EXAME

A Entrada

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 esquerdo[1] 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.

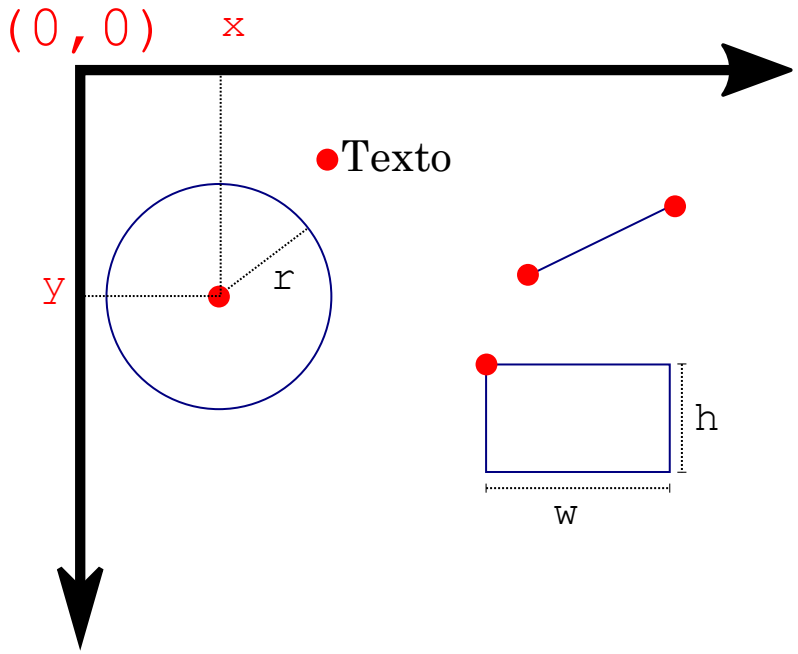


Ilustração 1: Formas no plano

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]

Alguns comandos utilizam memória auxiliar para armazenar identificadores

1 Note que o plano cartesiano está desenhado "de ponta-cabeça" em relação à representação usual.
2 http://www.december.com/html/spec/colorsvg.html.
https://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html

comando	parâmetros	descrição
c	`i x y r corb corp`	*desenhar círculo.* corb *é a cor da borda e* corp *é a cor do preenchimento*
r	`i x y w h corb corp`	*desenhar 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`	*Desenhar linha com extremidades nos pontos (x1,y1) e (x2,y2), com a cor especifiicada.*
t	`i x y corb corp a txto`	*desenha o texto* txto *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 txto é o último parâmetro do comando. Pode incluir espaços em branco e se estende até o final da linha.*
comandos .geo		

Algumas consultas fazem referência a filas e listas. O programa deve manter, pelo menos 1 fila para o polígono corrente e uma lista para as figuras selecionadadas.

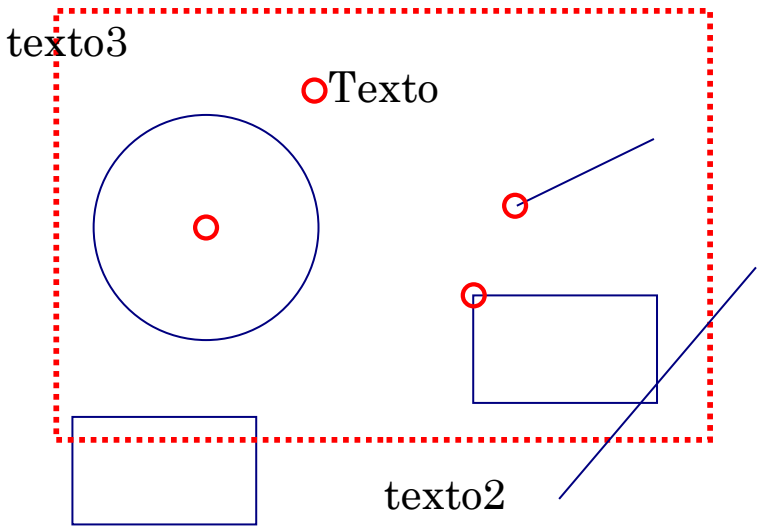
comando	parâmetros	descrição
inp	`i`	*Insere no polígono corrente as coordenadas da âncora da figura identificada por i. No caso de linhas, insere a extremidade de menor x (se empate, de menor y)* *TXT. Reportar a coordenada inserida e os dados da respectiva figura*
rmp		*Remove a coordenada mais antiga inserida no poligono corrente.* *TXT. Reportar a coordenada removida*
pol	`i d e corb corp`	*Produz um conjunto de linhas (e os insere no "banco de dados"). As linhas produzidas correspondem à borda do polígono e as linhas de seu preenchimento. Os identificadores das linhas produzidas são sequenciais, a partir de i. O parâmetro d é a distância entre as linhas do preenchimento; e é a espessura da linhas; corb a cor das linhas do polígono; corp, a cor das linhas do preenchimento.*
clp		*Remove todas as coordenadas do polígono corrente*
sel	`x y w h`	*Seleciona as figuras inteiramente dentro da região especificada pelos parâmetros. (Desconsidera seleções anteriores)* *SVG: desenhar região. Desenhar um anel vermelho em volta da âncora das figuras selecionadas.* *TXT: reportar identificador e tipo das figuras selecionada*
sel+	`x y w h`	*Semelhante à operação sel, porém, acrescenta (união) as figuras selecionadas àquelas anteriormente selecionadas.* ***ATENÇÃO****: as figuras selecionadas por este comando devem suceder (na lista) àquelas selecionadas por comandos anteriores.* *SVG: igual a sel.* *TXT: igual a sel. Também informar o total de figuras anteriormente selecionadas e o número total das figuras selecionadas por este comando*

comando	parâmetros	descrição
dels		*Remove do "banco de dados" todas as figuras selecionadas.* *SVG: A figuras removidas naturalmente não aparecerão no SVG (pois, não existem mais)* *TXT: Reportar o id e os dados das figuras removidas. Se o fator de degradação da árvore foi superado, reportar a árvore antes e depois da reconstrução (ver seção Implementação).*
dps	`i dx dy corb corp`	*Cria novas formas, semelhantes às selecionadas, porém, transladadas de dx e dy, cujos identificadores começam em i e são incrementados sequencialmente .*
ups	`corb corp dx dy n`	*Altera as cores e a posição de n figuras antes (se n<0) ou depois (se n>0) da última figura selecionada pelo último sel ou sel+. Atribui corb à borda das figuras e corp ao preenchimento. Translada a figura em dx unidades na horizontal e dy na vertical. Note que dx e dy podem ser valores negativos.*
Comandos .qry		

A Saída

A saída deve estar de acordo com a descrição dos comandos. Vale a pena ilustrar alguns detalhes destes comandos.

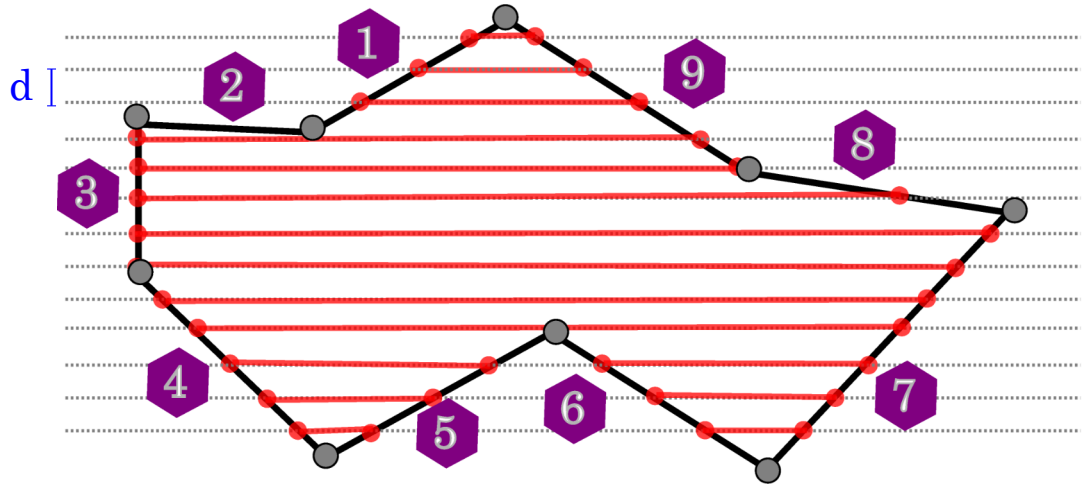
Os comandos **sel** e **sel+** devem mostrar a região de seleção e a âncora das figuras selecionadas.



A figura abaixo ilustra o efeito do comando **pol**. Os pontos cinzas correspondem às

coordenadas produzidas pelo comando **`inp`**. Os segmentos pretos são as bordas do polígono determinado pela ordem de execução de comandos **`inp`**. Estes segmentos devem ser acrescentados ao "banco de dados". Os segmentos vermelhos constituem o preenchimeto do polígono. Também devem ser inseridos no "banco de dados". Porém, os pontos vemelhos e as linhas tracejadas cinzas são apenas explicativas e não devem ser inseridas no "banco de dados".

O algoritmo para o preenchimento do polígono é muito simples. Ele faz uma linha horizontal (cinza tracejada na figura) "andar" sobre o polígono em saltos de distância **`d`**. As intersecções desta linha com segmentos das bordas do polígono (ponto vermelho) determinam dos extremos das linhas de preenchimento.



9 lados + 16 linhas preenchimento = 25 linhas inseridas

IMPLEMENTAÇÃO

As estruturas de dados lista e fila devem ser implementadas **conforme** o TAD mostrado em aula.

É **terminantemente proibido** declarar 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").

É **muito importante** que cada arquivo .h tenha no ínicio uma descrição geral do objetivo do módulo[3] e que cada procedimento seja bem explicado (o que faz? o que são os parâmetros? Alguma restrição sobre algum parâmetro?).

Muito **cuidado** com **procedimentos extensos** (mais de 20 linhas). Passível penalização na nota. Grande chance de bug! Provavelmente, você poderá quebrá-lo em vários procedimentos auxiliares.

Muito **cuidado** com a**ninhamentos de condicionais**: grande chance de bug, dificuldade de depuração!!

Enfatizo as duas últimas recomendações. Não segui-las muito provavelmente ocasionará perda de nota: ou por existência de bugs ou por desconto.

O "banco de dados" **deve** ser a árvore xyy do trabalho 2, **tal como especificada** no arquivo .h provido no classroom. Porém, deve ser feita uma modificação na implementação da árvore: a sub-árvore esquerda deve conter nós cuja coordenada y é estritamente menor que a coordenada y do nó. As outra duas sub-árvores contêm nós cuja coordenada y é maior ou igual à coordenada y do nó: uma dessas sub-árvores contém coordenadas x estritamente menores que a

3 Lembre-se: esta explicação inicial **deve** responder a pergunta: **"O que é uma instância disto?"**

coordenada x do nó; a outra, maiores ou iguais à do nó. Assim, podemos chamar esta variante de Árvore Yxx. (Veja figura abaixo)

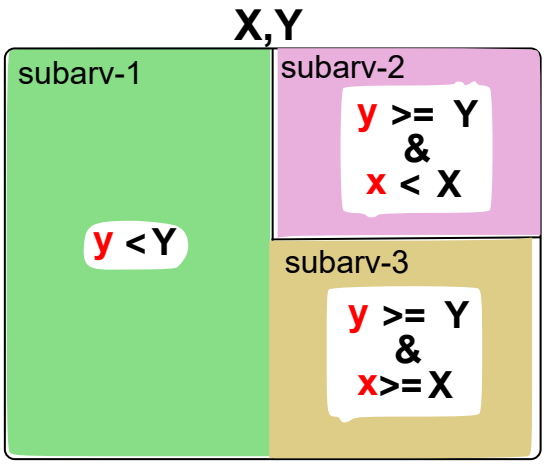


Figura 1:

Importante: em buscas, **não devem** ser exploradas sub-árvores que, com certeza, não conterão o dado procurado.

Importante: Sobre os nós removidos. Adotar a estratégia descrita no trabalho 2, porém:

- o vetor usado para reconstrução da árvore deverá ter capacidade de até **97** elementos.
- o vetor deve ser ordenado por meio do algoritmo qsort da biblioteca padrão do C.
- o fator de degradação será informado por um novo parâmetro do programa (-fd)

Ao final da execução do arquivo de consulta, uma representação da árvore deve ser impressa no arquivo txt correspondente.

- A árvore deve ser percorrida em largura.
- As informações de um nó devem ser impressas na mesma linha (ou seja, uma linha por nó)
- Nós no mesmo nível deve ter a mesma identação que deve ser crescente à medida da descida. Sugestão, a cada descida, aumentar a identação em 2 caracteres.

```
número de nós: …
número de nós removidos: …
fator de degradação: …
[id: 11] x: 100.00  y: 120.00  tipo: retângulo  cor: blue
  [id: 27] … dados desta forma
  [X][id: --] x: … y: … REMOVIDA
  [id: 23] … dados desta forma
    [id: 31] … dados desta forma
    [id: 32] … dados desta forma
    [id: 35] … dados desta forma
    [id: 33] … dados desta forma
    [X][id: --] x: … y: … REMOVIDA
    [id: 36] … dados desta forma
    [id: 38] … dados desta forma
      [id: 94] … dados desta forma
      [id: 84] … dados desta forma
      [id: 54] … dados desta forma
      [id: 64] … dados desta forma
      [id: 74] … dados desta forma
```

Para outros detalhes sobre a árvore, consultar a descrição do Trabalho 2.

Obs.: usar implementação dinâmica e encadeamento duplo para lista; implementação estática (**circular**) para as filas.

AVALIAÇÃO

A avaliação consistirá da execução dos testes e da inspeção de código. Além disso, deverá ser produzido um vídeo de, aproximadamente, 5 minutos no qual apresenta o programa. O aluno deve ter por objetivo convencer o avaliador que: (a) o programa funciona; (b)
o programa foi bem implementado O vídeo deve ser colocado no Youtube e o seu link deve estar anotado no arquivo leiame.

O Que Entregar

Submeter no Classroom o arquivo .zip com os fontes, conforme descrito anteriormente.

RESUMO DOS PARÂMETROS DO PROGRAMA TED

Parâmetro / argumento	Opcional	Descrição
`-e` ***`path`***	S	Diretório-base de entrada (**BED**)
`-f` ***`arq`*`.geo`**	N	Arquivo com a descrição da cidade. Este arquivo deve estar sob o diretório **BED**.
`-o` ***`path`***	N	Diretório-base de saída (**BSD**)
`-q` ***`arqcons`*`.qry`**	S	Arquivo com consultas. Este arquivo deve estar sob o diretório **BED**.
`-fd n`	S	Fator de degradação da árvore. Número entre 0.00 e 1.00. Valor default: 0.5.

RESUMO DOS ARQUIVOS PRODUZIDOS

-f	-q	comando com sufixo	arquivos
arq.geo			arq.svg
arq.geo	***arqcons***.qry		arq.svg arq-arqcons.svg arq-arqcons.txt
arq.geo	***arqcons***.qry	***sufx***	arq.svg arq-arqcons.svg arq-arqcons.txt arq-arqcons-sufx.[svg|txt][4]

ATENÇÃO:
* os fontes devem ser compilados com a opção `-fstack-protector-all`.
* adotamos o padrão C99. Usar a opção `-std=c99`.

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