Estruturas de Dados

Módulo 8 – Tipos Estruturados

Referências

Waldemar Celes, Renato Cerqueira, José Lucas Rangel, *Introdução a Estruturas de Dados*, Editora Campus (2004)

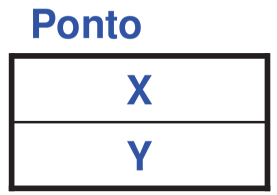
Capítulo 8 – Tipos estruturados

Tópicos

- Tipo estrutura
- Definição de novos tipos
- Aninhamento de estruturas
- Vetores de estruturas
- Tipo união
- Tipo enumeração

Tipo Estrutura

- Motivação:
  - manipulação de dados compostos ou estruturados
  - Exemplos:
    - ponto no espaço bidimensional
      - representado por duas coordenadas (x e y), mas tratado como um único objeto (ou tipo)
    - dados associados a aluno:
      - aluno representado pelo seu nome, número de matrícula, endereço, etc ., estruturados em um único objeto (ou tipo)



Tipo Estrutura

- Tipo estrutura:
  - tipo de dado com campos compostos de tipos mais simples
  - elementos acessados através do operador de acesso “ponto” (.)

```
struct ponto                    /* declara ponto do tipo struct */
{  float x;
   float y;
};
...
struct ponto p;                 /* declara p como variável do tipo struct ponto */
...
p.x = 10.0;                     /* acessa os elementos de ponto */
p.y = 5.0;
```

Tipo Estrutura

```
/* Captura e imprime as coordenadas de um ponto qualquer */
#include <stdio.h>
struct ponto {
   float x;
   float y;
};
int main (void)
{
   struct ponto p;
   printf("Digite as coordenadas do ponto(x y): ");
   scanf("%f %f", &p.x, &p.y);
   printf("O ponto fornecido foi: (%.2f,%.2f)\n", p.x, p.y);
   return 0;
}
```

Basta escrever `&p.x` em lugar de `&(p.x)`.

O operador de acesso ao campo da estrutura tem precedência sobre o operador "endereço de"

Tipo Estrutura

- Ponteiros para estruturas:
  - acesso ao valor de um campo x de uma variável estrutura p: p.x
  - acesso ao *valor* de um campo x de uma variável ponteiro pp: pp->x
  - acesso ao *endereço* do campo x de uma variável ponteiro pp: &pp->x

```
struct ponto *pp;

(*pp).x = 12.0;        /* formas equivalentes de acessar o valor de um campo x */

pp->x = 12.0;
```

Tipo Estrutura

- Passagem de estruturas para funções – por valor:
  - análoga à passagem de variáveis simples
  - função recebe toda a estrutura como parâmetro:
    - função acessa a cópia da estrutura na pilha
    - função não altera os valores dos campos da estrutura original
    - operação pode ser custosa se a estrutura for muito grande

```
/* função que imprima as coordenadas do ponto */
void imprime (struct ponto p)
{
   printf("O ponto fornecido foi: (%.2f,%.2f)\n", p.x, p.y);
}
```

Tipo Estrutura

- Passagem de estruturas para funções – por referência:
  - apenas o ponteiro da estrutura é passado, mesmo que não seja necessário alterar os valores dos campos dentro da função

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

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

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

Correção da segunda linha - função captura:
scanf("%f %f", &pp->x, &pp->y);

Tipo Estrutura

- Alocação dinâmica de estruturas:
  - tamanho do espaço de memória alocado dinamicamente é dado pelo operador sizeof aplicado sobre o tipo estrutura
  - função malloc retorna o endereço do espaço alocado, que é então convertido para o tipo ponteiro da estrutura

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

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

Definição de Novos Tipos

- typedef
  - permite criar nomes de tipos
  - útil para abreviar nomes de tipos e para tratar tipos complexos

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

Vetor v;    /* exemplo de declaração usando Vetor */
...
v[0] = 3;
```

  - UChar o tipo char sem sinal
  - PInt um tipo ponteiro para int
  - Vetor um tipo que representa um vetor de quatro elementos

Definição de Novos Tipos

- typedef
  - Exemplo: definição de nomes de tipos para as estruturas

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

typedef struct ponto Ponto;
typedef struct ponto *PPonto;
```

  - ponto representa uma estrutura com 2 campos do tipo float
  - Ponto representa a estrutura ponto
  - PPonto representa o tipo ponteiro para a estrutura Ponto

Definição de Novos Tipos

- typedef
  - Exemplo: (definição utilizando um só typedef)

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

typedef struct ponto Ponto, *PPonto;
```

  - ponto representa uma estrutura com 2 campos do tipo float
  - Ponto representa a estrutura ponto
  - PPonto representa o tipo ponteiro para a estrutura Ponto

Definição de Novos Tipos

- typedef
  - Exemplo: (definição em um comando só)

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

  - ponto representa uma estrutura com 2 campos do tipo float
  - Ponto representa a estrutura ponto

Aninhamento de Estruturas

- Aninhamento de estruturas:
  - campos de uma estrutura podem ser outras estruturas
  - Exemplo:
    - definição de Círculo usando Ponto

```
struct circulo {
   Ponto p;          /* centro do círculo */
   float r;          /* raio do círculo */
};

typedef struct circulo Circulo;
```

```
/* Função para determinar se um ponto está ou não dentro de um círculo:
            entrada:   ponteiros para um círculo e para um ponto
            saída:     1 = ponto dentro do círculo
                       0 = ponto fora do círculo
*/
int interior (Circulo* c, Ponto* p)
{
  float d = distancia(&c->p, p);
  return (d < c->r);
}

/* Função para a calcular distância entre 2 pontos:
            entrada:   ponteiros para os pontos
            saída:     distância correspondente
*/
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;
}
```

&c->p : ponteiro para centro de c

p : ponteiro para o ponto

c->r : raio do círculo

d < c->r : testa se d é menor do raio

cálculo da distância:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

sqrt da biblioteca math.h

```
#include <stdio.h>
#include <math.h>
typedef struct ponto {
   float x;
   float y;
} Ponto;

typedef struct circulo {
   Ponto p;              /* centro do círculo */
   float r;              /* raio do círculo */
} Circulo;

int main (void)
{ Circulo c;
   Ponto p;
   printf("Digite as coordenadas do centro e o raio do circulo:\n");
   scanf("%f %f %f", &c.p.x, &c.p.y, &c.r);
   printf("Digite as coordenadas do ponto:\n");
   scanf("%f %f", &p.x, &p.y);
   printf("Pertence ao interior = %d\n", interior(&c,&p));
   return 0;
}
```