Problemas de geometria

Implemente as funções declaradas no arquivo `geometria.h` (veja a listagem do código desse arquivo abaixo).

```
#ifndef GEOMETRIA
#define GEOMETRIA

struct s_ponto {
  double x, y;
};

typedef struct s_ponto ponto;
typedef struct s_ponto vetor;

struct s_segmento {
  ponto p, q;
};

typedef struct s_segmento segmento;

struct s_triangulo {
  ponto p, q, r;
};

typedef struct s_triangulo triangulo;

struct s_retangulo {
  ponto ie;    // inferior esquerdo
  ponto sd;    // superior direito
};

typedef struct s_retangulo retangulo;

//  01. Calcula o produto interno <u,v>
double produto_interno(vetor u, vetor v);

//  02. Calcula o vetor u - v
vetor subtrai(vetor u, vetor v);
```

```c
/*  03. Calcula o vetor resultante da rotação do vetor v
    de um ângulo de 90 graus no sentido anti-horário. */
vetor roda90(vetor v);

//  04. Calcula a distância entre os pontos p e q.
double distancia(ponto p, ponto q);

/*  05. Retorna 1 se o coseno do ângulo entre os vetores u e v é
    positivo e retorna -1 se for negativo e 0 se for nulo.
    Se u ou v for o vetor nulo, devolve 0. */
int sinal_do_coseno(vetor u, vetor v);

/*  06. Retorna 1 se p, q e r estão em sentido horário e -1 se for
    anti-horário. Se os pontos forem colineares devolva 0.
    Se dois desses pontos são iguais, devolve 0. */
int sentido(ponto p, ponto q, ponto r);

/*  07. Retorna 1 se os interiores dos segmentos se intersectam em
    um único ponto e retorna 0 caso contrário. */
int cruza(segmento s, segmento t);

/*  08. Retorna 1 se o ponto p está no interior do triângulo t
    e retorna 0 caso contrário. */
int dentro(ponto p, triangulo t);

/*  09. Devolve 1 se um retângulo é vazio e 0 caso contrário. */
/*  Um retângulo é vazio se a extremidade inferior esquerda
    (ponto ie da struct) se encontra estritamente a direita
    ou estritamente acima da extremidade superior direita
    (ponto sd da struct). */
int retangulo_vazio(retangulo r);

/*  10. Devolve o retângulo resultante da intersecção de
    dois retângulos fechados passados como argumento.
    Se a intersecção é vazia, qualquer representação
    de retângulo vazio serve como resposta da funçao! */
retangulo intersecta_ret(retangulo a, retangulo b);

// ################################################################
// OPCIONAL OPCIONAL OPCIONAL OPCIONAL OPCIONAL OPCIONAL OPCIONAL
// ################################################################
```

```c
// A partir daqui os exercícios são opcionais (e podem substituir
// alguma implementação de função acima que você não tenha feito).

/* 11. Devolve o ponto em que s e t se intersectam caso cruza(s, t)
   devolve 1 (ou seja, caso s e t se intersectem em um
   único ponto no interior dos dois segmentos) e devolve
   o ponto (0, 0) caso contrário eles não se intersectem em
   um único ponto. */
ponto cruzamento(segmento s, segmento t);

/* 12. Calcula o ponto que é a projeção ortogonal de p na reta que
   contém o segmento s. Ou seja, devolve o ponto da reta que
   contém s que é o mais próximo de p. */
ponto projeta(ponto p, segmento s);

/* 13. Devolve 1 se o triângulo é degenerado, isto é
   se seus três vértices são colineares e 0 caso contrário. */
int degenerado(triangulo t);

/* 14. Devolve 1 se o interior dos triângulos a e b se
   intersectam e devolve 0 caso contrário. */
int intersecta_tri(triangulo a, triangulo b);

#endif
```

Observações

As funções foram declaradas numa certa ordem. A ordem foi pensada de modo que, na hora de implementar uma função é bem provável que você precise chamar alguma função que foi declarada antes dela (exceto pelas primeiras que são muito simples).

Geometria analítica

Se você não se lembra de alguma definição, pode pesquisar no Google, ou no livro que você usou para o curso de geometria analítica. Algumas lembranças sobre conceitos de geometria analítica serão dadas em sala, durante a aula prática. Algumas funções da lista só podem ser implementadas de um jeito, outras têm soluções variadas, vai depender da modelagem matemática que você adotar para resolver cada problema geométrico associado.