Estruturas de Dados
Aula 7: Tipos Estruturados

Tipos Estruturados

- Permite estruturar dados complexos, nos quais as informações são compostas por diversos campos
- Tipo estrutura
  - Agrupa diversas variáveis dentro de um contexto

```
struct ponto {
    float x
    float y;
};
```

- Declaração da variável do tipo ponto:
  - struct ponto p;
  - p.x = 10.0;
  - p.y = 5.0;

Ponteiros para estruturas

- struct ponto *pp;
- Para acessar os campos
  - (*pp).x = 12.0;
- De maneira simplificada
  - pp->x = 12.0;
- Para acessar o endereço de um campo
  - &pp->x

Passagem de estruturas para funções

- A estrutura inteira é copiada para pilha

```
void imprime (struct ponto p)
{
  printf("O ponto fornecido foi: (%.2f, %.2f)\n", p.x, p.y);
}
```

- Apenas o ponteiro é copiado para pilha

```
void imprime (struct ponto* pp)
{
  printf("O ponto fornecido foi: (%.2f, %.2f)\n", pp->x, pp->y);
}
```

Passagem de estruturas para funções

- Exemplo para ler coordenadas do ponto
  - Precisamos usa ponteiro!

```
void captura (struct ponto* pp)
{
    printf ("Digite as coordenadas do ponto (x, y): ");
    scanf ("%f %f", &pp->x, &pp->y);
}

int main ()
{ struct ponto p;
  captura (&p);
  imprime (&p);
  return 0; }
```

Alocação Dinâmica de estruturas

- Podemos alocar estruturas em tempo de execução do programa

```
struct ponto* p;
p = (struct ponto*) malloc (sizeof (struct ponto));
```

- Para acessar as coordenadas:

```
...
p->x = 12.0;
...
```

Definição de novos tipos

- Podemos criar nomes de tipos em C
  - typedef float Real;
  - Real pode ser usado como mnemônico de float

```
typedef unsigned char UChar;
typedef int* PInt;
typedef float Vetor[4];
```

- Podemos declarar as seguintes variáveis:

```
Vetor v;
...
v[0] = 3;
...
```

Definição de novos tipos

- Podemos definir nomes para tipos estruturados

```
struct ponto {
   float x;
   float y;
}
typedef struct ponto Ponto;
typedef struct ponto *PPonto;
typedef struct ponto Ponto, *PPonto;
```

- Podemos definir as variáveis:

```
Ponto p;
PPonto pp;
```

Aninhamento de estruturas

- Campos de uma estrutura podem ser outras estruturas previamente definidas
- Exemplo: função que calcula distância entre pontos

```
float distancia (Ponto* p, Ponto* q)
{
   float d = sqrt ((q->x-p->x)*(q->x-p->x) + (q->y-p-
    >y)*(q->y-p->y));
   return d;
}
```

Aninhamento de estruturas

- Podemos representar um círculo como

```
struct circulo {
  float x, y;  //centro do círculo
  float r;  //raio
}
```

- Como já temos o tipo Ponto definido:

```
 struct circulo {
  Ponto p;
  float r;
}

typedef struct circulo Circulo;
```

Aninhamento de estruturas

- Para implementar uma função que determinar se um dado ponto está dentro de um círculo
  - Podemos usar a função da distância, visto que usamos o tipo ponto na definição do círculo

```
int interior (Circulo* c, Ponto* p)
{
   float d = distancia (&c->p, p);
   return (d<c->r);
}
```

Tipo União

- Localização de memória compartilhada por diferentes variáveis
  - Podem ser de tipos diferentes
  - São usadas para armazenar valores heterogêneos em um mesmo espaço de memória
  - Um único elemento da união pode estar armazenado!

```
union exemplo {
   int i;
   char c;
}
```

- Para declarar a variável:
  - union exemplo v;
- Para acessar os elementos (“.” ou “->”):
  - v.i = 10 ou v.c = ‘x’;