

Curso de Engenharia de Computação

Programação de Computadores I

2º. Sem 2022

Prof. Evandrino Gomes Barros

Conteúdo da aula

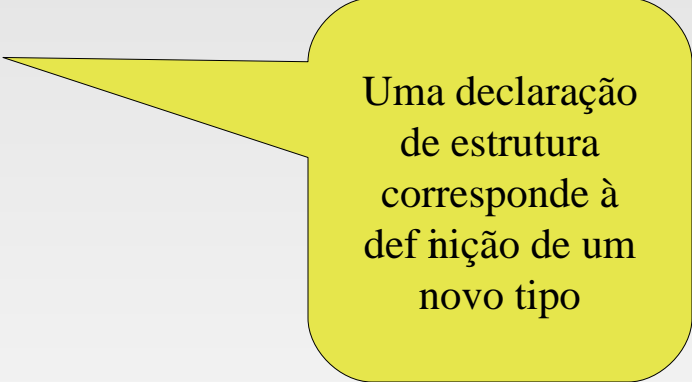
- Estruturas – Cap. 11
 - Declaração de estruturas
 - Declaração de variáveis do tipo estruturas
 - Acesso a membros de estruturas
 - Definição de novos tipos: typedef
 - Operações com estruturas

Estruturas: tipos definidos pelo programador

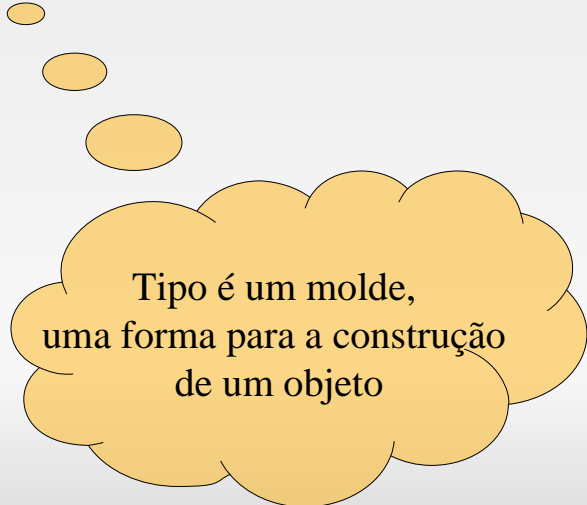
- **Estruturas** são tipos de dados compostos heterogêneos definidos pelo programador
 - permitem agrupar em uma única entidade tipos de dados diversos

- **Sintaxe**

```
struct nome_da_estrutura {  
    tipo campo1;  
    tipo campo2;  
    ...  
};
```



Uma declaração de estrutura corresponde à definição de um novo tipo



Tipo é um molde, uma forma para a construção de um objeto

Estruturas: exemplo

```
struct endereco {  
    char nome[TAM_STRING];  
    char rua[TAM_STRING];  
    unsigned int numero;  
    char cidade[TAM_STRING];  
    char estado[2];  
    char telefone[10];  
    unsigned long int cep;  
};
```

- Cada componente é denominado campo ou membro da estrutura
- Uma vez definido o tipo, a definição das variáveis é feita da mesma forma:

 struct endereco escola;

 struct endereco cliente[100], address, *lista;
- Acesso aos componentes é feito pelo operador `.`

 variável.campo

 escola.nome

 cliente[i].estado

Estruturas: exemplos

```
▪ struct Pessoa {  
    int cpf;  
    int idade;  
    char sexo;  
    char *nome;  
    char *mae, *pai;  
} cliente, Paulo, Tereza;
```

Declaração de
variáveis do tipo
struct Pessoa

- **Definição de estrutura cria um tipo**
 - especifica como a estrutura é composta, quais são seus membros e de que tipo
- a definição da estrutura não define variáveis
 - a declaração de variáveis deve ser explícita
- Atribuição de valores para variáveis
 - cliente.nome = "José Santos";
 - cliente.idade = 27;

Tipo é um molde, uma
forma para a construção
de um objeto

Operações permitidas em estruturas

```
struct item {  
    int modelo;  
    int codigo;  
    float preço;  
};
```

- inicialização na definição de **variáveis**

```
struct item exemplo = {1505, 253, 342.00};
```

- assinalamento `struct item encomenda = exemplo;`
- acesso aos membros `encomenda.preco = 329.00;`
- tomar o endereço `&exemplo`

Estruturas: atribuição de valores e cópia

```
int main() {
    struct exemplo {
        char c;
        int i;
        float f;
        double d;
    };

    struct exemplo s1, s2;
    s1.c = 'a';          // '.' referencia o membro
    s1.i = 1;
    s1.f = 3.14;
    s1.d = 0.00093;
    s2 = s1;              //atribuição!
    printf("a variável s2 contém %c %d %f %f\n",s2.c,s2.i,s2.f,s2.d);
    s1.d = 0.00011;      // essa atribuição não afeta s2.d !
}
```

Estruturas aninhadas

- estruturas dentro de estruturas

```
struct ponto { //define as coordenadas de um ponto no plano
    int x;
    int y;
};

struct ponto p1, p2;    // membros p1.x, p1.y, p2.x, p2.y

struct ponto max = {320,200};

double distancia_origem (struct ponto p) {
    return sqrt(p.x * p.x + p.y * p.y);}

struct retangulo { //define um retangulo com um par de pontos
    struct ponto r1;    //ponto inferior esquerdo
    struct ponto r2;    //ponto superior direito
};

struct retangulo tela = {{0,0},{800,600}};
```


Uso de estruturas em funções

//função que inicializa um ponto

```
struct ponto inicializa(int x, int y){
```

```
    struct ponto temp;
```

```
    temp.x = x;
```

```
    temp.y = y;
```

```
    return temp;
```

```
}
```

```
tela.r1 = inicializa(0,0);
```

```
tela.r2 = inicializa(XMAX, YMAX);
```

```
meio = inicializa (
```

```
(tela.r1.x + tela.r2.x)/2,
```

```
(tela.r1.y + tela.r2.y)/2);
```

/*função que verifica se um ponto

é interior a um retângulo */

```
int interno (struct ponto p,          struct  
             retangulo r)
```

```
{
```

```
    return (p.x > r.r1.x &&
```

```
            p.x < r.r2.x &&
```

```
            p.y > r.r1.y &&
```

```
            p.y < r.r2.y);
```

```
}
```

Apontadores para estruturas

- se uma estrutura grande deve ser passada para uma função, normalmente é mais eficiente passar um apontador para a estrutura do que copiar a estrutura
- notação especial
 - se `ap` é um apontador para estrutura então
`ap->membro` refere-se a um campo da struct

- como referenciar membros usando apontadores

```
struct ponto *ap; // apontador para objeto do tipo ponto
```

```
struct ponto origem;
```

```
ap = &origem;
```

```
printf("%d %d\n", (*ap).x, (*ap).y ); // ( ) NECESSÁRIOS
```

```
printf("%d %d\n", ap->x, ap->y);
```

Vetores de estruturas

- Podemos construir vetores de qualquer tipo, inclusive dos tipos definidos pelo usuário, como as estruturas
- Considere o problema de contar o número de ocorrências de cada palavra que aparece em um texto

```
struct item {  
    char * palavra;  
    int contador;  
    } indice[MAXPALAVRAS];
```

define um vetor de itens em que cada item contém uma palavra encontrada no texto e um contador de ocorrências da palavra

typedef

- uma declaração com o prefixo typedef declara um novo nome para um tipo => produz sinônimos para nomes
 - typedef é uma palavra reservada da linguagem
- exemplos:

```
typedef char * Pchar; // diz que Pchar é um char *
```

```
Pchar p1, p2; // p1 e p2 são apontadores para char
```

```
char *p3 = p1; // mesmo tipo
```

```
typedef unsigned char uchar;
```

define o sinônimo uchar para unsigned char;

```
typedef int inteiro; //define inteiro como sinônimo de int
```

```
typedef double real; //define real como sinônimo de double
```

typedef e estruturas

```
typedef struct Pessoa {
```

```
    int idade;
```

```
    char sexo;
```

```
    char nome[100];
```

```
    float salario;
```

```
} PESSOA;
```

```
struct Pessoa chefe, gerente;
```

OU

```
PESSOA chefe, gerente;
```

- a questão do ESCOPO
- onde definir estruturas e typedef
 - se for em uma função, somente a função conhece a definição
 - está no escopo da função
 - para serem visíveis em todo o programa devem ser declaradas fora de qualquer função e no início do programa

Estudar

- Fazer os exercícios dos capítulos 6 a 11
- Implementar os exercícios destes capítulos
- Terceira Avaliação: dia 28/11 (próxima aula)
- Matéria: capítulos 1 a 11 (toda matéria!)
- Valor: 25 pontos!



STUDY
HARD

