Процедурно програмиране

Процедурно програмиране = дава стъпка по стъпка инструкции на компютъра.

Програмата се разделя на процедури (поредица от инструкции, които се изпълняват в определен ред) и данни.
Програмата е структурирана чрез:
- основен блок
- подпрограми
- правила за видимост и жизнен цикъл на програмата.

Основни конструкции:
- цикли (Do, For, While);
- условни разклонения (If);
- управление на потока на изпълнение.

Организиране на цикли с Do, For, While

Ще разгледаме традиционните начини за организиране на цикли в *Mathematica*, т. е. начини за краен, предварително определен брой пъти повторения на блокове от кодове.

Оператор за цикъл Do

Do[израз, {i, imin, imax, d}] изчислява израза за стойности i=imin, imin+d, ... докато все още е по-малък или равен на imax.
Пример: Сумата от квадратите на първите 10 естествени числа.

```
In[1]:= x=0; Do[x=x+k^2, {k,1,10}]; x
Out[1]= 385
```

Ако искаме да видим последователните резултати от изпълнението на цикъла, използваме Print.

```
In[2]:= x=0; Do[x=x+k^2; Print["Итерация", k,":",x] {k,1,10}]; x
```

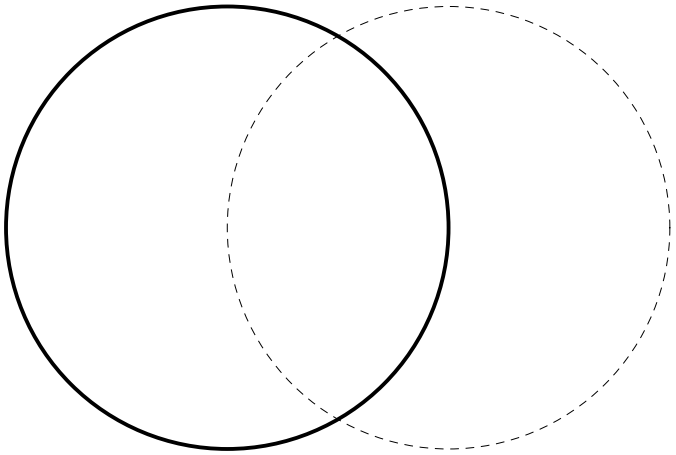
Syntax::sntxf: "+k^2; Print["Итерация", k," cannot be followed by
  "":,x] {k,1,10}]; x".

Графични директиви

Графичните директиви указват как да бъдат визуализирани основните графични елементи. Те се използват за да се модифицират основните елементи. Например единия кръг може да бъде плътен (Thick), а другия - пунктиран (Dashed).

In[2]:=
```
Graphics[{{Thick, Circle[{0,0},1]},{Dashed,Circle[{1,0},1]}}]
```

Out[2]=



Опции

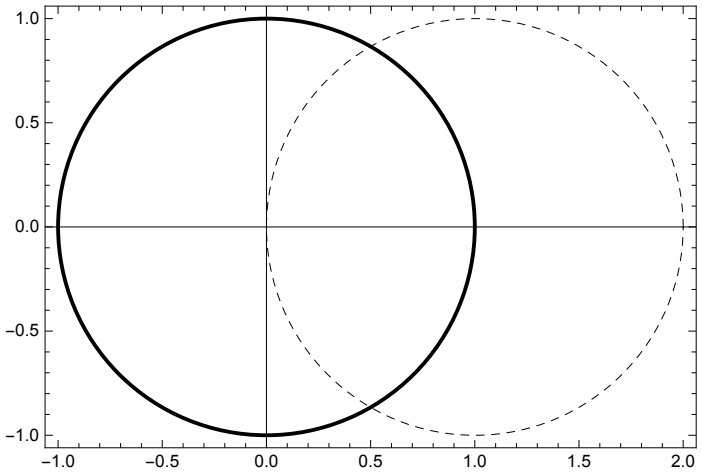
Използват се като настройки за да модифицират цялата графика. Цялата графика се оформя чрез използването на опциите. Ако липсват параметри за настройки (опции), то те са по подразбиране. Ако искаме допълнителна настройка, то е необходима собствена настройка, която се задама с опциите. Всяка опция има име. Те трябва да са след елементите или списъците с основни елементи, дадени като аргументи на функцията Graphics.

Задаването на потребителски настройки става чрез поредица от правила от вида име - > стойност.

Например, да добавим координатните оси (чрез Axes) и да сложим графиката в рамка (чрез Frame).

In[3]:=
```
Graphics[{{Thick, Circle[{0,0},1]},{Dashed,Circle[{1,0},1]}}, Axes ->
```

Out[3]=



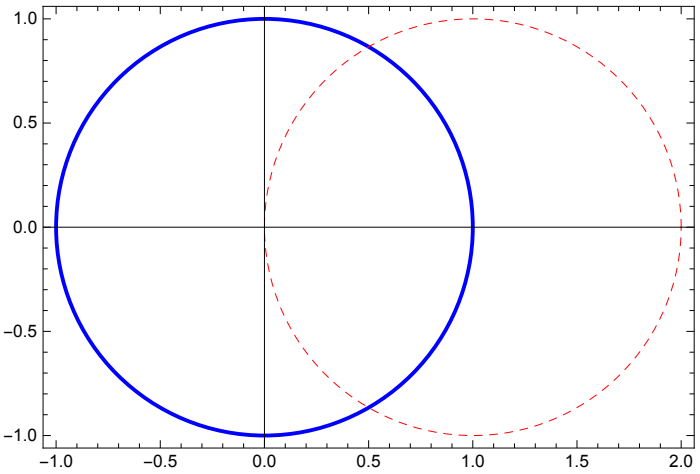
Смяна на цвят

ТОва е отново директива на графиката и става с помощта на RGBColor[{r,g,b}] - променя цвета на червен, зелен и син, като яркостта му се изразява с число от 0 до 1.

CMYKColor[{c,m,y,k}] променя цвета на синьо-зелен, червен, жълт и черен, като яркостта му се изразява с число от 0 до 1.

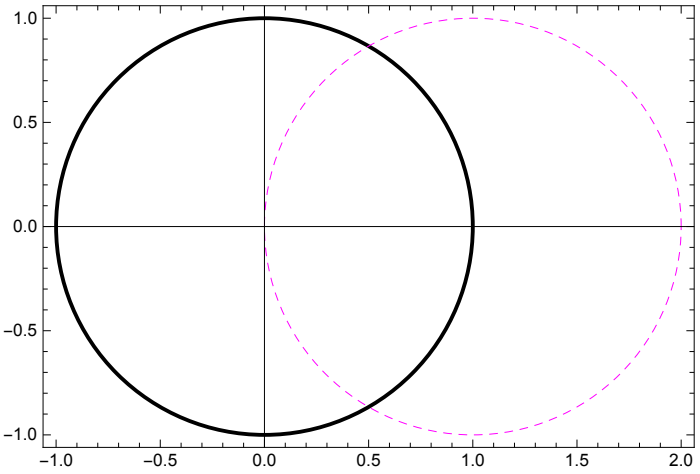
In[4]:=
```
Graphics[{{RGBColor[0,0,1], Thick, Circle[{0,0},1]}, {RGBColor[1,0,0],
```

Out[4]=



In[5]:=
```
Graphics[{{CMYKColor[{1,0,0,1}], Thick, Circle[{0,0},1]}, {CMYKColor[{
```

Out[5]=



Тримерни графики

За 3D графика се използва вградената функция Graphics3D, която изобразява основните елементи. Например, две сфери като основни елементи са в червен, и цилиндър в синьо.
Използват се две опции:
- FaceGrids добавя мрежа към всяка от стените на кутията;
- ViewVertical използвано е за да смени вертикалната посока на изображението.

In[6]:=
```
Graphics3D[{{Red, Sphere[{0,0,0}], Sphere[{2,2,2}]}, {Blue, Cylinder[{
```

Out[6]=

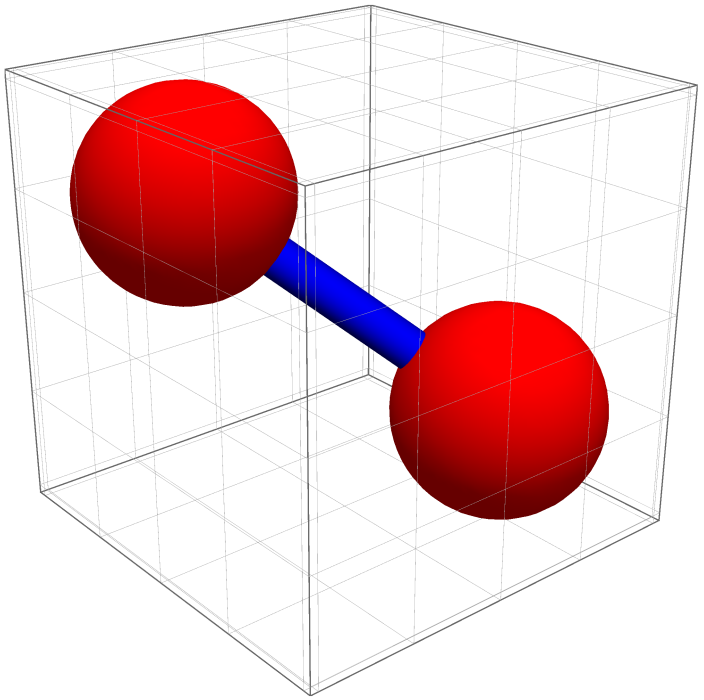


Таблица на основните графични елементи

`Point[{x,y}]` - точка с дадени координати

`Line[{{x1,y1},{x2,y2},...}]` - права през зададените точки

Rectangle[{{xmin,ymin},{xmax,ymax}}] - плътен правоъгълник

Двумерна графика на функции

Използва се командата Plot, която има два аргумента, отделени със запетая,

Plot[функция, {x, xmin, xmax}]

Първият аргумент е функцията, чиято графика се чертае, а втория се нарича итератор, оформен като списък. Той описва интервала от стойности на аргумента, в които се чертае графиката: от xmin до xmax.

Командата Plot може да чертае и няколко графики едновременно на една и съща координатна система като в този случай има вида

Plot[{функция1, функция2}, {x, xmin, xmax}].

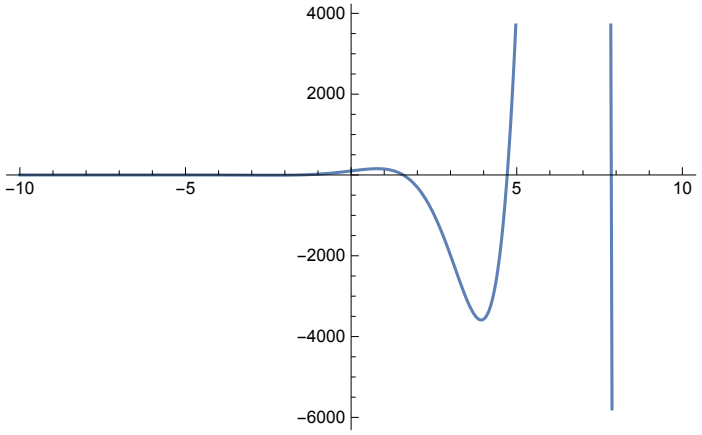
Всеки път, когато се използва Plot, *Mathematica* решава по подразбиране къде да сложи осите, и те НЕ ВИНАГИ се пресичат в началото (0,0). Ако искате да използвате осите, които се пресичат в началото, то трябва допълнително да се окаже това.

Например, да начертаем графиката на функцията 100cos(x)e^x в интервала от -10 до 10.

In[7]:=
```
f[x_]:=100*Cos[x]*Exp[x]
```

In[8]:=
```
Plot[f[x],{x,-10,10}]
```

Out[8]=



В този случай се оказва, че графиката е отрязана по y-оста и е необходимо да се разшири множеството от стойности на функцията, дадени на графиката.

Промяната на вертикалната скала става с PlotRange.

Тази вградена функция се слага като допълнителен аргумент на Plot. Трябва да се сложи след двата задължителни аргумента и е от вида

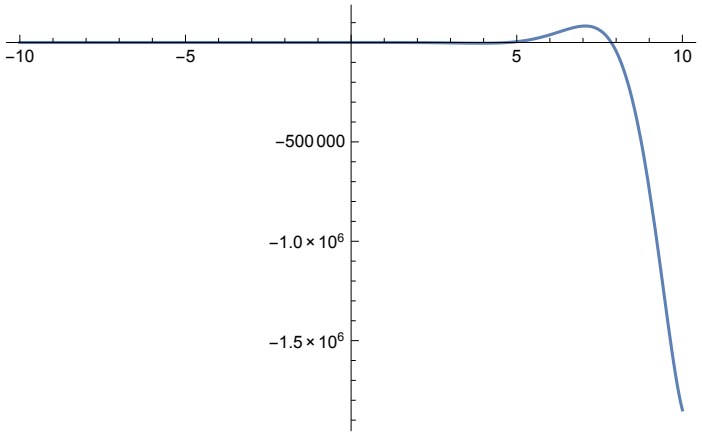
PlotRange -> Full или

PlotRAnge -> {a,b}

Ще използваме PlotRange -> Full

In[9]:=
```
Plot[f[x],{x,-10,10},PlotRange -> All]
```

Out[9]=

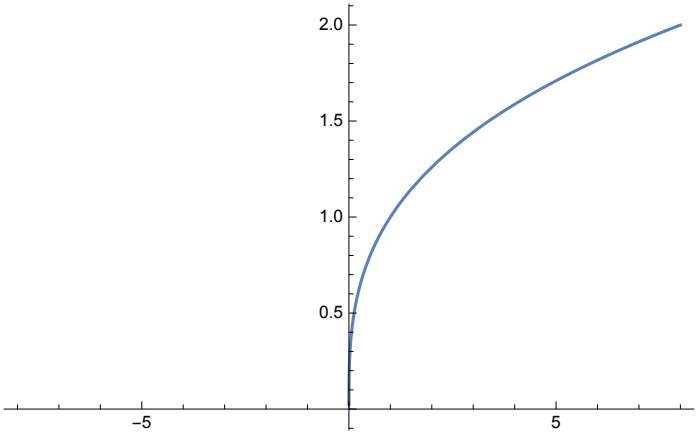


Накои куриозни примери:

Да начертаем графиката на кубичен корен от x

In[10]:=
```
Plot[x^(1/3),{x,-8,8}]
```

Out[10]=



Няма ли кубични корени от отрицателни числа? Ами кубичен корен от -8 е -2?! А защо го няма на графиката? Причината е в представянето на корена като комплексно число.

За чертане на тримерна графика се използва командата Plot3D. Например да начертаем графиката на функцията

```
Sin[x^2+y^2]*Exp[-x^2]+Cos[x^2+y^2]
```

In[11]:=
```
f[x_,y_]=Sin[x^2+y^2]*Exp[-x^2]+Cos[x^2+y^2];
Plot3D[f[x,y],{x,-2,2},{y,-2,2}]
```

Out[12]=