



Universidade Federal do Ceará
Departamento de Computação
Curso de Ciência da Computação

Disciplina	Programação - CK0226	Semestre	2018/2
Professor	Lincoln Souza Rocha		
Laboratório de Programação 04 – Tipos Estruturados e Tipos Abstratos de Dados			

Descrição Geral do Exercício de Codificação

Comandos para Compilação de Módulos C em Separado:

```
$ gcc -c <tad-impl>.c -o <tad-impl>.o  
$ gcc -c <prog-usa-tad>.c -o <prog-usa-tad>.o  
$ gcc -o <prog-final>.bin <tad-impl>.o <prog-usa-tad>.o
```

Este exercício consiste em criar uma biblioteca para manipular figuras geométricas. Para isso, você irá construir tipos estruturados que representam figuras geométricas e desenvolver funções que realizam operações sobre essas figuras geométricas. Dessa forma, você deverá exportar os seus tipos estruturados e as suas funções por meio de um arquivo de descrição (arquivo com extensão .h). Os tipos estruturados e as operações (.h) a serem codificados são:

1) Ponto, composto por x e y do tipo `float`

```
/* TAD: Ponto (x,y) */  
/* Tipo exportado */  
typedef struct ponto Ponto;  
  
/* Funções exportadas */  
  
/* Função criaP - Aloca e retorna um ponto com coordenadas (x,y) */  
Ponto* criaP(float x, float y);  
  
/* Função liberaP - Libera a memória de um ponto previamente criado */  
void liberaP(Ponto* p);  
  
/* Função acessaP - Retorna os valores das coordenadas de um ponto */  
void acessaP(Ponto* p, float* x, float* y);  
  
/* Função atribuiP - Atribui novos valores às coordenadas de um ponto */  
void atribuiP(Ponto* p, float x, float y);  
  
/* Função distanciaP - Retorna a distância entre dois pontos */  
float distanciaP(Ponto* p1, Ponto* p2);
```

2) Círculo, composto por um ponto e um raio do tipo `float`

```
/* TAD: Circulo (ponto,raio) */  
/* Tipo exportado */  
typedef struct circulo Circulo;  
  
/* Funções exportadas */  
  
/* Função criaC - Aloca e retorna um circulo com base no ponto e no raio informados */  
Circulo* criaC(Ponto* p, float raio);  
  
/* Função liberaC - Libera a memória de um circulo previamente criado */  
void liberaC(Circulo* c);  
  
/* Função acessaC - Retorna os valores das coordenadas de um circulo e seu raio */  
void acessaC(Circulo* c, float* x, float* y, float* r);  
  
/* Função atribuiC - Atribui novos valores às coordenadas de um ponto e seu raio */
```

```

void atribuiC(Circulo* c, float x, float y, float r);

/*Função pertenceC - Retorna 1 se o ponto pertence ao circulo ou 0, caso contrário */
int pertenceC(Circulo* c, Ponto* p);

/* Função areaC - Retorna o cálculo da área do circulo */
float areaC(Circulo* c);

```

3) Triângulo, composto por três pontos

```

/* TAD: Triangulo (ponto,ponto,ponto) */
/* Tipo exportado */
typedef struct triangulo Triangulo;

/* Funções exportadas */

/* Função criaT - Aloca e retorna um triangulo com base nos pontos */
Triangulo* criaT(Ponto* p1, Ponto* p2, Ponto* p3);

/* Função liberaT - Libera a memória de um triangulo previamente criado */
void liberaT(Triangulo* t);

/* Função acessaT - Retorna os valores dos pontos de um triângulo */
void acessaT(Triangulo* t, Ponto* p1, Ponto* p2, Ponto* p3);

/* Função atribuiT - Atribui novos valores aos pontos de um triângulo */
void atribuiT(Triangulo* t, Ponto* p1, Ponto* p2, Ponto* p3);

/* Função verificaT - Retorna 1 se os pontos do triângulo atendem a condição de
existência e 0, caso contrário */
int verificaT(Triangulo* t);

/* Função pertenceT - Retorna 1 se o ponto pertence ao triângulo ou 0, caso contrário
*/
int pertenceT(Triangulo* t, Ponto* p);

/* Função areaT - Retorna o cálculo da área do triângulo */
float areaT(Triangulo* t);

```

4) Quadrilátero, composto por quatro pontos.

```

/* TAD: Quadrilatero (ponto,ponto,ponto,ponto) */
/* Tipo exportado */
typedef struct quadrilatero Quadrilatero;

/* Funções exportadas */

/* Função criaQ - Aloca e retorna um quadrilatero com base nos pontos */
Quadrilatero* criaQ(Ponto* p1, Ponto* p2, Ponto* p3, Ponto* p4);

/* Função liberaQ - Libera a memória de um quadrilatero previamente criado */
void liberaQ(Quadrilatero* q);

/* Função acessa - Retorna os valores dos pontos de um quadrilatero */
void acessaQ(Quadrilatero* q, Ponto* p1, Ponto* p2, Ponto* p3, Ponto* p4);

/* Função atribuiQ - Atribui novos valores aos pontos de um quadrilatero */
void atribuiQ(Quadrilatero* q, Ponto* p1, Ponto* p2, Ponto* p3, Ponto* p4);

/* Função pertenceQ - Retorna 1 se o ponto pertence ao quadrilatero ou 0, caso
contrário */
int pertenceQ(Quadrilatero* q, Ponto* p);

/* Função areaQ - Retorna o cálculo da área do quadrilatero */
float areaQ(Quadrilatero* q);

```