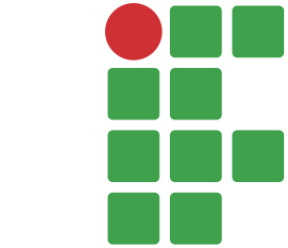
ESTRUTURAS



INSTITUTO FEDERAL
Paraíba

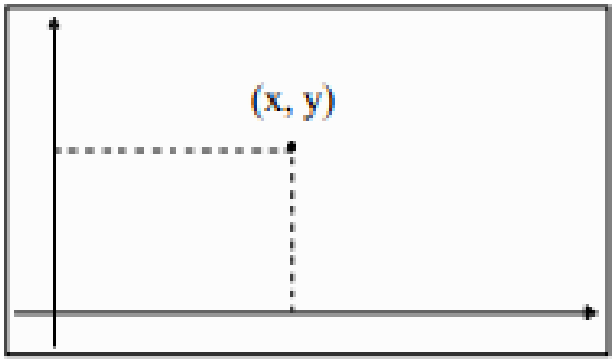
Professor Msc Paulo de Tarso F. Júnior

paulodt@gmail.com

Estruturas

- ***Struct*** são coleções de dados heterogêneos agrupados em uma mesma estrutura de dados

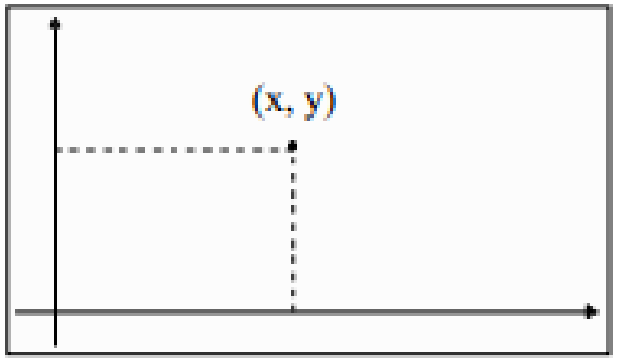
- Ex: armazena as coordenadas (x,y) de um ponto:



Estruturas

- Declaração:

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



- a estrutura contém dois inteiros, x e y
- p1 e p2 são duas variáveis tipo ***struct*** contendo duas coordenadas cada.

Declaração

- Formato da declaração:

```
struct nome_da_estrutura {
        tipo_1 dado_1;
        tipo_2 dado_2;
        ...
        tipo_n dado_n;
} lista_de_variaveis;
```

- A estrutura pode agrupar um número arbitrário de dados de tipos diferentes
- Pode-se nomear a estrutura para referencia-la

Nomeando uma estrutura

struct ponto define um novo tipo de dado

```
struct {
        int x;
        int y;
} p1;
struct {
        int x;
        int y;
} p2;
```

Repetição

```
struct ponto {
        int x;
        int y;
}; struct ponto p1, p2;
```

Pode-se definir novas variáveis do tipo ponto

Estruturas

- acesso aos dados:
  - struct-var.campo
- Ex:
  - p1.x = 10; /*atribuição */
  - p2.y = 15;
  - if (p1.x >= p2.x) && (p1.y >= p2.y) ...

Atribuição de Estruturas

- Inicialização de uma estrutura:
  - struct ponto p1 = { 220, 110 };
- Atribuição entre estruturas do mesmo tipo:
  - struct ponto p1 = { 220, 110 };
  - struct ponto p2; p2 = p1; /* p2.x = p1.x e p2.x = p1.y */
- Os campos correspondentes das estruturas são automaticamente copiados do destino para a fonte

Atribuição de Estruturas

- Atenção para estruturas que contenham ponteiros:

```
struct aluno {
        char *nome;
        int idade;
} a1, a2;
a1.nome = “Afranio”;
a1.idade = 32;
a2 = a1;
```

- Agora a1 e a2 apontam para o mesmo string nome:
  a1.nome == a2.nome == “Afranio”

Composição de Estruturas

```
struct retangulo {
        struct ponto inicio;
        struct ponto fim;
};
struct retangulo r = { { 10, 20 }, { 30 , 40 } };
```

- Acesso aos dados:
  - r.inicio.x += 10;
  - r.inicio.y -= 10;

Estruturas como parâmetros

```
struct ponto cria_ponto (int x, int y) {
        struct ponto tmp;
        tmp.x = x;
        tmp.y = y;
        return tmp;
}
main () {
        struct ponto p = cria_ponto(10, 20);
}
```

Operações

- Operações entre membros das estruturas devem ser feitas membro a membro:

```
/* retorna uma cópia de p1 = p1 + p2 */
struct soma_pts (struct ponto p1, struct ponto p2) {
        p1.x += p2.x;
        p1.y += p2.y;
        return p1; /* retorna uma copia de p1 */
}
```

Ponteiros para Estruturas

- Estruturas grandes são passadas como parâmetro de forma mais eficiente através de ponteiros

```
struct ponto *pp;
struct ponto p1 = { 10, 20 };
pp = &p1;
printf("Ponto P1: (%d %d)\n", (*pp).x, (*pp).y);
```

- acesso via operador "->":

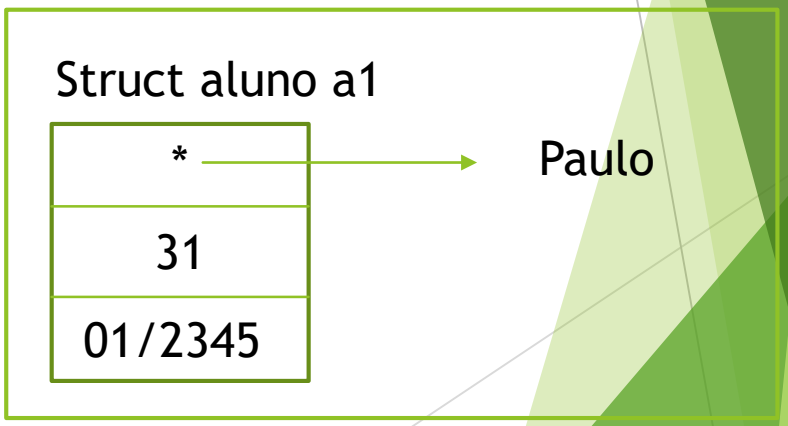
```
printf("Ponto P1: (%d %d)\n", pp->x, pp->y};
```

Arrays de Estruturas

```
struct ponto arp[10];
/* cria um array de 10 pontos */
arp[1].x = 5; /*atribui 5 a coordenada x do 2º ponto */
struct jogador {
        char *nome;
        int idade;
};
struct jogador Brasil[11] = {
        “Felix”, 32,
        “Carlos Alberto”, 24, ...
};
```

Espaço para uma Estrutura

```
struct aluno {
        char *nome;             /* ponteiro 4 bytes */
        short idade;            /* 4 bytes */
        char matricula[8];      /* array 8 bytes
};
struct aluno al;
al.nome = “Paulo”;
al.idade = 31;
strcpy(al.matricula, “01/2345”);
```



Exercícios

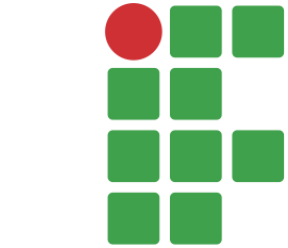
1. Escrever um programa que cadastre o nome, a matrícula e duas notas de vários alunos. Em seguida imprima a matrícula, o nome e a média de cada um deles.

2. Escrever um programa que cadastre o nome, a altura, o peso, o cpf e sexo de algumas pessoas. Com os dados cadastrados, em seguida localizar uma pessoas através do seu CPF e imprimir o seu IMC (**peso (quilos) ÷ altura^2 (metros)**).

Referências

- Material baseado na aula “Curso C: Estruturas”do professor Ricardo Pezzuol Jacobi – Departamento de Ciência da Computação Universidade de Brasília
- Trabalhando com Struct. Disponível em: https://programacaodescomplicada.wordpress.com/2012/08/16/aula-36-trabalhando-com-struct/. Acesso em 08/08/2017

ESTRUTURAS



INSTITUTO FEDERAL
Paraíba

Professor Msc Paulo de Tarso F. Júnior

paulodt@gmail.com