

Trabalho II

Alienígenas invadiram o planeta Terra! Eles lançam 3 tipos de bombas: destruição, pintura e clonagem. Os alunos do Depto de Física da UEL inventaram um tinta que, quando usadas em anteparos, bloqueiam a propagação da explosão da bomba. Uma das tarefas dos alunos de Estrutura de Dados é determinar região afetada pela explosão de uma bomba.

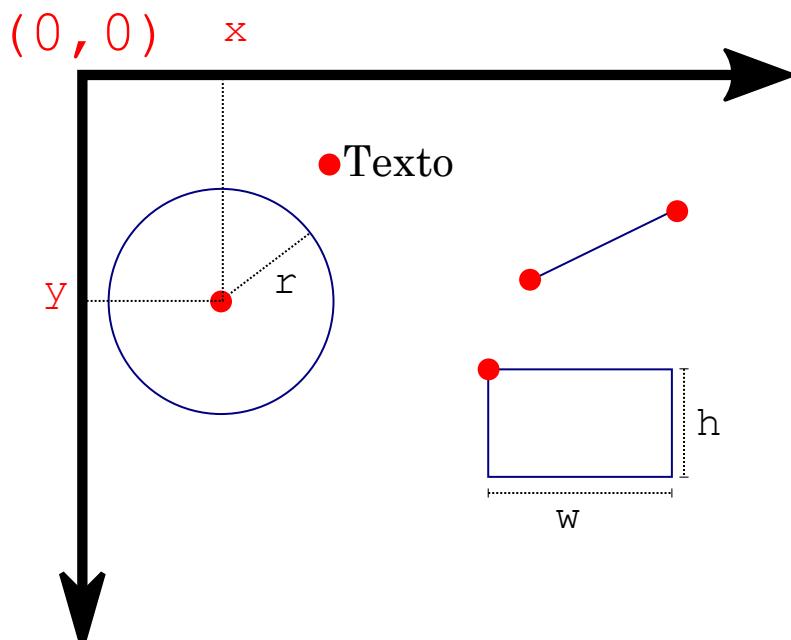
A Entrada

ATENÇÃO: Não esqueça de ler a descrição geral dos projetos (Sala de Aula).

A entrada do algoritmo será basicamente um conjunto de formas geométricas básicas (retângulos, círculos, etc) dispostos numa região do plano cartesiano .

Considere a Ilustração 1. Cada forma geométrica é definida por uma coordenada âncora (marcada, na figura, por um pequeno ponto vermelho) e por suas dimensões. A coordenada âncora do círculo é o seu centro e sua dimensão é definida por seu raio (r , na figura). A coordenada âncora do retângulo é seu canto inferior esquerdo¹ e suas dimensões são sua largura (w) e sua altura (h). A coordenada âncora de um texto, normalmente, é o início do texto, porém, pode ser definida como o meio ou o fim do texto. Por fim, uma linha é determinada por duas âncoras em suas extremidades. As coordenadas que posicionam as formas geométricas são valores reais.

Cada forma geométrica é identificada por um número inteiro.



As tabelas abaixo mostram os formatos dos arquivos de entrada (.geo e .qry). Cada comando tem um certo número de parâmetros. Os parâmetros mais comuns são:

- i, j, k : número inteiro, maior ou igual a 1. Identificador de uma forma geométrica.
- r : número real. Raio do círculo.
- x, y : números reais. Coordenada (x,y) .
- cor : string. Cor válida dentro do padrão SVG.²

¹ Note que o plano cartesiano está desenhado "de ponta-cabeça" em relação à representação usual.

comando	parâmetros	descrição
c	i x y r corb corp	Cria um círculo com identificador i: (x,y) é o centro do círculo; r, seu raio, corb é a cor da borda e corp é a cor do preenchimento
r	i x y w h corb corp	Cria um retângulo com identificador i: (x,y) é a âncora do retângulo, w é a largura do retângulo e h, a altura. corb é a cor da borda e corp é a cor do preenchimento
l	i x1 y1 x2 y2 cor	Cria uma linha com identificador i, com extremidades nos pontos (x1,y1) e (x2,y2), com a cor especificada.
t	i x y corb corp a txt	Cria o texto txt com identificador i, nas coordenadas (x, y) e com a cores indicadas. corb é a cor da borda e corp é a cor do preenchimento. O parâmetro a determina a posição da âncora do texto: i, no início; m, no meio, f, no fim. O texto txt é o último parâmetro do comando. Pode incluir espaços em branco e se estende até o final da linha.
ts	fFamily fWeight fSize	Muda o estilo dos textos (comando t) subsequentes. font family: sans (sans-serif), serif, cursive; font weight (n: normal, b: bold, b+: bolder, l: lighter)
comandos .geo		

O programa a ser executado está no arquivo .qry que contém os seguinte comandos.

comando	parâmetros	descrição
a	i j [v h]	As formas cujos identificadores estão na faixa [i,j] são pintadas com a tinta bloqueante (viram anteparos) Círculos são transformados em segmentos horizontais (h) ou verticais (v). Segmentos devem ter identificadores únicos. Cor do segmento é o mesmo da cor da borda da figura original. TXT: reportar id e tipo da figura original, id e extremos dos segmentos produzidos

2 <http://www.december.com/html/spec/colorsrgb.html>.
<https://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html>

comando	parâmetros	descrição
d	x y sfx	<i>Bomba de destruição é lançada na coordenada (x,y)</i> <i>TXT: reportar id e tipo das formas destruídas</i> <i>SVG: desenhar a região de visibilidade da bomba. A região de visibilidade deve ser desenhada em um novo arquivo cujo nome tem o sufixo sfx ou no próprio arquivo svg final, caso sfx seja “-”.</i>
p	x y cor sfx	<i>Bomba de pintura é lançada na coordenada (x, y). Formas dentro da região de visibilidade tem suas cores de preenchimento e de borda pintadas com a cor cor.</i> <i>TXT: reportar id e tipo das formas pintadas</i> <i>SVG: semelhante ao comando d.</i>
cln	x y dx dy sfx	<i>Bomba de clonagem é lançada na coordenada (x,y). Os clones são trasladados em dx, dy (nos eixos x e y, respectivamente). Note que clones devem ter identificadores únicos.</i> <i>TXT: id e tipo das figuras originais e dos clones.</i> <i>SVG: semelhante ao comando d.</i>
Comandos .qry		

Transformação em Anteparos

As figuras, quando pintadas com a tinta bloqueante, são transformadas em anteparos: a figura original deixa de existir e são substituídas por segmentos (linhas), exceto a própria linha que não é substituída (mas passa a ser anteparo). Os retângulos são substituídos por segmentos correspondentes aos seus lados. O círculo é substituído por um segmento horizontal ou vertical (em seu diâmetro), obviamente passando pelo centro do círculo (Figura 1).

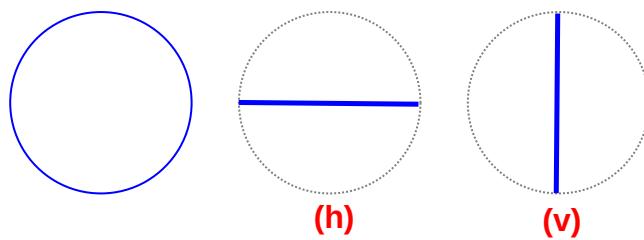


Figura 1: Substituição de um círculo por segmento

Para textos, fazemos a seguinte convenção:

Seja: (xt, yt) a âncora do texto, $|t|$ o número de caracteres do texto
O texto é considerado como se fosse um segmento $(x1, y1) - (x2, y2)$.
se posição da âncora do texto for:

- ‘i’: $x1 = xt, y1 = yt,$
 $x2 = xt + 10.0 * |t|, y2 = yt$
- ‘f’: $x1 = xt - 10.0 * |t|, y1 = yt,$
 $x2 = xt, y2 = yt$
- ‘m’: $x1 = xt - 10.0 * |t| / 2, y1 = yt,$
 $x2 = xt + 10.0 * |t| / 2, y2 = yt$

comprimento da linha (cl):
 $10.0 * |abcde| = 50.0$

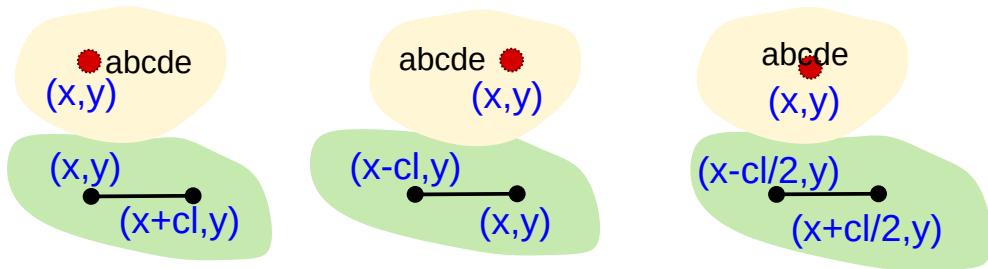


Figura 2: Substituição texto para segmento

A saída textual é um arquivo texto comum contendo as informações solicitadas pelas instruções contidas no arquivo .qry.

Ao final do processamento do arquivo .geo, deve ser produzido um arquivo .svg mostrando todas as formas existentes. Da mesma forma, ao final do processamento do arquivo .qry, outro arquivo .svg deve ser produzido mostrando as formas remanescente, além das “anotações” indicadas na descrição dos comandos. Note que instruções do .qry podem remover formas (que não aparecerão no .svg), modificar atributos de algumas formas (apenas os atributos finais aparecerão no .svg) ou criar novas formas (que aparecerão no arquivo .svg).

Como determinar se forma está (parcialmente) dentro da região de visibilidade

A região de visibilidade é um polígono que pode ser representado por uma sequência ordenada de vértices ou de segmentos. Uma visão esquemática dos casos de sobreposição é apresentada na Figura 3

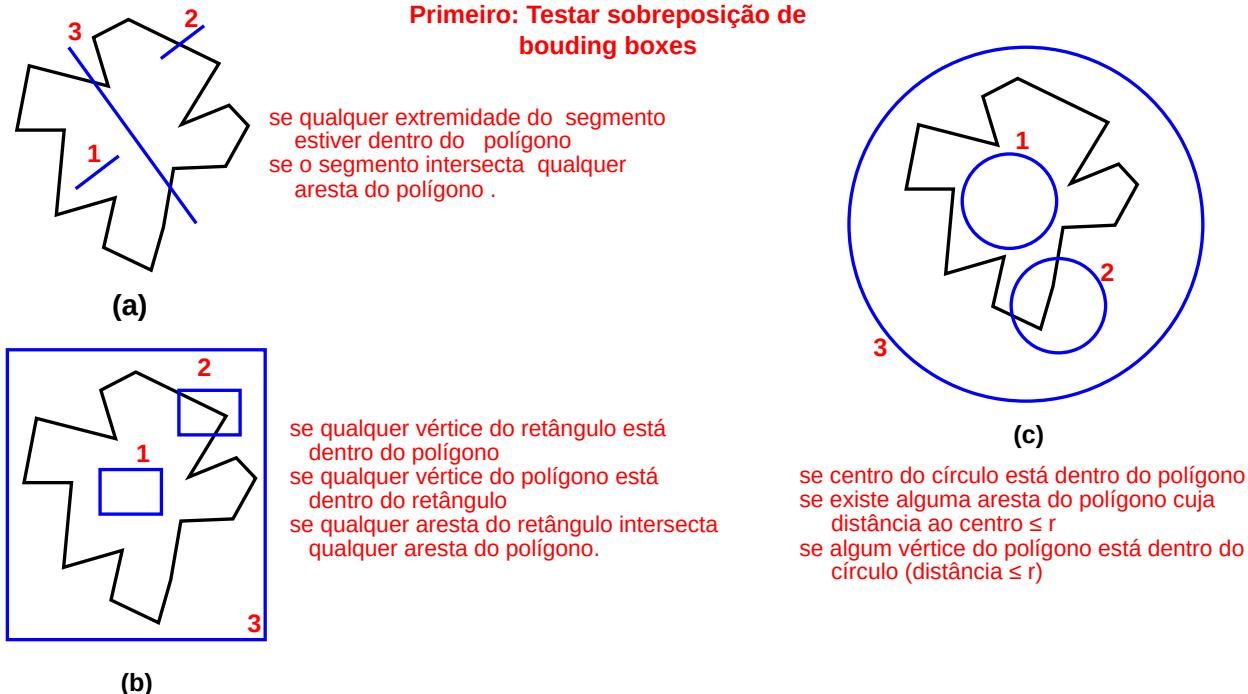


Figura 3: Casos de sobreposição

IMPLEMENTAÇÃO

Implementar dos TADs postados em nossa Sala de Aula.

É terminantemente proibido definir structs nos arquivos de cabeçalho (.h).

O programa deve estar bem modularizado (arquivos .h e .c). Cada estrutura de dados deve estar em um módulo separado. O arquivo .h deve estar muito bem documentado (lembre-se que é um “contrato”). Para cada módulo, escrever um programa que o teste (em um arquivo .c separado): teste unitário.

Usar uma árvore para os segmentos ativos. Usar listas para armazenar as formas.

O algoritmo de visibilidade utiliza algoritmo de ordenação. Ele deve estar preparado para usar o `qsort` (da biblioteca padrão do C) e uma versão modificada do mergesort (implementado por você). Quando subvetor for “pequeno” (ver parâmetro `-i`), deve ser usado o insertionsort.

AVALIAÇÃO

Espera-se uma atitude pró-ativa para a aquisição dos conhecimentos (i.e., estudo) para resolver o problema proposto.

A avaliação consistirá da execução dos testes e da inspeção de código. A nota será proporcional ao número de testes bem sucedidos, aplicados os descontos escritos abaixo.

ATENÇÃO: Caso algum tipo de FRAUDE seja detectada: Nota ZERO a TODOS os envolvidos.

Critério	Desconto
Definir struct em arquivo .h	2.5
Modularização Pobre: .h mal projetado, mal documentado	até 2.0
Não implementado conforme especificado	

<ul style="list-style-type: none">• não implementar estruturas pedidas (gravíssimo)	
Procedimentos extensos e/ou complicados	até 1.0
Escolha/uso de estrutura pouco eficiente	até 2.0
Poucos commits ou commits concentrados perto do final do prazo.	tipicamente até 3.0; porém, o padrão de commit pode indicar possível fraude.
Implementação ineficiente e/ou desidiosa.	Até 1.5.
Não usar o makefile provido	Em geral, nenhum desconto. Mas, se ocorrer problema de compilação ou execução que poderia ter sido evitado pelo uso do modelo do makefile provido, a consequência pode ser grave.
Erro de compilação	Nenhum teste será executado. Portanto, nota Zero, especialmente se for causado por negligência do aluno.

O Que Entregar

Os arquivos fontes devem estar em um repositório GIT. Submeter ao Classroom um arquivo-texto (.txt) com uma linha, com o formato abaixo. A URL fornecida será usada para clonar o repositório (git clone)

apelido url-do-repositorio
Exemplo: joseMane https://github.com/zemane/t2.git

O diretório clonado deve estar organizado como explicado na descrição geral.

RESUMO DOS PARÂMETROS DO PROGRAMA TED

Parâmetro / argumento	Opcional	Descrição
<code>-e <i>path</i></code>	S	Diretório-base de entrada (BED)
<code>-f <i>arq.geo</i></code>	N	Arquivo com a descrição da cidade. Este arquivo deve estar sob o diretório BED .
<code>-o <i>path</i></code>	N	Diretório-base de saída (BSD)
<code>-q <i>arqcons.qry</i></code>	S	Arquivo com consultas. Este arquivo deve estar sob o diretório BED .
<code>-to [q m]</code>	S	Tipo de ordenação a ser usado, q: qsort, m: mergesort. Default: q
<code>-i <i>n</i></code>	S	Default: 10

RESUMO DOS ARQUIVOS PRODUZIDOS

-f	-q	comando com sufixo	arquivos
<code>arq.geo</code>			<code>arq.svg</code>
<code>arq.geo</code>	<code>arqcons.qry</code>		<code>arq.svg</code> <code>arq-arqcons.svg</code> <code>arq-arqcons.txt</code>
<code>arq.geo</code>	<code>arqcons.qry</code>	<code>sufx</code>	<code>arq.svg</code> <code>arq-arqcons.svg</code> <code>arq-arqcons.txt</code> <code>arq-arqcons-sufx.[svg txt]³</code>

ATENÇÃO:

* os fontes devem ser compilados com a opção `-fstack-protector-all`.

* adotamos o padrão C99. Usar a opção `-std=c99`.

³ Podem ser produzidos os respectivos arquivos `.svg` e/ou `.txt`, dependendo da especificação do comando.