

## PROJETO 2020

O projeto a ser implementado durante o ano letivo é um sistema de informação geográfica simplificado, como definido abaixo.

*A geographic information system (GIS) is a computer system for capturing, storing, checking, and displaying data related to positions on Earth's surface. GIS can show many different kinds of data on one map. This enables people to more easily see, analyze, and understand patterns and relationships.*

**Fonte:** <http://www.nationalgeographic.org/encyclopedia/geographic-information-system-gis/>

O sistema será implementado incrementalmente e em fases. É importante enfatizar que em cada fase o sistema evolui, isto é, novas funcionalidades são acrescentadas, requisitos de implementação podem ser mudados, porém, as funcionalidades existentes **devem continuar funcionais**.

### LEMBRETE: AVALIAÇÃO (PROGRAMA DO CURSO)

A avaliação será feita por meio de provas (P) e de trabalhos individuais (Ti) e em equipe (Te). Serão atribuídas notas numéricas às avaliações (0-10). A avaliação poderá ser feita considerando apenas o artefato produzido (texto, programa, etc), como também considerando sua apresentação oral. Espera-se que no caso de trabalhos em equipe, todos os membros trabalhem assiduamente na execução do trabalho. Em caso de dúvidas sobre a efetiva participação de um membro, o professor poderá argui-lo durante a apresentação do trabalho ou convocar a equipe para uma entrevista. Caso o aluno não demonstre efetivo domínio do trabalho produzido, sua nota poderá ser diferente da dos outros membros da equipe. Muita atenção, neste caso, sua nota provavelmente será muito baixa (não excluída a nota nula).

O peso das atividades individuais (provas e trabalhos) será 2. O peso das atividades em equipe será 1. A última atividade do ano letivo terá peso maior, visando diminuir a possibilidade de o aluno ser aprovado sem entregar/realizar a última atividade do ano. O peso da última atividade será 6, seja ela individual ou em equipe.

O que é esperado do aluno:

1. frequência em todas as aulas e pontualidade. Isto é muito importante, pois as aulas são planejadas como uma sequência de atividades que incrementalmente farão os alunos adquirirem os conhecimentos e habilidades desejadas.
2. Participação efetiva nos trabalhos. Isto também é muito importante, pois muitas habilidades e conteúdos só serão adquiridos pela prática. A forma de organização da equipe deve possibilitar que todos os membros da equipe tenham o mesmo nível de aprendizado.
3. Estudo sério da parte teórica. Esta é uma disciplina de Ciência da Computação em uma Universidade. Espera-se que o aluno dedique algumas horas semanais para o aprofundamento da teoria exposta. Espera-se que o aluno estude a parte teórica antes de que inicie a implementação da parte prática.
4. Que o aluno não use apenas as transparências para estudar. O professor disponibilizará as transparências usadas apenas como forma de orientar os estudos. As transparências, propositalmente, contém apenas o esqueleto do conteúdo e figuras para facilitar a explicação do professor. Espera-se que o aluno faz uso efetivo dos livros listados na bibliografia.

**IMPORTANTE:** não será tolerado nenhum tipo de fraude nas provas ou nos trabalhos. Qualquer fraude detectada durante ou após sua realização será punida com nota zero a todos os envolvidos.

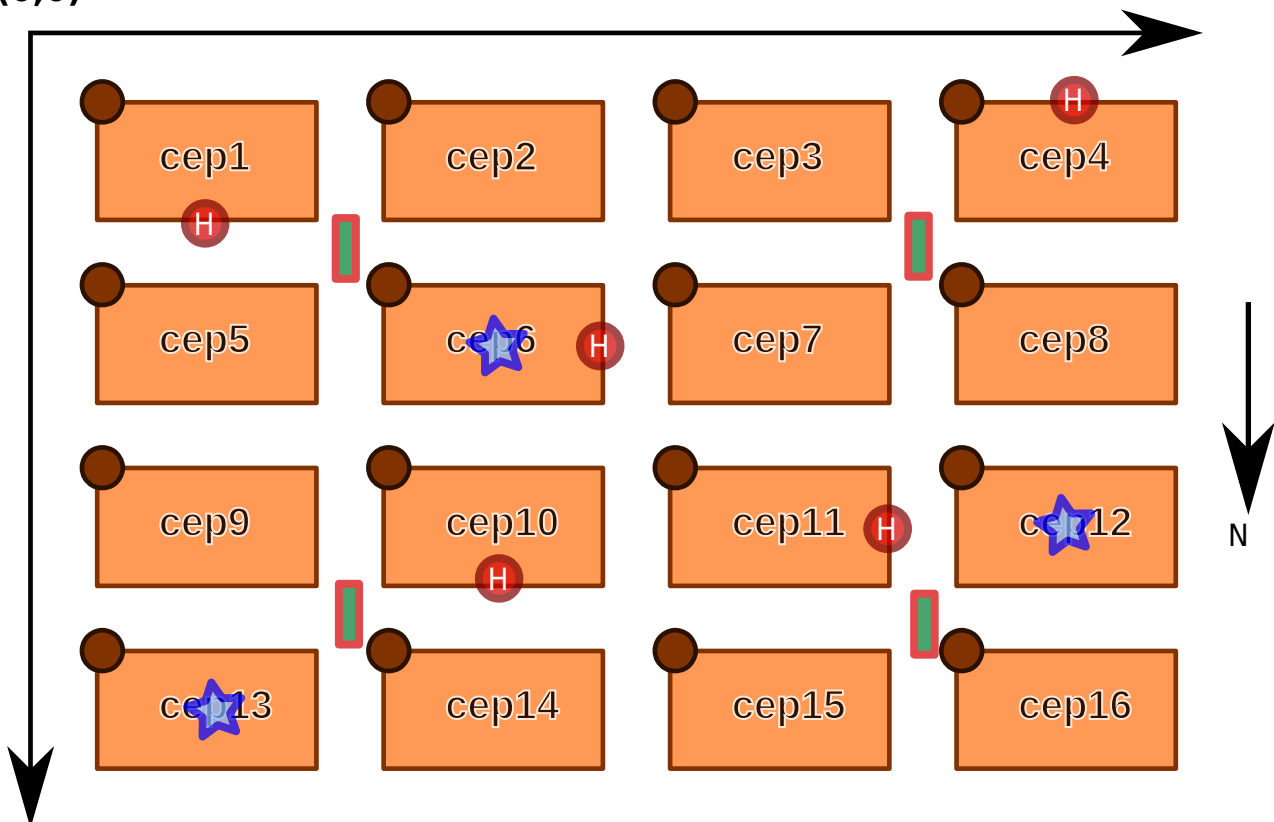
## DESCRIÇÃO

Um Sistema de Informações Geográficas (SIG), para a nossa finalidade, é um sistema que contém (não exclusivamente) dados geo-referenciados, isto é, dados com algum atributo de localização espacial (uma coordenada).

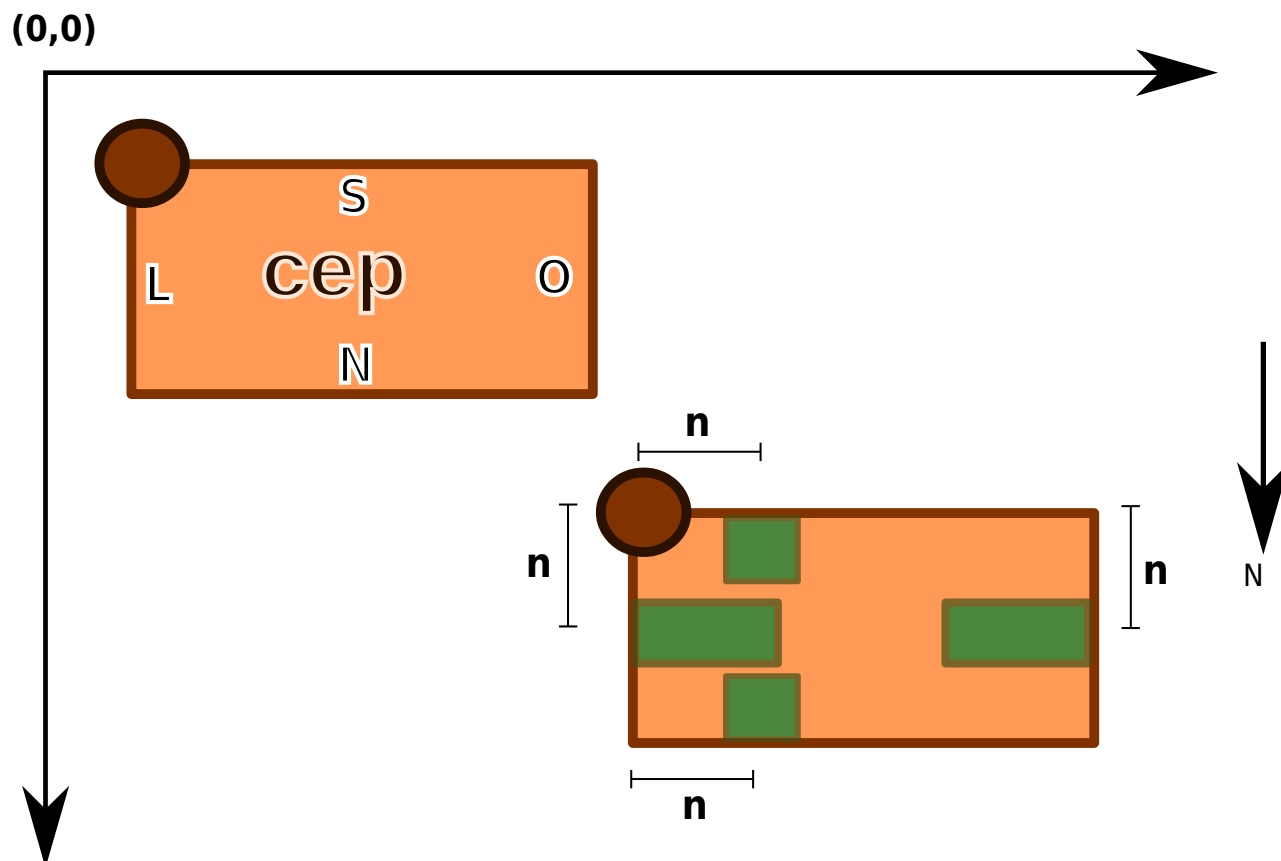
O sistema manipulará o mapa de uma cidade e algumas informações relacionadas.

O mapa de uma cidade é composto por um conjunto de retângulos que representam as quadras; e, um conjunto de equipamentos urbanos (hidrantes, semáforos, torres de celular, pontos de ônibus, etc). Cada equipamento urbano é localizado no mapa por um único ponto, conforme o exemplo abaixo.

(0,0)



A cidade exemplificada acima chama-se **Bitnópolis** e possui 16 quadras. O sistema de endereçamento de Bitnópolis é inspirado no de nossa capital federal. Cada **quadra** possui 4 **faces** (N,S,L,O) e é identificada por um **CEP** alfanumérico. O número de uma casa ou estabelecimento comercial é a **distância** da frente da casa até uma projeção do ponto de ancoragem do retângulo que representa a quadra (veja figura abaixo). Assim, um endereço é da forma CEP/Face/número, por exemplo, cep15/S/45. O ponto de ancoragem do retângulo é o canto sudeste da quadra.



## 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.<sup>1</sup>

## SAIDA DE DADOS

O 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.<sup>2</sup>

## 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.<sup>3</sup> 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:

<sup>1</sup> Indicado pela opção -e.

<sup>2</sup> Indicado pela opção -o.

<sup>3</sup> 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.

**[abreviatura-nome]**

LEIA-ME.txt

*Por exemplo, josers.**colocar matrícula e o nome do aluno. Atenção: O número da matrícula de estar no início da primeira linha do arquivo. Só colocar os números; não colocar qualquer pontuação.*

\*

*Outros arquivos podem ser solicitados a cada fase.*

/src

*(arquivos-fonte)*

makefile

*deve ter target para a geração do arquivo objeto de cada módulo e o target **siguel** 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, talvez, distribuídos em alguns outros sub-diretórios*

\alunos

*(contém um diretório para cada aluno)*

\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 siguel**
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.

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

# FASE I

## A Entrada

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

Considere a Ilustração 1. Cada forma geométrica é definida por uma coordenada âncora e por suas dimensões. A coordenada âncora do círculo é o seu centro e sua dimensão é definida por seu raio. A coordenada âncora do retângulo é seu canto inferior esquerdo<sup>4</sup> e suas dimensões são sua largura e sua altura. As coordenadas que posicionam as formas geométricas são valores reais e estão contidas dentro da região delimitada pelos cantos  $(0,0)$  e  $(x_{\max}, y_{\max})$ . Cada forma geométrica é indentificada por um número inteiro.

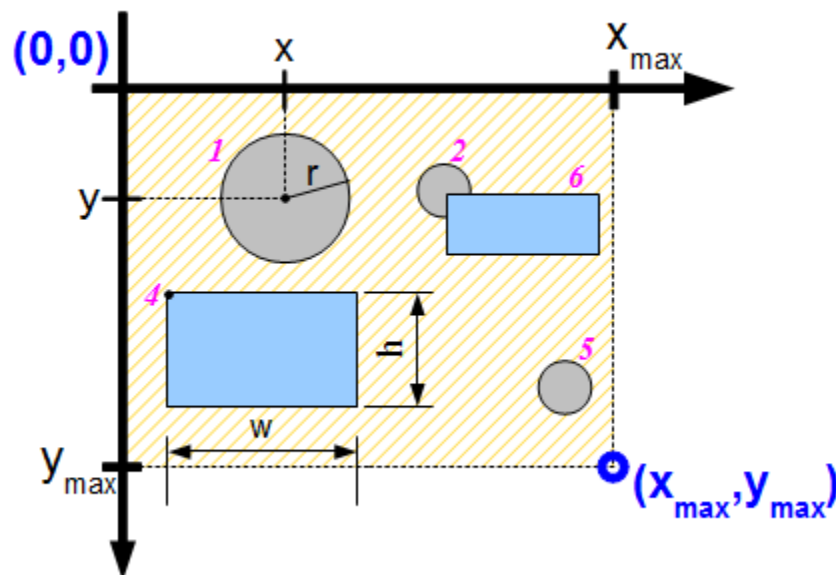


Ilustração 1

A tabelas abaixo mostram os formato dos arquivos de entrada (.geo e .qry). Os arquivos de entrada são compostos, basicamente, por conjunto de comandos (um por linha), a saber: **c** (desenhe um círculo), **r** (desenhe um retângulo), **t** (escreva um texto), etc.

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 criada pelos comandos **c** ou **r**.
- **r**: número real. Raio do círculo.
- **x, y**: números reais. Coordenada  $(x,y)$ .
- **cor**: string. Cor válida dentro do padrão SVG.<sup>5</sup>

<sup>4</sup> Note que o plano cartesiano está desenhado "de ponta-cabeça" em relação à representação usual.

<sup>5</sup> <http://www.december.com/html/spec/colorsvg.html>.  
<https://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html>

comando	parâmetros	descrição
<b>nx</b>	<i>i</i>	<i>Altera o número máximo de círculos e retângulos (i.e., <math>c + r + t</math>) criados no arquivo. O valor default é 1000.</i>
<b>c</b>	<i>i r x y corb corp</i>	<i>desenhar círculo. corb é a cor da borda e corp é a cor do preenchimento</i>
<b>r</b>	<i>i w h x y corb corp</i>	<i>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</i>
<b>t</b>	<i>i x y corb corp txto</i>	<i>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</i>
<b>comandos .geo</b>		

comando	parâmetros	descrição
<b>o?</b>	<i>j k</i>	<i>As formas geométricas cujos indentificadores são <b>j</b> e <b>k</b> se sobrepõem?<sup>6</sup> <b>Saída: .txt:</b> copiar o comando e, na linha seguinte escrever uma mensagem informando o tipo das formas j e k e se sobrepõe ou não <b>.svg:</b> traçar um retângulo que envolva ambas figuras: de bordas tracejadas, se não se sobrepõem; linha cheia, se se sobrepõem</i>
<b>i?</b>	<i>j x y</i>	<i>O ponto (x,y) é interno à j-ésima forma geométrica?<sup>7</sup> <b>Saída: .txt:</b> copiar o texto da consulta e, na linha seguinte, informar o tipo da forma geométrica e se é interno ou não. <b>.svg:</b> colocar um ponto (pequeno círculo) nas coordenadas (x,y) e pintá-lo de azul se for interno e magenta se for externo. Colocar uma linha (da mesma cor do ponto) ligando o ponto ao centro de massa da figura j.</i>

<sup>6</sup> A borda da figura pertence à figura. Assim, as figuras que coincidem apenas nas bordas também se sobrepõem.

<sup>7</sup> Um ponto na borda da figura pertence à figura, **mas não é interno** à figura.

comando	parâmetros	descrição
<b>pnt</b>	j corb corp	Mudar as cores da borda (corb) e do preenchimento (corp) da forma/texto de identificador j. <i>Saída: .txt:</i> copiar o texto da consulta e, na linha seguinte, informar as coordenadas originais da forma/texto. <i>.svg:</i> a figura afetada é mostrada com as novas cores
<b>pnt*</b>	j k corb corp	Semelhante ao comando pnt, aplicado às formas/texto entre os identificadores j e k, inclusive.
<b>delf</b>	j	Remove a forma/texto de idenficador j. <i>Saída: .txt:</i> copiar o texto da consulta e mostrar todas as informações sobre esta forma/texto. <i>.svg:</i> o elemento removido não deve aparecer no svg.
<b>delf*</b>	j k	Semelhante ao comando delf, aplicado às formas/texto entre os idenficadores j e k, inclusive.
comandos .qry		

A figura abaixo mostra um exemplo de um arquivo de entrada (consistente com a Ilustração 1). Note que a extensão do arquivo é **.geo**. As primeiras operações desenham círculos e retângulos.

```
c 1 50.00 50.0 30.00 grey magenta
r 6 121.0 46.0 100.0 30.0 cyan yellow
c 2 grey magenta 120.0 45.0 15.0
r 4 10.0 150.0 90.0 40.0 cyan yellow
c 5 230.0 180.0 13.0 grey magenta
```

**a01.geo**

```
o? 2 6
i? 5 210.0 160.0
```

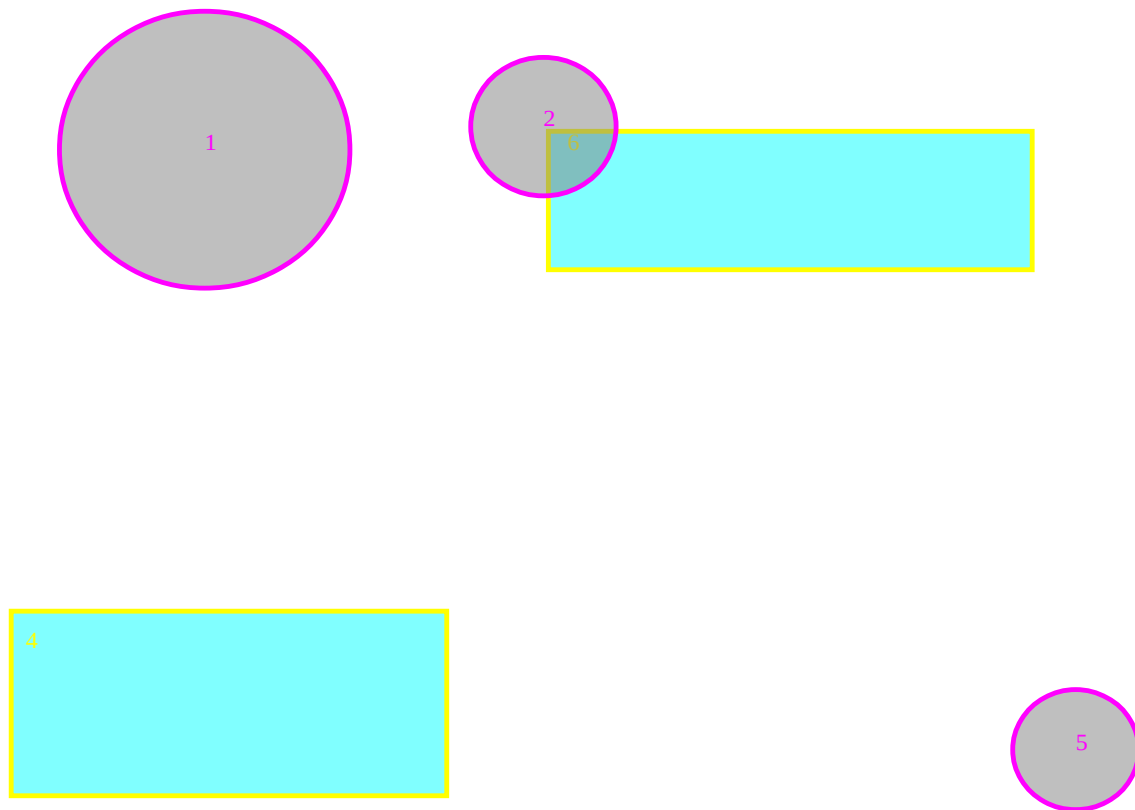
**q.qry**

## A Saída

O programa deverá produzir um arquivo **.svg** e um arquivo **.txt** ambos com o mesmo nome base do arquivo **.geo**.

O arquivo .svg produzido deve mostrar as formas geométricas. Além disso, para o comando

O, caso as figuras identificadas se sobreponham, elas devem ser envolvidas por um retângulo tracejado contendo a palavra "sobrepoe". Existem várias ferramentas que renderizam arquivos .svg. As figuras abaixo mostram um exemplo de arquivo .svg e sua respectiva renderização.



*Illustration 2: Arquivo a01.svg*

Se o arquivo .geo (a01.geo, no nosso exemplo) for processado em conjunto com um arquivo .qry (q.qry, no exemplo), além de a01.svg, deverá ser produzido o arquivo a01-q.svg, contendo os círculo, retângulos, etc acrescentados aos resultados da consulta.

Também produzir um arquivo-texto (a01-q.txt, no exemplo) contendo o resultado textual de todas as consultas. Neste arquivo deve ser copiado em uma linha o texto da consulta e, na linha seguinte, o seu resultado.

```
o? 2 6
2: retângulo 6: círculo SIM

i? 5 210.0 160.0
5: círculo NAO INTERNO
```

**Arquivo a01-q.txt**



## FASE II

Nesta fase serão acrescentados quadras e equipamento urbanos. A nossa cidade começa a tomar forma. Temos quadras, hidrantes, radio-bases de telefonia móvel e semáforos. As quadras são identificadas por seu CEP; os hidrantes, semáforos e rádio-bases são identificadas por um código alfa-numérico (por exemplo, *h-1234.ns\_abc23*).

As quadras ocupam uma região retangular, enquanto que os hidrantes, semáforos e rádio-bases são representadas por um ponto (coordenada) no mapa.

Nesta fase serão acrescentados novos comandos ao arquivo **.geo**:

comando	parâmetros	
<b>q</b>	cep x y w h	<i>Insere uma quadra (retângulo e cep)</i>
<b>h</b>	id x y	<i>Insere um hidrante</i>
<b>s</b>	id x y	<i>Insere um semáforo</i>
<b>rb</b>	id x y	<i>Insere uma rádio-base (torre de celular)</i>
<b>cq</b>	sw cfill cstrk	<i>Cores do preenchimento (cfill) e da borda (cstrk) das quadras, espessura da borda (sw) (a partir deste comando)</i>
<b>ch</b>	sw cfill cstrk	<i>Cores do preenchimento e da borda (e sua espessura) dos hidrantes (a partir deste comando)</i>
<b>cr</b>	sw cfill cstrk	<i>Cores do preenchimento e da borda (e sua espessura) das torres de celular (a partir deste comando)</i>
<b>cs</b>	sw cfill cstrk	<i>Cores do preenchimento e da borda (e sua espessura) dos semáforos (a partir deste comando)</i>
<b>sw</b>	cw rw	<i>Espessuras das bordas, respectivamente, dos círculos (comando c) e dos retângulos (comando r) (a partir deste comando)</i>
<b>nx</b>	i nq nh ns nr	<i>Informa o número máximo de formas(círculos e retângulos), quadras, hidrantes, semáforos e rádio-bases, respectivamente, criados no arquivo. O valor default é 1000.</i>
<b>Novos comandos do arquivo .geo</b>		

Alguns comandos de atualização e consulta podem ser colocados em um arquivo **.qry**:

comando	parâmetros	
<b>dq</b>	<b>[#] id r</b>	<p>remove todas quadras que estiverem inteiramente dentro a uma distância de no máximo <i>r</i> do equipamento urbano identificado por <i>id</i>. O parâmetro <i>#</i> pode estar ausente.</p> <p><b>No arquivo .svg:</b> Se o parâmetro <i>#</i> estiver ausente, as quadras removidas não devem aparecer; caso contrário, as quadras removidas devem ter cantos arredondados,<sup>8</sup> ser pintadas de “beige” (fundo) e “olive” (borda). O equipamento urbano em questão deve enfatizado com um anel grosso de duas cores. Desenhar o círculo de raio <i>r</i>, mostrando a região de delição.</p> <p><b>No arquivo .txt:</b> deve apresentar os ceps das quadras removidas, o <i>id</i> e respectivas informações do equipamento urbano.</p>
<b>del</b>	<b>( cep   id )</b>	<p>Remove a quadra, hidrante, semáforo ou torre de identificação <i>id</i> (ou <i>cep</i>).</p> <p><b>No arquivo .svg:</b> quadra ou equipamento urbano removido não deve aparecer. Deve ser colocada uma linha vertical com início no centro do elemento removido até o topo do mapa. Também colocar (no topo) ao lado da linha vertical o <i>cep</i> ou o <i>id</i>.</p> <p><b>No arquivo .txt:</b> reportar os dados relacionados da quadra ou equipamento urbano removido.</p>
<b>cbq</b>	<b>x y r cstrk</b>	<p>Muda a cor da borda para <i>cstrk</i> de todas as quadras que estiverem inteiramente contidas dentro do círculo de centro em (<i>x,y</i>) e de raio <i>r</i>.</p> <p><b>No arquivo .svg:</b> quadras eleitas pintadas conforme descrito.</p> <p>Reporta no <b>arquivo .txt</b> o <i>cep</i> das quadras que tiveram a cor da borda alterada</p>
<b>crd?</b>	<b>( cep   id )</b>	<p>Imprime no <b>arquivo .txt</b> as coordenadas e a espécie do equipamento urbano de um determinado <i>cep</i> ou com uma determinada identificação.</p>

8 Veja atributo *rx* de <rect>

comando	parâmetros	
<b>car</b>	x y w h	<p>Calcula a área total das quadras e equipamentos urbanos que estiverem inteiramente dentro do retângulo (x,y,w,h). No <b>arquivo .svg</b>: desenhar o retângulo da consulta e traçar uma linha vertical do ponto (x,y) até o topo do mapa. No topo do mapa, ao lado da linha vertical, escrever a área total. No centro de cada quadra selecionada, escrever a sua respectiva área. No <b>arquivo .txt</b>: escrever os ceps das quadras selecionadas e suas respectivas áreas. Escrever a área total.</p>
Comandos do arquivo .qry		

## EXEMPLOS

```

cq 2px blue black
ch 1px red yellow
ct 3px black red
q cep_001-10 37.00 15.00 89.00 40.00
q cep_001-20 137.00 15.00 89.00 40.00
q cep_001-30 237.00 15.00 89.00 40.00
cq 1.5px yellow green
q cep_002-10 37.00 115.00 89.00 40.00
q cep_002-20 137.00 115.00 89.00 40.00
q cep_002-30 237.00 115.00 89.00 40.00
h h-12 320.00 60.00
c 3 29.00 720.00 458.00 green yellow
r 4 5.00 29.00 1049.00 479.00 green blue
sw 3.4px 1.3px
r 5 55.00 42.00 215.00 702.00 green red
c 6 51.00 139.00 635.00 chocolate black

```

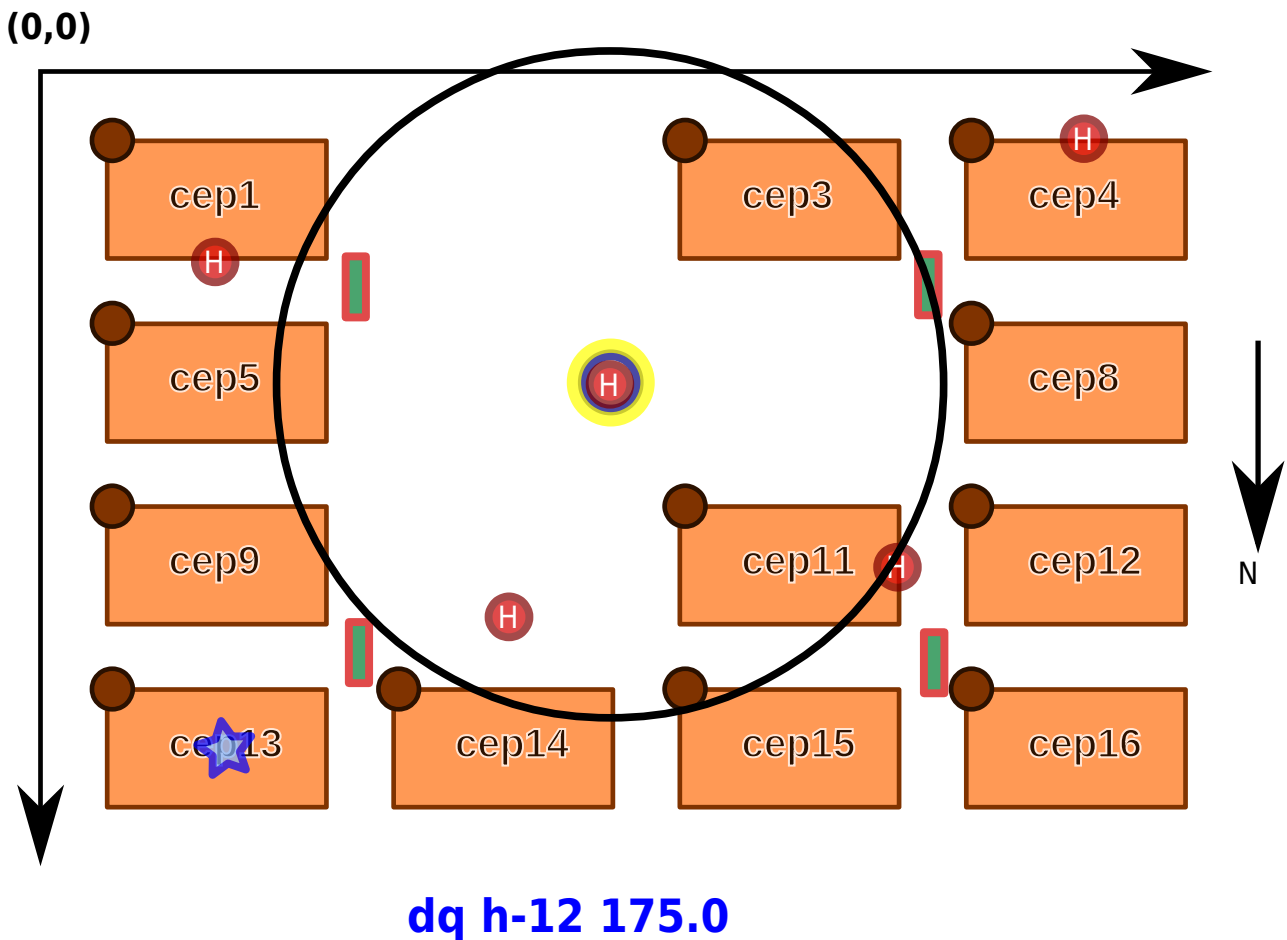
**a1.geo**

```

dq h-12 120.0
crd? cep_001-10
crd? h-12

```

**q1.qry**



### A SAIDA

O processamento dos arquivos .geo e .gry devem continuar produzindo as mesmas saídas da fase anterior (com as modificações apresentadas nesta fase). Os novos comandos e consultas devem produzir as saídas como descrito.

### A IMPLEMENTAÇÃO

O TAD Listas mostrado em sala de aula deve ser **completamente** implementado. A implementação da lista deve ser a dinâmica duplamente encadeada. As quadras e equipamentos urbanos devem ser armazenados em listas diferentes (ou seja, são 4 listas diferentes). Outros TADs devem ser projetados.

Todos os TADs devem ser implementados como módulos, conforme descrito em sala de aula (.h, .c). É **expressamente proibida** a declaração de structs em arquivo .h. O arquivo .h deve ser escrito antes do respectivo arquivo .c. O arquivo .h deve estar bem comentado (lembrar que este arquivo é um contrato).

## O Programa

O nome do programa deve ser **siguel** e aceitar quatro parâmetros:<sup>9</sup>

**siguel** [-e *path*] -f *arq.geo* [-q *consulta.qry*] -o *dir*

O primeiro parâmetro (-e) indica o diretório base de entrada. É opcional. Caso não seja informado, o diretório de entrada é o diretório corrente da aplicação. O segundo parâmetro (-f) especifica o nome do arquivo de entrada que deve ser encontrado sob o diretório informado pelo primeiro parâmetro. O terceiro parâmetro (-q) é um arquivo de consultas. O último parâmetro (-o) indica o diretório onde os arquivos de saída (*\*.svg* e *\*.txt*) deve ser colocados. Note que o nome do arquivo pode ser precedido por um caminho relativo; *dir* e *path* é um caminho absoluto ou relativo (ao diretório corrente).

A seguir, alguns exemplos de possíveis invocações de **siguel**:

- `siguel -e /home/ed/testes/ -f t001.geo -o /home/ed/alunos/aluno1/o/`
- `siguel -e /home/ed -f ts/t001.geo -o /home/ed/alunos/al1/o`
- `siguel -f ./tsts/t001.geo -e /home/ed -o /home/ed/alunos/aluno1/o/`
- `siguel -o ./alunos/aluno1/o -f ./testes/t001.geo`
- `siguel -o ./alunos/aluno1/o -f ./testes/t001.geo -q ./t001/q1.qry`
- `siguel -e ./testes -f t001.geo -o ./alunos/aluno1/o/ -q ./q1.qry`

---

9 Novos parâmetros são acrescentados no decorrer do ano

## A Avaliação

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

A avaliação se baseará em dois critérios: **(a)** inspeção do código-fonte (especial atenção na especificação); **(b)** compilação e testes do executável.

## O Que Entregar

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

## RESUMO DOS PARÂMETROS DO PROGRAMA SIGUEL

Parâmetro / argumento	Opcional	Descrição
-e <i>path</i>	S	Diretório-base de entrada ( <b>BED</b> )
-f <i>arq.geo</i>	N	Arquivo com a descrição da cidade. Este arquivo deve estar sob o diretório <b>BED</b> .
-o <i>path</i>	N	Diretório-base de saída ( <b>BSD</b> )
-q <i>arqcons.qry</i>	S	Arquivo com consultas. Este arquivo deve estar sob o diretório <b>BED</b> .

## RESUMO DOS ARQUIVOS PRODUZIDOS

-f	-q	comando com sufixo	arquivos
<i>arq.geo</i>			arq.svg
<i>arq.geo</i>	<i>arqcons.qry</i>		arq.svg arq-arqcons.svg arq-arqcons.txt
<i>arq.geo</i>	<i>arqcons.qry</i>	<i>sufx</i>	arq.svg arq-arqcons.svg arq-arqcons.txt arq-arqcons-sufx.[svg txt] <sup>10</sup>

**ATENÇÃO:**

\* o fontes devem ser compilados com a opção -fstack-protector-all.

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

<sup>10</sup> Podem ser produzidos o respectivo arquivos .svg e/ou .txt, dependendo da especificação do comando.