A.3. Usando Maple

O Maple é um sistema de álgebra computacional comercial de uso genérico. Constitui um ambiente informático para a computação de expressões algébricas, simbólicas, permitindo o desenho de gráficos a duas ou a três dimensões. O seu desenvolvimento começou em 1981 pelo Grupo de Computação Simbólica na Universidade de Waterloo em Waterloo, no Canadá, província de Ontário. A versão actual é Maple 12.0.

Estas notas, baseadas no respectivo manual de instruções, explicam como podemos começar a trabalhar com o programa.

1. Começando: cálculo numérico

```
> restart;
```

```
> 1-5*(3-1/2);
```

$$\frac{-23}{2}$$

Para fazer cálculos sem mostrar o resultado, usa-se ":" em vez de ";":

```
> A:=1-2/3: B:=1/3:
```

```
> A+B;
```

$$\frac{2}{3}$$

O símbolo (%) refere o valor obtido no último cálculo.

```
> 1/3+%;
```

$$1$$

```
> 3/5-%;
```

$$\frac{-2}{5}$$

Para obter o resultado na forma decimal, usa-se a função **evalf**

```
> evalf(%);
```

$$-.4000000000$$

Pode especificar-se o número de dígitos:

```
> evalf(1/3,20);
```

$$.33333333333333333333$$

```
> evalf(sqrt(2));
```

$$1.414213562$$

Algumas constantes:

```
> Pi; evalf(Pi);
```

$$\pi$$
$$3.141592654$$

```
> exp(1); evalf(exp(1));
```

$$\mathbf{e}$$
$$2.718281828$$

```
> I^2;
```

$$-1$$

2. Mais alguma informação básica sobre o Maple

O valor de uma expressão pode ser dada por meio de uma fracção:

```
> (121/14-3^2)*11^2;
```

$$\frac{-605}{14}$$

Se pretender o resultado na forma decimal, basta usar **evalf**:

```
> evalf(%);
```

$$-43.21428571$$

Se pretende uma aproximação com um certo número de dígitos:

```
> evalf(%,4);
```

$$-43.21$$

O número π em Maple:

```
> Pi;
```

$$\pi$$

Aproximações com 6 e 20 dígitos, respectivamente:

```
> evalf(Pi,6); evalf(Pi,20);
```

3.14159
3.14159265358979323 85

Outro exemplo:

```
> sqrt(3);
```

$$\sqrt{3}$$

```
> evalf(sqrt(3),4);
```

1.732

Funções básicas:

```
> exp(x);
```

$$e^x$$

```
> exp(2);
```

$$e^2$$

```
> sin(Pi/2);
```

1

Somatórios(exemplos):

```
> sum(k*k, k=0..3);
```

14

```
> sum(2*k-1,k=1..50);
```

2500

```
> seq(sum(k,k=1..n), n=1..30);
```

1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66, 78, 91, 105, 120, 136, 153, 171, 190, 210, 231, 253, 276, 300, 325, 351, 378, 406, 435, 465

```
> sum(2*k-1,k=1..50);
```

2500

```
> sum(i,i=1..n);
```

$$\frac{(n+1)^2}{2} - \frac{n}{2} - \frac{1}{2}$$

3. Polinómios

```
> (y+x*(y+z))^3;
```

$$(y + x(y + z))^3$$

```
> expand(%);
```

$$y^3 + 3xy^3 + 3y^2xz + 3x^2y^3 + 6x^2y^2z + 3yx^2z^2 + x^3y^3 + 3x^3y^2z + 3x^3yz^2 + x^3z^3$$

```
> collect(%,x);
```

$$(y^3 + 3y^2z + 3yz^2 + z^3)x^3 + (3y^3 + 6y^2z + 3yz^2)x^2 + (3y^3 + +3y^2z)x + y^3$$

```
> factor(%);
```

$$(y + xy + xz)^3$$

Exercício 1. Factorize e indique as raízes do polinómio

$$x^4 + x^3 - 3x^2 - x + 2$$

Um exemplo sobre juros. Pretende-se um modelo para calcular o capital acumulado a partir de uma determinada quantia de dinheiro, sabendo que o juro é de 10% ao ano. Começando com um exemplo, suponhamos que a quantia inicial é de 1000 euros.

```
> p(0)=1000;
```

$$p(0) = 1000$$

Para nos podermos referir a esta equação mais tarde vamos dar-lhe um nome, por exemplo, eq0.

```
> eq0:=%;
```

$$p(0) = 1000$$

Vamos atribuir também um nome à equação $j = 0.1$. (Note-se que $p(0) + jp(0)$ é a quantia obtida no final do primeiro ano de depósito.)

```
> eq1:=(j=.1);
```

$$eq1 := j = .1$$

Denotando por $p(1)$ o total do depósito no final do primeiro ano, temos a equação:

```
> p(1)=p(0)+j*p(0);
```

$$p(1) = p(0) + jp(0)$$

Se quisermos saber o valor de p(1) para os valores que assumimos no início, basta usar as equações eq0 e eq1 para especificar esses valores, com o comando subs. A seguir ilustra-se a sequência dos cálculos:

A rotina **subs** permite obter valores para uma expressão envolvendo variáveis pondo

subs(variável1=certa coisa1, variável2=certa coisa2, ..., variáveln=certa coisan, expressão)

Usando esta rotina:

```
> subs(eq0,eq1,%);
```

$$p(1) = 1100.0$$

Exercício 2. Use $p(1)$ para determinar o capital em depósito após o primeiro ano, supondo que o capital inicial foi de 2500 euros e o juro anual é de 6%.

Claro que agora nos interessa uma fórmula que permita calcular o capital ao fim de cada n anos. O capital no final do ano $n+1$ pode exprimir-se em função do capital no final do ano n por meio da igualdade:

```
> p(n+1)=p(n)+i*p(n);
```

$$p(n+1) = p(n) + ip(n)$$

Uma equação deste tipo é dita de recorrência. um cálculo com lápis e papel leva facilmente ao valor de cada $p(n)$ em função de $p(0)$. Mas vamos usar o Maple para isso, dando simultaneamente um nome à equação obtida.

```
> eqn:=p(n)=rsolve(%,p(n));
```

$$eqn := p(n) = p(0)(1+i)^n$$

Isto pode ser usado para criar uma função que, a partir dum dado capital inicial $p(0)$, calcula o capital que se tem ao fim de n anos. A fórmula para $p(n)$ é:

```
> subs(eqn,p(n));
```

$$p(0)(1+i)^n$$

Uma função que calcula esta expressão em função de n e i é:

```
> f:=unapply(%,n,i);
```

$$f := (n,i) \to p(0)(1+i)^n$$

Portanto, o capital após 10 anos à taxa de 18% é:

```
> f(10,.18);
```

$$5.233835554p(0)$$

Para $p(0) = 1000$ tem-se:

```
> subs(p(0)=1000,%);
```

$$5233.835554$$

Exercício 3. Use a função f para determinar qual o capital que terá num certo banco daqui a 9 anos, onde depositou 1500 euros no dia 31 de Dezembro de 2001 a uma taxa anual de 15%.

4. Iteração

(a) Iteração sobre uma progressão aritmética (*for loop*):

```
> for i from 5 to 11 by 3 do
> i*i;
> od;
```

```
25
64
121
```

(b) Iteração sobre os elementos de uma lista (*for loop*):

```
> mylist:=[5,8,11]:
> for i in mylist do
> i*i;
> od;
```

```
25
64
121
```

Alternativamente:

```
> mylist:={5,8,11}:
> for i in mylist do
> i*i;
> od;
```

```
25
64
121
```

Exercício 4. Calcule o cubo dos números 2, 4, 8 e 248.

(c) Iterações mais gerais (*while loop*):

```
> i:=3:
> while i<=9 do
> print(i*i);
> i:=i+2;
> od:
```

```
      9
      25
      49
      81
```

```
> i:=1:
> while i^7 < 150 do
> print(i);
> i:=i+i^2;
> od:
```

```
      1
      2
```

Também se pode combinar o *for* com o *while*:

```
> for i from 3 to 44 while i^2<50 do
> i, i^2;
> od;
```

```
      3, 9
      4, 16
      5, 25
      6, 36
      7, 49
```

Exercício 5. Use um *while loop* para determinar todos os valores inteiros positivos n tais que $n^3 < 500$.

Exercício 6. Determine o valor de $\sqrt{n}$ para todo o n de 1 a 10,

(a) usando um *for loop*.

(b) usando um *while loop*.

5. Lógica

Em Maple, **true** e **false** designam os valores lógicos **V** e **F**.

```
> true, false;
```

true, false

Atribuindo o valor V à proposição p,

```
> p:=true;
```

p :=true

Em toda a referência posterior, p é considerada com o valor V. Se quisermos que p volte ao seu significado anterior, isto é, que seja apenas a variável p, basta escrever:

```
> p:='p';
```

$p := p$

A negação, a conjunção e a disjunção:

```
> not p;
```

$\neg p$

```
> p and q;
```

$p \wedge q$

```
> p or q;
```

$p \vee q$

Se atribuirmos valores lógicos a p e q, podemos calcular os respectivos valores lógicos das expressões anteriores:

```
> p:=true: q:=false:
> not p;
```

false

```
> p and q;
```

false

```
> p or q;
```

true

6. Funções e procedimentos

Uma função pode ser definida facilmente em Maple usando o operador $->$. Para uma função de p variáveis $v_1, v_2, \ldots, v_p$ a sintaxe é:

$$(\mathbf{v1}, \mathbf{v2}, ..., \mathbf{vp}) -> \mathbf{exp}$$

onde **exp** é uma expressão escrita explicitamente como uma função das variáveis $v_1, v_2, \ldots, v_p$ que corresponde ao resultado dado pela função.
Por exemplo:

```
> f_1:=x->(x-1)^2:
> f_2:=(x,y)->xy:
> f_3:=(a,b,c)-> a and b or not c:
```

Podemos agora calcular o valor das funções para determinados valores das variáveis:

```
> f_1(7);
```

$$36$$

```
> f_2(2,-3);
```

$$-6$$

```
> f_3(true,true,false);
```

$$\text{true}$$

Podemos também aplicar a função a valores indeterminados:

```
> f_1(t);
```

$$(t-1)^2$$

```
> f_1(a/b);
```

$$(\frac{a}{b}-1)^2$$

Se for necessário ver qual a actual definição de f, usa-se o comando **eval**:

```
> eval(f_1);
```

$$x \to (x-1)^2$$

Um procedimento consiste em várias instruções delimitadas pelas duas palavras reservadas **proc** e **end**.
A seguir vamos usar procedimentos com a seguinte sintaxe simples:

proc(v1,v2,...vp)
<instrução1>;

<instrução2>;
...
<instruçãon>;
end

Um exemplo:

```
> f:=proc(x,y)
> if x<y then x+y
> else x*y fi;
> end:
```

Note que usamos um **if**, com a seguinte sintaxe:

if <condição1> then <expressão1>
else <expressão2> fi

Aqui **fi** está em vez de *end if*.

Usando o procedimento acima podemos agora calcular valores de $f(x, y)$:

```
> f(1,2);
```

$$3$$

```
> f(2,1);
```

$$2$$

7. Conjuntos

O conjunto vazio:

```
> {};
```

$$\{\ \}$$

Um conjunto em Maple pode conter qualquer objecto conhecido pelo `Maple`. Exemplos típicos são:

```
> {1,2,3};
```

$$\{1, 2, 3\}$$

```
> {a,b,c};
```

$$\{a, b, c\}$$

Um dos comandos mais úteis para construir conjuntos ou listas é **seq**. Por exemplo para obter todos os quadrados módulo 30 de inteiros positivos inferiores a 61:

```
> s1:=seq(i^2 mod 30,i=1..60);
```

$$s1 := 1, 4, 9, 16, 25, 6, 19, 4, 21, 10, 1, 24, 19, 16, 15, 16, 19, 24, 1, 10, 21, 4, 19, 6, 25, 16, 9, 4, 1, 0, 1, 4, 9, 16, 25, 6, 19, 4, 21, 10, 1, 24, 19, 16, 15, 16, 19, 24, 1, 10, 21, 4, 19, 6, 25, 16, 9, 4, 1, 0$$

Se quisermos o conjunto de todos esses quadrados módulo 30:

```
> s2:={s1};
```

$$s2 := \{0, 1, 4, 6, 9, 10, 15, 16, 19, 21, 24, 25\}$$

Note-se que, sendo s2 um conjunto, dentro das chavetas não são repetidos elementos.
Um exemplo interessante obtém-se com o procedimento **randpoly** que cria um polinómio aleatório de grau igual ao especificado pela opção **degree**. Vamos gerar um conjunto constituído por 5 polinómios quadráticos aleatórios:

```
> {seq(randpoly(x,degree=2),i=1..5)};
```

$$\{-84x^2 + 19x - 50, 83x^2 - 86x + 23, 88x^2 - 53x + 85, 41x^2 - 58x - 90, 53x^2 - x + 94\}$$

A ordem pela qual o `Maple` escreve os elementos de um conjunto não é a mesma usada pelo utilizador para definir esse conjunto:

```
> A:={2,4,1,0,5};
```

$$A := \{0, 1, 2, 4, 5\}$$

Isto acontece porque o `Maple` escreve os elementos de um conjunto pela ordem pela qual eles estão armazenados na memória.
Por definição, os elementos de um conjunto não têm de aparecer por nenhuma ordem particular, e o `Maple` tira vantagem desse facto para organizar o armazenamento dos conjuntos e elementos de tal modo que as comparações sejam mais fáceis para o `Maple`.
Se a ordem for importante, deve usar-se listas em vez de conjuntos:

```
> r:=[2,4,1,0,5];
```

$$r := [2, 4, 1, 0, 5]$$

As listas de números podem ser ordenadas por ordem crescente:

```
> sort([5,3,1,9,10]);
```

$$[1, 3, 5, 9, 10]$$

Para determinar o número de elementos de um conjunto, usa-se o comando **nops**:

```
> nops({seq(i,i=12..97)});
```

$$86$$

Se quisermos saber quantos elementos distintos existem numa lista, transformamos primeiro a lista num conjunto:

```
> L:=[seq(i^3 mod 15, i=1..34)];
```

$$L := [1, 8, 12, 4, 5, 6, 13, 2, 9, 10, 11, 3, 7, 14, 0, 1, 8, 12, 4, 5, 6, 13, 2, 9, 10, 11, 3, 7, 14, 0, 1, 8, 12, 4]$$

```
> Lc:= convert(L,set);
```

$$Lc := \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}$$

```
> nops(Lc);
```

$$15$$

Para comparar dois conjuntos, escreve-se a igualdade $A = B$ e força-se a comparação com o comando **evalb**.

```
> A:={seq(4*i,i=4..666)}:
> B:={seq(2*i+2,i=4..666)}:
> evalb(A=B);
```

$$false$$

Para a diferença de conjuntos, usa-se **minus**; para a união, **union**; para a intersecção, **intersect**:

```
> A:={1,2,3,4}; B:={2,1,3,2,2,5};
```

$$A := \{1, 2, 3, 4\}$$
$$B := \{1, 2, 3, 5\}$$

```
> A minus B;
```

$$\{4\}$$

```
> B minus A;
```

$$\{5\}$$

```
> A union B;
```

$$\{1, 2, 3, 4, 5\}$$

```
> A intersect B;
```

$$\{1, 2, 3\}$$

modp usa-se para determinar o resto da divisão; assim:

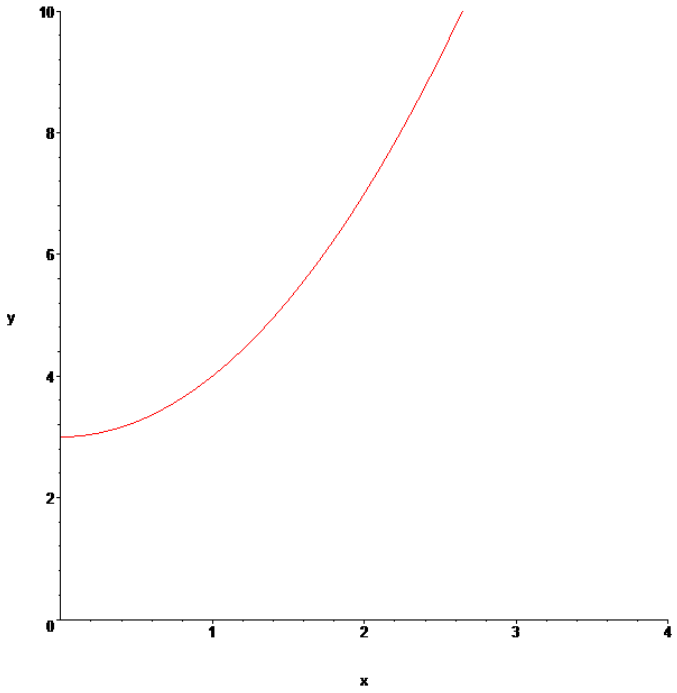
```
> modp(14,5);
```

$$4$$

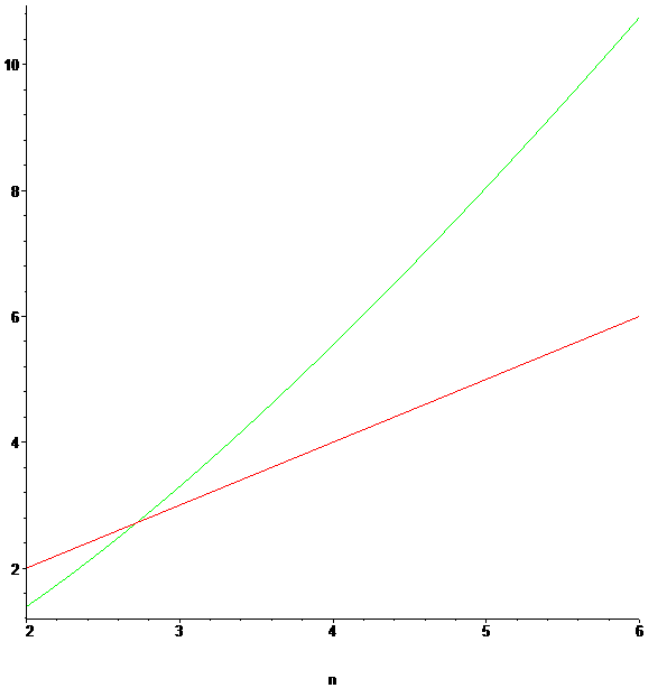
8. Gráficos

Basta usar o comando **plot**:

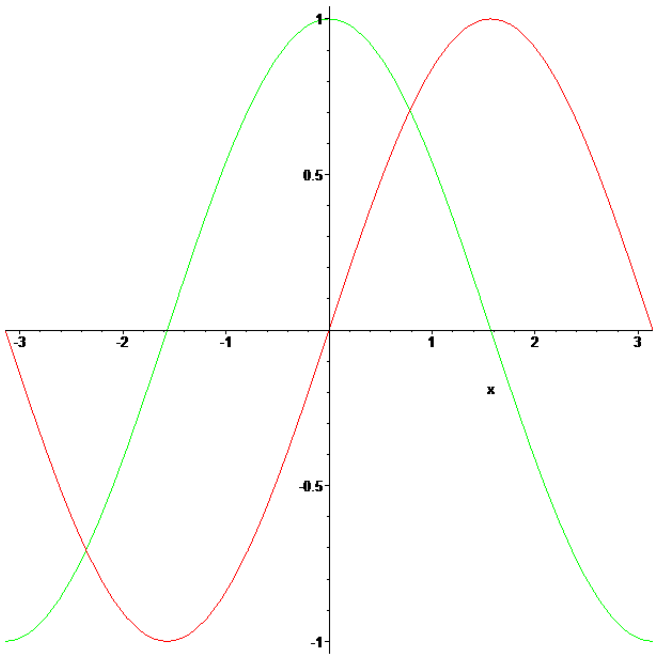
```
> plot(x^2+3, x=0..4, y=0..10);
```



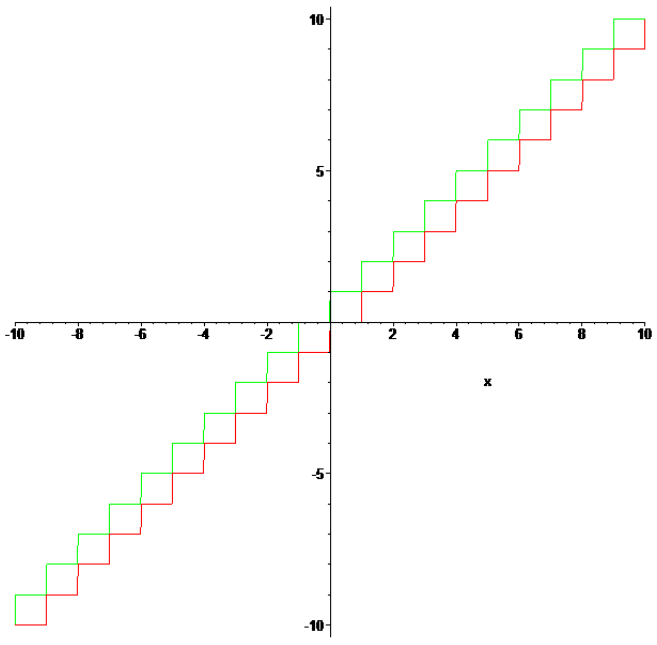

```
> plot({n,n*ln(n)}, n=2..6);
```



```
> plot({sin(x),cos(x)},x=-Pi..Pi);
```




```
> plot({floor(x), ceil(x)}, x=-10..10);
```



floor (função *floor*): característica de um número, i.e. o maior número inteiro que é menor ou igual que o número dado; **ceil** (função *ceiling*): menor número inteiro que é maior ou igual do que o número.

9. Grafos em Maple

O Maple tem vários comandos relacionados com teoria de grafos, que fazem parte da *package* **networks**.

```
> with(networks):
```

Para criar um novo grafo, G1, usa-se o comando **new**:

```
> G1:=new():
```

Este grafo ainda não tem vértices nem arestas, têm de se lhe juntar usando comandos como **addvertex**, **addedge** e **connect**.
Vamos construir um grafo não dirigido simples onde os vértices são cidades e as arestas representam redes de comunicação entre elas.

```
> addvertex({‘Porto‘,‘Viseu‘,‘Aveiro‘,‘Coimbra‘,‘Braga‘,‘Leiria‘},G1);
```

$$Porto, Coimbra, Aveiro, Viseu, Leiria, Braga$$

Vamos agora introduzir as arestas:

```
> addedge({{‘Porto‘,‘Viseu‘}, {‘Porto‘,‘Coimbra‘}, {‘Viseu‘, ‘Coimbra‘},
{‘Viseu‘, ‘Aveiro‘}, {‘Viseu‘, ‘Braga‘}, {‘Braga‘, ‘Leiria‘}}, G1);
```

$$e1, e2, e3, e4, e5, e6$$

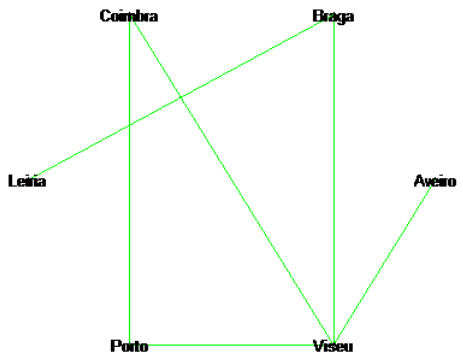
Para saber quais os vértices associados a uma aresta:

```
> ends(e2,G1);
```

$$\{Porto, Coimbra\}$$

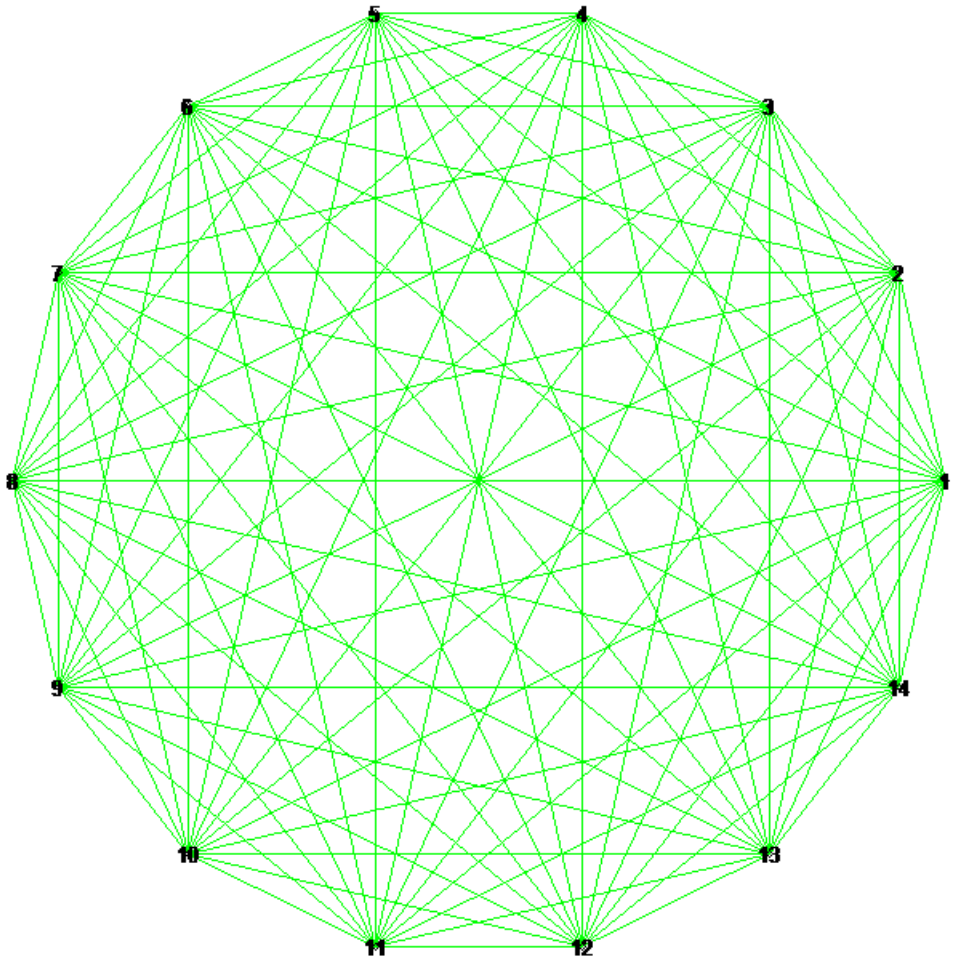
Para representar um grafo graficamente usa-se o comando **draw**:

```
> draw(G1);
```



O grafo completo de ordem n é obtido por meio de **complete(n)**:

```
> K14:=complete(14):
> draw(K14);
```



Para introduzir um grafo com pesos procede-se como ilustrado a seguir:

```
> new(S1):
> addvertex({a,b,c,d,y,z},S1);
```

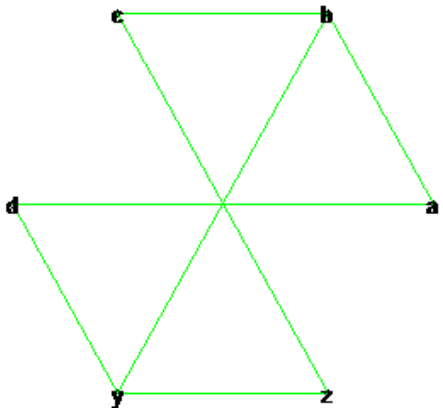
$$a, z, c, d, y, b$$

```
> addedge([{a,b},{a,d},{b,c},{b,y},{c,z},{d,y},{y,z}], weights=[4,2,3,3,2,3,1],S1);
```

$$e1, e2, e3, e4, e5, e6, e7$$

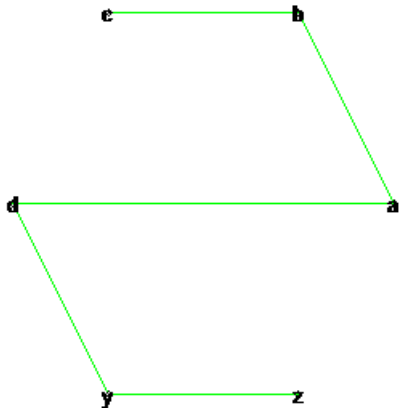
O comando **draw** não indica os pesos:

```
> draw(S1);
```



O comando **shortpathtree** usa o algoritmo de Dijkstra para construir uma árvore dos caminhos mais curtos de um dado vértice para todos os outros:

```
> T1:=shortpathtree(S1,a):
> draw(T1);
```



Agora para saber a distância entre a e um dado vértice basta usar **vweight**:

```
> vweight(z,T1);
```

$$6$$