PROJETO 1

ENTRADA DE DADOS

A entrada de dados, via de regra, ocorrerá por meio de um ou mais arquivos. Estes arquivos estarão sob um diretório, referenciado por **BED** neste texto.[1]

SAIDA DE DADOS

Os dados produzidos serão mostrados na saída padrão e/ou em diversos arquivos-texto. Alguns resultados serão gráficos no formato SVG. Os arquivos de saída serão colocados sob um diretório, referenciado por **BSD** neste texto.[2]

DESCRIÇÃO

A entrada do algoritmo será basicamente um conjunto de retângulos dispostos numa região do plano cartesiano e algumas consultas, por exemplo, que indagam se dois retângulos se sobrepõem. Os comandos estão contidos num arquivo `.geo` e as consultas num arquivo `.qry`.

Considere a Figura 1. Cada retângulo é definido por uma coordenada âncora (veja ponto roxo na figura) e por suas dimensões. A coordenada âncora do retângulo é seu canto inferior esquerdo[3] e suas dimensões são sua largura (w) e sua altura (h). Cada retângulo é identificado por um código alfa-numérico.

1 Indicado pela opção -e.
2 Indicado pela opção -o.
3 Note que o plano cartesiano está desenhado "de ponta-cabeça" em relação à representação usual.

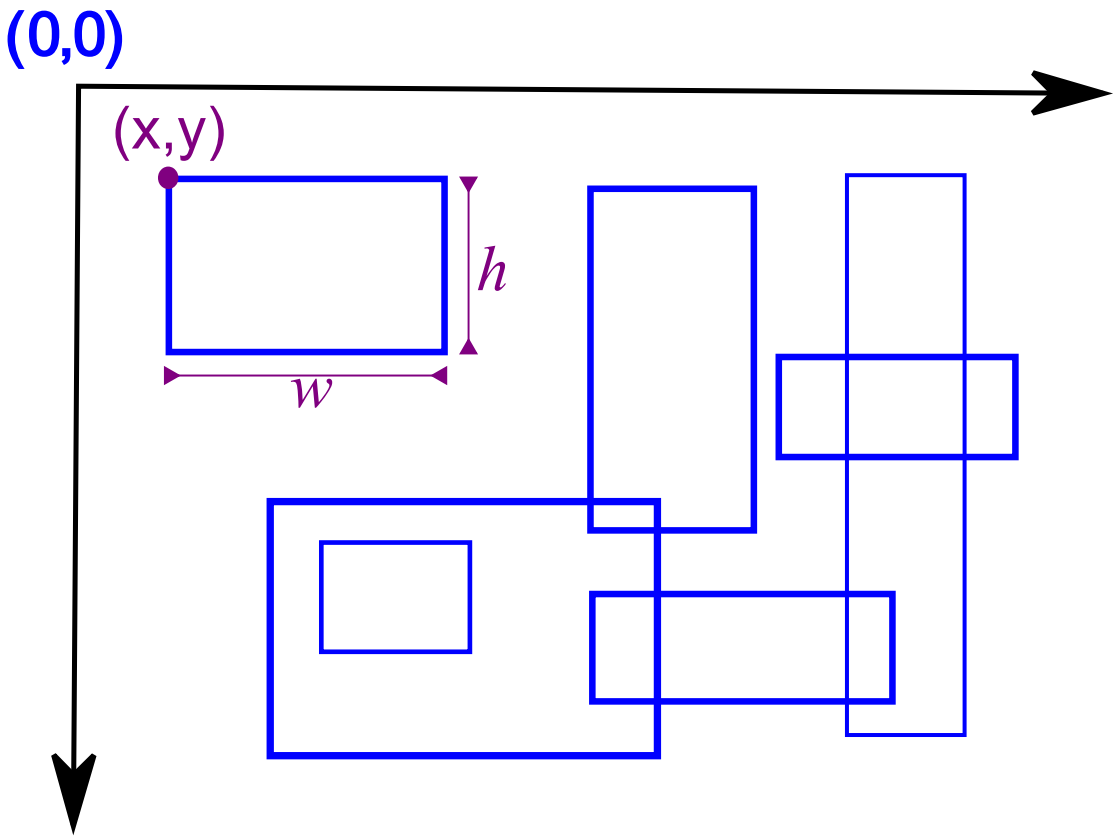


Figura 1: Retângulos no plano

As tabelas abaixo mostram os formatos dos arquivos de entrada (.geo e .qry). Os arquivos de entrada são compostos, basicamente, por conjunto de comandos (um por linha).

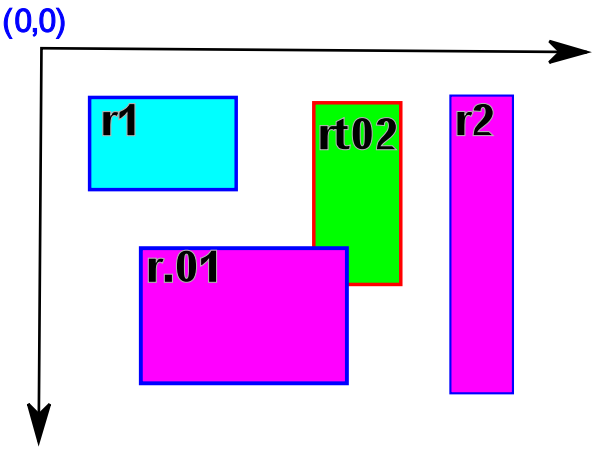
Cada comando tem um certo número de parâmetros, separados por um espaço. Os parâmetros mais comuns são:

- `id`: um identificador alfa-númerico. Por exemplo, r18, retang-0.1.2.3, etc.
- `w,h`: números reais. Dimensões do retângulo.
- `x,y`: números reais. Coordenada (x,y).
- `cor`: string. Cor válida dentro do padrão SVG.[4] Em lugar da cor, pode ser colocado a caractere @, indicando que não deve ser usada nenhuma cor.

comando	parâmetros	descrição
nx	n	*Número aproximado de retangulos*
cc	cor	*Cor para o contorno dos retângulo*
cp	cor	*Cor preenchimento dos retângulos*
r	id x y w h	*desenhar retângulo: w é a largura do retângulo e h, a altura.*
comandos .geo		

4 http://www.december.com/html/spec/colorsvg.html.
https://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html

Abaixo, um exemplo de um arquivo geo, com sua respectiva representação pictórica. Vamos supor que o nome do arquivo seja a1.geo.




```
cc  blue
cp  cyan
r  r1  21.09  24.035  37.3  23.7
cp  magenta
r  r.01  33.799  61.44  52.00 34.500
r  r2  111.00  23.70  16.000  74.5
cc  red
cp  lime
r  rt02  76.85  25.35  22.50  46.00
```

Arquivo geo

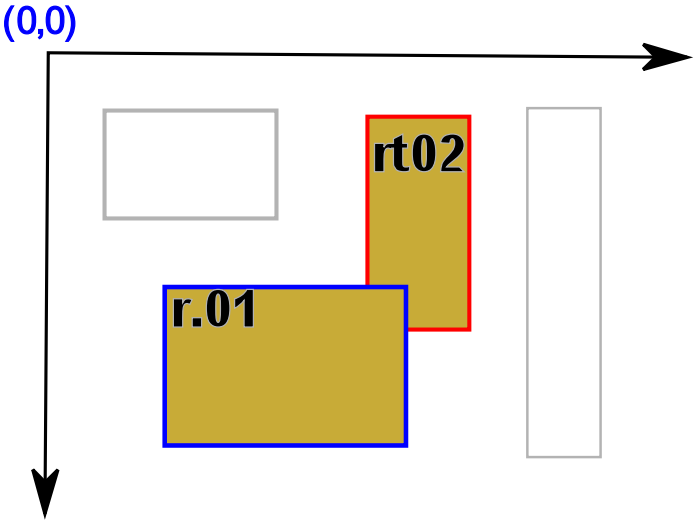
As consultas estão descritas na tabela abaixo. Note que muitas consultas são semelhantes. Espera-se que as partes comuns sejam fatoradas e reaproveitadas.

comando	parâmetros	descrição
tp		*Todos os pares de retângulos que se sobrepõem.* *TXT: escrever os identificadores dos retângulos (um par por linha)* *SVG: desenhar cada par de retângulos com a mesma cor de preenchimento (a sua escolha, usar pelo menos 10 cores diferentes). Só desenhar retângulos que tenham pelo menos uma sobreposição*
tpr	`x y w h`	*Semelhante à consulta tp, mas só considerar os retângulos que estejam inteiramente dentro da área delimitada pelo retângulo especificado nos parâmetros*
dpi	`x y`	*Remover todos os retângulos para os quais o ponto (x,y) é interno.* *TXT: reportar os identificadores dos retângulos removidos.* *SVG: retângulo removido não deve aparecer*

dr	id	Semelhante ao dpi, mas remove retângulos que estejam inteiramente dentro do retângulo de identificador id. Não remove o triângulo id.
bbi	x y	Desenhar o "bounding box" contendo todos os retângulos para os quais o ponto (x,y) é interno. TXT: identificadores dos retângulos selecionados e respectivas cores (preenchimento e contorno) originais. SVG: inverter as cores dos retângulos selecionados (preenchimento ↔ contorno). Desenhar o retângulo envolvente sem preenchimento com contorno vermelho grosso e tracejado. Desenhar o ponto (x,y) com cores de preenchimento e contorno contrastantes.
bbid	id	Semelhante ao bbi, porém selecionar todos os retângulos que estejam inteiramente contidos no retângulo com identificador id
iid	id k	Reporta os dados relativos ao retângulo identificado por id e os k elementos posteriores, ou anteriores, se k < 0. TXT: reportar id, âncora, largura, altura, cor de contorno, cor de preenchimento de cada um dos retângulos
diid	id k	Semelhante a iid, porém, além de reportar dados, remove os k elementos anteriores ou posteriores. **Não** remove o elemento id. TXT: como descrito em iid. SVG: elementos removidos não devem aparecer no svg.
comandos de consulta (.qry)		

Exemplo tp

A figura abaixo exemplifica a saída produzida pela consulta tp contida num arquivo .qry (o nome do arquivo poderia ser q1.qry) sobre o arquivo .geo (a1.geo, no nosso exemplo) . Note que são produzidos dois arquivos no diretório de saída (a1-q1.txt e a1-q1.svg, no nosso exemplo)



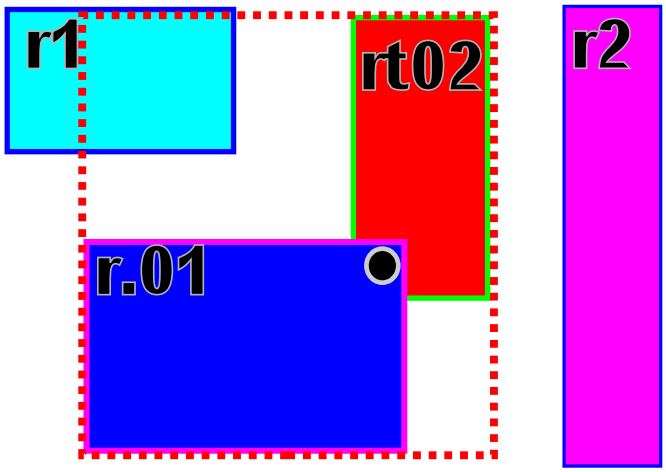

```
tp
```
Arquivo .qry

Arquivo .svg

Obs.: "perfis" em cinza claro mostram os retângulos que não devem aparecer, pois não há sobreposição.

```
tp
r.01 rt02
```
Arquivo .txt

Exemplo de bounding box




```
bbi 58.00  39.00
```
Arquivo de consulta

SVG

https://youtu.be/PQxtlY19kto

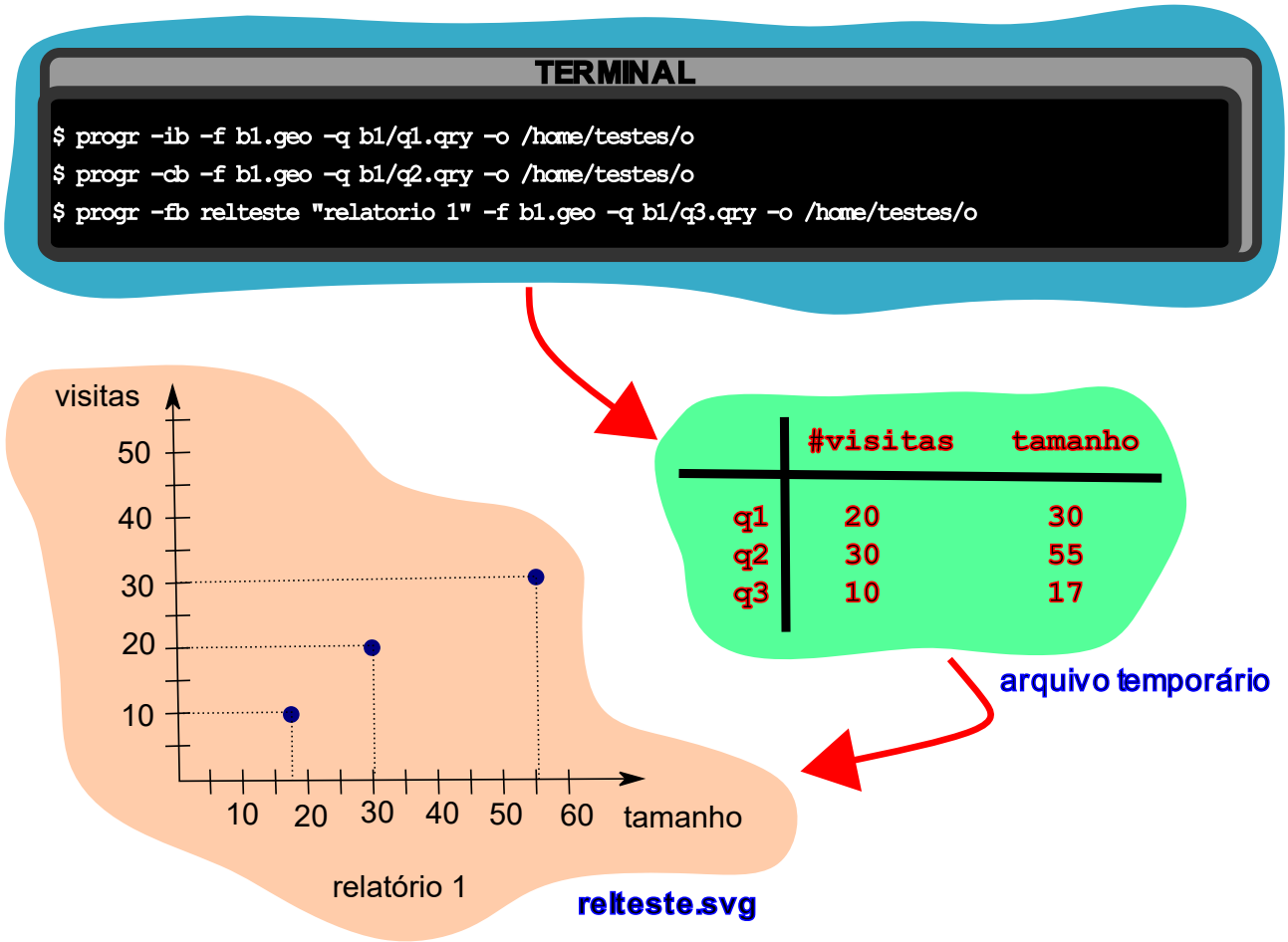
IMPLEMENTAÇÃO

- Os dados de entrada devem ser armazenados em listas
- O TAD Lista deve ser inteiramente implementado, i.e., todas as operações como descritas no TAD
- Devem ser feitas duas implementações da lista: estática simplesmente encadeada, dinâmica duplamente encadeada

- Novos retângulos (comando r) devem ser inseridos no final da lista. Assim, o retângulo do primeiro comando r processado estará no início da lista e o último retângulo estará no final da lista. (Note que isto é importante para as consultas iid e diid).
- Ambas implementações devem ser instrumentadas para registrar o desempenho da lista. Devem, para cada execução, contabilizar o tamanho da lista e o número de nós da lista visitados
- É **expressamente proibido** declarar structs em arquivos .h.

Relatório de desempenho

Caso solicitado, o programa deverá coletar dados para posterior análise de desempenho.



A figura acima mostra o processo de coleta de dados de execução para posterior confecção do relatório de desempenho. O objetivo é contabilizar, em cada execução, a quantidade de nós da lista visitados.

O programa reconhece 3 novos parâmetros, a saber:

- -ib: marca o início do processo de coleta de dados

- -cb: informa ao programa que a coleta deve continuar
- -fb arquivo-saida titulo: marca o final da coleta de dados. Os dados coletados devem ser processados. É produzido o arquivo arquivo-saida.svg (no diretório de saída) contendo um gráfico tamanho X #nós-visitados.

ORGANIZAÇÃO DA ENTREGA

O trabalho deve ser submetido no formato **ZIP**, cujo nome deve ser curto, mas suficiente para identificar o aluno ou a equipe.[5] Este arquivo deve estar organizado como descrito à frente.

PROCESSO DE COMPILAÇÃO E TESTES DO TRABALHO

Organização do ZIP a ser entregue

A organização do zip a ser entregue pelo aluno deve ser a seguinte:

[abreviatura-nome]	*Por exemplo, josers.*
LEIA-ME.txt	*colocar matrícula e o nome do aluno. Atenção: O número da matricula de estar no início da primeira linha do arquivo. Só colocar os números; não colocar qualquer pontuação.*
*	*Outros arquivos podem ser solicitados.*
/src	*(arquivos-fonte)*
makefile	*deve ter target para a geração do arquivo objeto de cada módulo e o target* **progr** *que produzirá o executável de mesmo nome dentro do mesmo diretório* **src**. *Os fontes devem ser compilados com a opção -fstack-protector-all.* ** adotamos o padrão C99. Usar a opção -std=c99.*
*.h e *.c	***Atenção:*** *não devem existir outros arquivos além dos arquivos fontes e do makefile*

Organização do diretório para a compilação e correção dos trabalhos (no computador do professor):

[HOME_DIR]	
*.py	*scripts para compilar e executar*
\t	*diretório contendo os arquivos de testes*
*.geo *.qry	*arquivos de consultas, em geral, distribuídos em alguns outros sub-diretórios. Por exemplo, as consultas relativas a um arquivo a1.geo, normalmente, estarão no subdiretório a1.*
\alunos	*(contém um diretório para cada aluno)*

5 Por exemplo, josers.zip (se aluno se chamar José Roberto da Silva), josers-mariabc.zip (para uma equipe com dois alunos. Evite usar maiúsculas, caracteres acentuados ou especiais.

`\abrnome` *diretório pela expansão do arquivo submetido (p.e., josers)*

outros subdiretórios para os arquivos de saída informados na opção `-o`

Os passos para correção serão os seguintes:

1. O arquivo .zip será descomprimido dentro do diretório alunos, conforme mostrado acima
2. O makefile provido pelo aluno será usado para compilar os módulos e produzir o executável. Os fontes serão compilados com o compilador gcc em um máquina virtual Linux. Os executáveis devem ser produzidos no mesmo diretório dos arquivos fontes O professor usará o GNU Make. Serão executadas (a partir dos scripts) o seguinte comando:
    - **make progr**
3. O programa será executado automaticamente várias vezes: uma vez para cada arquivo de testes e o resultado produzido será inspecionado visualmente pelo professor. Cada execução produzirá (pelo menos) um arquivo **.svg** diferente dentro do diretório informado na opção **-o**. Possivelmente serão produzidos outros arquivos .svg e .txt.

RESUMO DOS PARÂMETROS DO PROGRAMA

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**.
`-ib`	S	Inicia coleta de dados de desempenho
`-cb`	S	Continua a coletar dados
`-fb arq titulo`	S	Finaliza a coleta, produz no diretório **BED** o arquivo arq.svg contendo o gráfico com o título especificado
`-ldd`	S	(default) Usar lista duplamente encadeada com alocação dinâmica
`-lse`	S	Usar lista simplesmente encadeada com alocação estática

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][6]

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

APENDICE

https://www.gnu.org/software/make/manual/make.html
http://opensourceforu.com/2012/06/gnu-make-in-detail-for-beginners/

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