Primeira lista de Geometria Computacional

Entrega dia 13/10.
Mande para walterfm@ime.usp.br,
assunto: lista 1 seu #USP.

O objetivo desta lista é exercitar os conceitos básicos discutidos até a aula de 13/09. Para isto você implementará conceitos na linguagem de sua escolha dentre C, C++, Java ou C#.

Como pré-requisito para os exercícios, defina quais classes ou structs você usará para representar pontos, polígonos e segmentos. Você deverá lidar com pontos com coordenadas do tipo `int`, que está presente nas quatro linguagens acima. O projeto destas classes/structs faz parte dessa lista e será avaliado.

Seu código deverá ter as seguintes funcionalidades:

1. Determinação do sentido de um triângulo. Implemente uma função ou método `ccw` que toma três pontos como argumento e retorna 0 se eles estiverem alinhados, 1 se o triângulo formado por eles for anti-horário e -1 se o triângulo for horário.

2. Intersecção de segmentos. Dados dois segmentos, faça uma função que retorne 1 se eles se interceptarem e 0 caso contrário.

3. Comparação das intersecções com uma reta horizontal. Dados dois segmentos que interceptam a reta $y = y_0$, faça uma função que retorne $0, \pm 1$ dependendo da ordem em que os segmentos interceptam a reta horizontal dada.

4. "Trapezoidalização". Implemente uma função $O(n \log n)$

   ```
   int trapezoides(const char* destFile, const char* sourceFile)
   ```

   que lê os vértices de um polígono do arquivo `sourceFile` e escreve a sua decomposição em trapezóides com topo e base horizontais no arquivo `destFile`. A função deve retornar 1 se ambos arquivos forem válidos e 0 caso contrário. O arquivo `sourceFile` estará no formato

   ```
   nVertices
   x1,y1
   x2,x2
   x3,y3
   ...
   xn,yn
   ```

   onde `nVertices`, `xi` e `yi` são inteiros no range do `int`. O arquivo `destFile` deverá estar no seguinte formato

```
nTrapezios
iBase1,iTopo1,iEsquerdo1,iDireito1,
iBase2,iTopo2,iEsquerdo2,iDireito2,
iBase3,iTopo3,iEsquerdo3,iDireito3,
...
iBasen,iTopon,iEsquerdon,iDireiton,
```

onde `nTrapezios` é o número de trapézios e cada linha contém

iBase: índice do vértice na base do trapézio,

iTopo: índice do vértice do topo do trapézio,

iEsquerdo: índice do primeiro vértice do lado esquerdo do trapézio k,

iDireito: índice do primeiro vértice do lado direito do trapézio k.

Para implementar a função `trapezoides` você deve seguir o que foi esboçado nas aulas. Você deverá usar "simplicidade simulada" para lidar com polígonos nos quais há vértices na mesma horizontal e pode usar um container da STL ou da sua biblioteca predileta para administrar a lista de segmentos cortados pela scanline, desde este container suporte as operações necessárias com a complexidade apropriada.

Observações:

1. Para avaliar o seu trabalho, eu testarei o seu código usando pontos com coordenadas no "range" do `int`, isto é, os pontos terão coordenadas no intervalo $[-2^{31}, 2^{31})$.

2. Seu código deverá lidar corretamente com polígonos "malucos", com muitas voltas e vários vértices com coordenadas x ou y coincidentes. Você pode assumir que os dados são consistentes, ou seja, não há (x, y) repetidos, só x's ou y's. Além disto, os lados não se cruzam.

3. Cuidado com as contas: a soma de dois inteiros já pode dar errado por causa de overflow.