
Laboratório de Introdução à Ciência da Computação I

Aula 10 – Struct, union, typedef

Professor: Jó Ueyama

Monitor PAE: Fernando Alva

Struct

```
//definição
struct coord{
    int x;
    int y;
};
```

```
//declaração
struct coord primeira;
```

```
//atribuição
primeira.x = 10;
primeira.y = 20;
```

```
//definição e declaração
struct coord{
    int x;
    int y;
}primeira;
```

```
//definição, declaração e
atribuição
struct coord{
    int x;
    int y;
}primeira = {10, 20};
```

Struct

//definição e declaração de mais de uma instância

```
struct coord{  
    int x;  
    int y;  
}primeira, segunda;
```

//definição e declaração e atribuição

```
struct coord{  
    int x;  
    int y;  
}primeira = {10, 20}, segunda = {50,60};
```

Struct - exemplo

```
#include <iostream>
struct coord{
    int x;
    int y;
} primeira, segunda;

void mostra(struct coord temp);

int main(int argc, char *argv[]){
    primeira.x = 10;
    primeira.y = 20;
    segunda = primeira;
    mostra(segunda);

    system("PAUSE");
    return EXIT_SUCCESS;
}

void mostra(struct coord temp){
    printf("x = %d, y = %d \n", temp.x, temp.y);
}
```

Struct que contem struct

```
//definição e declaração
struct coord{
    int x;
    int y;
}supesq, infdir;
```

```
//definição e declaração
struct retangulo{
    struct coord supesq;
    struc coord infdir;
}meuretangulo;
```

```
//atribuição
meuretangulo.supesq.x = 10;
meuretangulo.supesq.y = 20;
```

Struct - vetores/matrizes

```
//definição e declaração
```

```
struct registro{  
    char nome[30];  
    char snome[30];  
    int idade;  
}cliente;
```

```
//atribuição
```

```
strcpy(cliente.nome,"Joao");  
strcpy(cliente.snome,"Alves");  
cliente.idade = 20;
```

Struct - vetores/matrizes

```
//definição e declaração
```

```
struct registro{  
    char nome[30];  
    char snome[30];  
    int idade;  
}cliente[2];
```

```
//atribuição
```

```
strcpy(cliente[0].nome,"Joao");  
strcpy(cliente[0].snome,"Alves");  
cliente[0].idade = 20;
```

```
//copiando registro
```

```
cliente[1] = cliente[0];
```

Union

- Declarada e usada do mesmo modo que uma “struct”
- Todos os membros de uma união ocupam uma mesma área de memória (ocorre sobreposição)
- Somente um valor de cada vez pode ser armazenado numa “union” (lógica OR)

```
//definição
union coord{
    int x;
    int y;
};
```

```
//declaração
union coord primeira;
//atribuição
primeira.x = 10;
```

Typedef

//definição de um novo nome para um tipo existente

```
typedef int inteiro;
```

```
inteiro x;
```

//definição para tipos compostos com struct

```
typedef struct{
```

```
    int x;
```

```
    int y;
```

```
}coord;
```

//declaração

```
coord supesq, infdir;
```

Exercício

Considere a seguinte estrutura:

```
typedef struct
{
int NumConta;
char Cliente[100];
float Saldo;
int Senha;
char Chave;
} contabancaria;
```

faça um programa com os requisitos:

- a) Cria uma conta
- b) Consulta o saldo do cliente (entra com o numero da conta conferindo apenas a senha)
- c) Deposita um valor (entra com o numero da conta e confere o nome do cliente)
- d) Sacar um valor (entra com o numero da conta e confere senha e chave, o cliente tem apenas autorização de sacar o seu dinheiro, conta sem limite)
- e) Encerre a conta (entra com o numero da conta e confere senha e chave, faça uma pergunta para que o cliente confira a operação e apague seus dados)

Obs.: Para cada item faça uma função.

05/12/11