e `aluno2` do tipo `taluno`

`aluno1.numMat`

`aluno1.CR`

`aluno1.nome`

`aluno2.numMat`

`aluno2.CR`

`aluno2.nome`

- As únicas operações legais em um estrutura são copiá-las e atribuí-las como uma unidade
- Cópia e atribuição incluem a passagem de argumentos para funções e também o retorno
- As estruturas não podem ser comparadas como uma unidade
(`aluno1 == aluno2` ✖)

FUNÇÃO COM ESTRUTURAS

```
struct tipoPonto {  
    int x;  
    int y;  
};  
typedef struct tipoPonto tPonto;
```

```
int main (void) {  
    tPonto A,B,C;  
    A.x = 1;  
    A.y = 2;  
    B.x = 3;  
    B.y = 4;  
    C = somaPonto(A,B);  
    printf("%d %d", C.x, C.y);  
    return 0;  
}
```

```
tPonto somaPonto(tPonto M, tPonto N) {  
    tPonto K;  
    K.x = M.x + N.x;  
    K.y = M.y + N.y;  
    return K;  
}
```

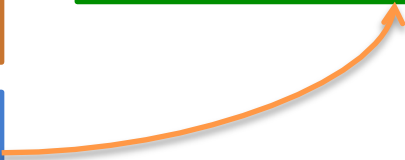

VETOR DE ESTRUTURAS

- Quando precisamos de mais de um vetor do mesmo tamanho, e as informações possuem ligação lógica, podemos criar um vetor de estruturas
- Exemplo de uma turma com 40 alunos:

```
struct tipoAluno {  
    int numMat;  
    float CR;  
    char nome[40];  
};  
typedef struct tipoAluno tAluno;
```

```
tAluno vetAluno[40];
```

```
vetAluno[i].numMat  
vetAluno[i].CR  
vetAluno[i].nome
```



EXERCÍCIOS

1. Faça um programa que leia o quadro de cargos e salários de uma empresa e que calcule a média salarial.
 - Crie uma estrutura com os campos cargo e salário
 - Ao final do programa escreva os cargos que estão acima da média calculada
2. Faça um programa que contenha uma função que receba duas estruturas do tipo `dma`, cada uma representando uma data válida, e que devolva o número de dias que decorreram entre as duas datas.

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

APONTADORES PARA ESTRUTURAS

- Estrutura também pode ser passada como parâmetro

```
struct tipoAluno {  
    int numMat;  
    float CR;  
    char nome[40];  
};  
typedef struct tipoAluno tAluno;  
tAluno aluno;
```

- exemploFunc(aluno); → por valor
- exemploFunc(&aluno); → por referência

APONTADORES PARA ESTRUTURAS

```
void exemploFunc(tAluno aluno){
```

```
    ...
```

```
    printf("Matrícula: %d", aluno.numMat);
```

```
    printf("CR: %f", aluno.CR);
```

```
}
```

```
void exemploFunc(tAluno *aluno){
```

```
    ...
```

```
    printf("Matrícula: %d", (*aluno).numMat);
```

```
    printf("CR: %f", (*aluno).CR);
```

```
}
```

APONTADORES PARA ESTRUTURAS

- Apontadores para estruturas são tão usados que existe uma notação especial

```
(*aluno).numMat OU aluno->numMat
```

```
void exemploFunc(tAluno *aluno){  
    ...  
    printf("Matrícula: %d", aluno->numMat);  
    printf("CR: %f", aluno->CR);  
}
```

EXERCÍCIO

- Considere a estrutura ponto

```
struct tipoPonto {  
    int x;  
    int y;  
};
```

- Faça um programa que

leia 2 pontos e implemente uma única função que:

- Receba os pontos lidos como parâmetros
- Retorne a multiplicação dos pontos
- Retorne a soma dos pontos