ÁLGEBRA LINEAR ALGORÍTMICA–PLE–LABORATÓRIO 2

1. Neste laboratório investigaremos como utilizar operadores lineares para transformar quadrados de lado um de maneira a produzir figuras variadas. Faremos isto criando uma animação que, começando do quadrado Q de vértices

$$(0,0),(1,0),(0,1) \text{ e } (1,1)$$

e utilizando uma série de operadores lineares e translações, gera uma sequência de letras.

2. Começamos listando as funções do MAXIMA que você precisará usar.

Função	Efeito
`l[k]`	retorna o k-ésimo elemento da lista `l`
`max(x1,..., xn)`	retorna o máximo entre os argumentos `x1,..., xn`
`min(x1,..., xn)`	retorna o mínimo entre os argumentos `x1,..., xn`
`polygon(lx,ly)`	desenha o polígono cujos vértices têm abscissas em `lx` e ordenadas em `ly`
`length(l)`	retorna o tamanho da lista `l`
`cons(a,l)`	acrescenta o elemento `a` à lista `l`
`reverse(l)`	reordena uma lista de trás para a frente

Por fim, se $\texttt{exp}_\texttt{k}$ é uma expressão que depende de um parâmetro k,

`makelist(`$\texttt{exp}_\texttt{k}$`,k,`$\texttt{k}_\texttt{i}$`,`$\texttt{k}_\texttt{f}$`);`

gera a lista de elementos $\texttt{exp}_\texttt{k}$, com k variando de $\texttt{k}_\texttt{i}$ a $\texttt{k}_\texttt{f}$. Por exemplo,

`makelist(`$\texttt{k}^2$`,k,1,10);`

gera a lista dos quadrados de números inteiro entre 1 e 10.

⚠ Algumas observações importantes:

1. `cons(a,l)` acrescenta um novo elemento `a` no início da lista `l`, *mas não atribui esta lista à variável* `l`. Se o que você quer é uma lista, que continua sendo chamada de `l`, mas que agora é encabeçada por um novo elemento `a`, você tem que atribuir `cons(a,l)` a `l`, fazendo `l:cons(a,l)`.

2. É necessário um pouco de cuidado no uso da função `polygon(lx,ly)`. Em primeiro lugar, a j-ésima posição nas listas `lx` e `ly` deve conter a abscissa e a ordenada *de um mesmo vértice* do polígono. Em segundo lugar, os vértices devem estar listados em `lx` e `ly` na ordem em que os pontos seriam ligados se estivéssemos desenhado o polígono *sem tirar o lápis do papel.* Por exemplo, `polygon(lx,ly)` gera um quadrado se `lx:[0,1,1,0]` e `ly:[0,0,1,1]`, mas não se `lx:[0,1,1,0]` e `ly:[0,1,0,1]`. Vale a pena experimentar este segundo caso para ver o que acontece.
3. Antes de utilizar as funções que fazem desenhos é necessário carregar a biblioteca `draw`.

3. A primeira função a ser implementada, que chamaremos de `retangulo`, tem como entradas um operador linear, representado por sua matriz M, e uma lista que contém as coordenadas de um vetor u. A saída do programa será o desenho do quadrilátero obtido transformando o quadrado Q por M e transladando o resultado usando u. Em outras palavras, o ponto do quadrado correspondente à extremidade de um vetor w do plano será desenhado como a extremidade do vetor $u + Mw$.

À primeira vista pode parecer que precisamos de um `for` para calcular $u + Mw$ para cada vértice w, mas na verdade isto não é necessário. Basta criar uma matriz `quad` cujas colunas são os vértices de Q. Os vértices do quadrilátero obtido aplicando M a Q são as colunas de M.`quad`. Reciclaremos esta ideia muitas vezes ao longo do curso, de maneira que é essencial você se convencer de que funciona corretamente antes de continuar a fazer o programa.

O esqueleto do código da função `retangulo` é dado a seguir.

```
retangulo(M,u):=
block(
    lista das variáveis locais à função,
    crie uma matriz quad cujas colunas são os vértices do quadrado Q,
    crie uma matriz v cujas 4 colunas são iguais a u,
    quad:M.quad+v,
    crie a list lx das abscissas dos pontos de quad,
    crie a list ly das ordenadas dos pontos de quad,
    calcule o máximo maxh dos elementos de lx,
    calcule o mínimo minh dos elementos de lx,
    calcule o máximo maxv dos elementos de ly,
```

```
   calcule o mínimo minv dos elementos de ly,
   retangulo:polygon(lx,ly),
   return([retangulo,[minh,maxh],[minv,maxv]])
)$
```

⚠ Precisamos retornar o máximo e o mínimo das ordenadas e abscissas dos vértices do retângulo para podermos ajustar sua imagem ao tamanho da tela de saída.

4. A segunda função a ser implementada, que chamaremos de `letra`, tem como entradas uma lista `lm` de operadores lineares, representados por suas matrizes, e uma lista `lu` de vetores do plano. A saída do programa será o desenho de uma letra formada por quadriláteros obtidos aplicando a função `retangulo` à matriz `lm[k]` e ao vetor `lu[k]` para k variando entre 1 e o tamanho de `lm`.

⚠ Algumas observações importantes sobre o código da função `letra`:

- as listas `lm` e `lu` têm que ter a mesma quantidade de entradas;
- `minf` e `inf` são os símbolos para $-\infty$ e $+\infty$ no MAXIMA ;
- na lista `lret` serão postos os polígonos retornados pela função `retangulo`;
- a lista vazia é `[]`;
- o k-ésimo elemento da lista `l` é `l[k]`.

Um esboço do código da função `letra` é apresentado a seguir.

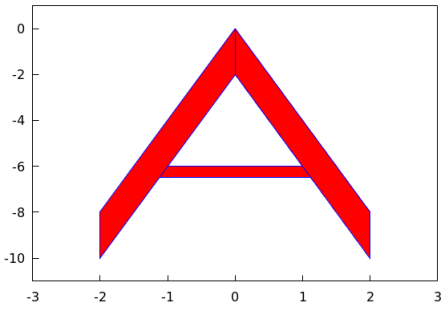
```
letra(lm,lu):=
block(
   liste as variáveis locais à função,
   crie uma lista vazia lret,
   minh:inf,
   maxh:minf,
   minv:inf,
   maxv:minf,
   for k:1 thru length(lm) do
   (
     ret:retangulo(lm[k],lu[k]),
     minh:min(minh,ret[2][1]),
     maxh:max(maxh,ret[2][2]),
     minv:min(minv,ret[3][1]),
     maxv:max(maxv,ret[3][2]),
```

```
      acrescente o polígono ret[1] à lista lret,
      ),
    draw2d(
      xrange=[minh-1,maxh+1],
      yrange=[minv-1,maxv+1],
      lret)
)$
```

5. Invente uma lista de matrizes `lm` e de vetores `lu` que, quando tomados como entrada da função `letra` fazem o programa retornar as letras maiúsculas:

I, L, F, H, V, Y, W e S em caracteres romanos e I, Y em itálico.

Algumas letras requerem apenas uma matriz e um vetor, algumas requerem várias matrizes e vetores. Por exemplo, para gerar o I romano basta esticar o quadrado ao longo da vertical, mas para gerar um V romano é necessário gerar um I itálico e refleti-lo em torno do eixo vertical. Minha versão da letra A aparece abaixo:



Note que, para posicionar a barra horizontal do A é necessário transladar a barra da origem. É para dar conta desse tipo de problema que são usados os vetores da lista `lu`.

6. Finalmente, vamos animar os desenhos para que possamos ver as letras aparecendo sucessivamente na tela gráfica do MAXIMA . Para ver as letras uma após a outra basta pôr os códigos que definem cada letra, um após o outro, na ordem em que você quer que apareçam na tela. Antes de começar, clique com o botão direito do mouse na margem superior da tela gráfica (onde aparece escrito gnuplot) e escolha *Always on top*, para que a tela sempre fique acima do console do MAXIMA. Assim você consegue ver cada letra à medida que é gerada pelo computador ou celular. O principal problema é a extrema velocidade com que o MAXIMA gera as letras. Contornei este problema pondo o programa para executar algo demorado entre uma letra e outra. No caso do meu computador, usei

```
for j:1 thru 1500 do (factorial(j));
```

para obrigar o MAXIMA a calcular 1500 fatoriais, que têm um custo razoavelmente alto. Dependendo da velocidade de processamento do computador (ou celular) você pode precisar aumentar ou diminuir a quantidade de fatoriais calculados. Sugestões mais inteligentes para contornar este problema são muito bem-vindas.