Mathematica като среда за програмиране

Програмиране=когато се поставят няколко команди заедно, за постигне на обща цел (тази цел обикновено не може да се постигне с коя да е команда поотделно)

С някои от основните елементи на програмирането вече се запознахме:

-дефиниране на променливи:

x=стойност x=y=стойност

-**дефиниране на потребителски функции**:

f[x_]= израз на x

-**незабавно и отложено присвояване**:

променлива=израз, който се изчислява и веднага се присвоява на променливата

променлива:= израз, който се изчислява и се присвоя на променливата, едва, след като се обърнем към променливата

за предпочитане

Mathematica като език за програмиране

Mathematica включва различни парадигми и методи за програмиране, от които най важните са

- ***Фунционално програмиране***= използването на функциите като основен стил на програмиране

-***Процедурно програмиране***=програмата се разделя на

процедури(поредица от инструкции, които се изпълняват в определен ред) и

данни

Функционално програмиране.

Функционално програмиране= използването на функциите като основен стил на програмиране

Основна идея : прилагането на функции върху функции(същите или други) с цел създаване на друга функция

Ключов елемент: Възможността една функция да се използва като аргумент на друга функция, да бъдат връщани като стойност, или да бъдат част от обекти, като масивите, например.

Функционалното програмиране включва прилагането на функции към аргументите им и обикновено действа на целия израз изведнъж.

Функции за манипулиране на изрази (Map, Apply, Thread)

Map [функция, списък] - прилага функцията към всеки елемент на списъка

```
In[2]:= Map[f, {a, b, c}]
Out[2]= {f[a], f[b], f[c]}
```

Пример: даден е двумерен списък. Да разменим местата на съответните елементи на списъка.

```
In[4]:= Map[Reverse, {{a, b}, {c, d}, {e, f}}]

Out[4]= {{b, a}, {d, c}, {f, e}}
```

Пример: да повдигнем на квадрат всеки елемент на даден списък и да добавим по 1

Създаваме списък

```
In[6]:= vec = {2, π, γ}
```

Дефинираме функция

```
In[7]:= f[x_] := x^2 + 1
```

Прилагаме я към всеки елемент на списъка

```
In[8]:= Map[f, vec]

Out[8]= {5, 1 + π^2, 1 + γ^2}
```

Пример: даден е списък от константи. Да проверим коя от тях е просто число

```
In[7]:= Map[PrimeQ, {2, 3, 4, 5, 6, 2.4, -2, 0}]

Out[7]= {True, True, False, True, False, False, True, False}
```

Apply [функция, израз] към главната част на израза се прилага функцията

Използва се, за да смени структурата на израз

```
In[10]:= Apply[h, g[a, b, c]]
Out[10]= h[a, b, c]
```

Функцията h е приложена към израза и се променя главната част на израза g с h

```
In[13]:= Apply[Plus, {1, 2, 3, 4}]
Out[13]= 10
```

Функцията Plus се прилага към списъка и променя структурата му, става `Plus[a, b, c, d]`

което е вътрешното представяне на сумата на тези 4 символа

```
In[14]:= Plus[a, b, c, d]
Out[14]= a + b + c + d
```

Да сравним действието на Map и Apply

```
In[16]:= Map[h, {{a, b}, {c, d}}]
Out[16]= {h[{a, b}], h[{c, d}]}
```

Функцията действа върху всеки елемент на списъка

```
In[18]:= Apply[f, {{a, b}, {c, d}}]
Out[18]= f[{a, b}, {c, d}]
```

Функцията действа върху главната част на списъка и променя вида му

Thread[функция, списък] - прилага функцията към всеки елемент на списъка

Thread “нанизва” функцията към няколко списъка

Пример: Да дефинираме функцията на два аргумента

```
In[24]:= f[x_, y_] := x^2 - y^3

In[25]:= Thread[f[{1, 2, 3}, {2, 3, 4}]]
Out[25]= {-7, -23, -55}
```

В двата списъка, функцията взема първите елементи на всеки списък като стойности на двата си аргумента съответно и я прилага, т.е. 1^2-2^3=-7, после 2^2-3^3=-23 и т.н.

Итеративни функции Nest, NestList, Fold, FoldList

Често се налага една функция да се прилага последователно, при което се оформя функция от функция, сложна функция. Тогава е удобно да се използва вградената функция

Nest[функция, xo,n]

xo е начална стойност на аргумента, n- брой пъти на прилагане на функцията

```
In[1]:= Nest[g, a, 4]
Out[1]= g[g[g[g[a]]]]
```

NestList[функция, xo, n]

Дава отделните прилагания на функцията

```
In[2]:= NestList[g, a, 4]
Out[2]= {a, g[a], g[g[a]], g[g[g[a]]], g[g[g[g[a]]]]}
```

Примери използване на Nest

Да се дефинира функцията

$$\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+x}}}}}$$

```
In[9]:= f[x_] = 1 / (1 + x)
```

$$\text{Out[9]=}\ \frac{1}{1+x}$$

```
In[10]:= Nest[f, x, 7]
```

$$\text{Out[10]=}\ \cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+\cfrac{1}{1+x}}}}}}}$$

Да се намери стойността ѝ при x=1

```
In[11]:= Nest[f, 1, 7]
```

$$\text{Out[11]=}\ \frac{21}{34}$$

Да се дефинира функцията

$$\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+x}}}}$$

```
In[12]:= f[x_] = Sqrt[1 + x]
```

$$\text{Out[12]=}\ \sqrt{1+x}$$

```
In[13]:= Nest[f, x, 4]
```

$$\text{Out[13]=}\ \sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+x}}}}$$

Да се намери стойността й при x=1

```
In[15]:= Nest[f, 1, 4]
```

$$\text{Out[15]=}\ \sqrt{1+\sqrt{1+\sqrt{1+\sqrt{2}}}}$$

```
In[16]:= Nest[f, 1, 4] // N
Out[16]= 1.61185
```

Вградени функции Fold и FoldList

Функцията Fold се използва за да итерира функция на два аргумента само върху първия аргумент

Fold[функция, а, списък]

-функцията е на два аргумента

-*а* е стойността на първия аргумент

-списъка указва стойностите на втория аргумент, върху които се итерира функцията

FoldList[функция, а, списък]

Дава не само резултатът, а и последователните итерации

```
In[36]:= Fold[f, a, {1, 3, 5, 8, 9}]

Out[36]= f[f[f[f[f[a, 1], 3], 5], 8], 9]

In[38]:= FoldList[f, a, {1, 3, 5, 8, 9}]

Out[38]= {a, f[a, 1], f[f[a, 1], 3], f[f[f[a, 1], 3], 5],
          f[f[f[f[a, 1], 3], 5], 8], f[f[f[f[f[a, 1], 3], 5], 8], 9]}
```

```
In[58]:= FoldList[f, 1, Range[2, 5]]
Out[58]= {1, f[1, 2], f[f[1, 2], 3], f[f[f[1, 2], 3], 4], f[f[f[f[1, 2], 3], 4], 5]}
```

Предполага се , че функцията е на 2 аргумента, като първия аргумент винаги остава 1, а втория итеративно приема стойности 2,3, 4, 5

```
In[59]:= f[x_, y_] = x + y
Out[59]= x + y

In[60]:= FoldList[f, 1, Range[2, 5]]
Out[60]= {1, 3, 6, 10, 15}
```

т.е. пресмята итеративно f(1,2), f(3,3)=6, f(6,4)=10,f(10,5)=15,
т.е. показва последователно сумите на 1+2+3+4+5

Пример: Намерете и покажете последователно сумите на всички естествени числа от 1 до n, където n =20,

Т.е. 1

1+2

1+2+3

1+2+3=4

1+2+3+4+5

И т.е.

```
f[x_, y_] = x + y

In[63]:= FoldList[f, 1, Range[2, 20]]

Out[63]= {1, 3, 6, 10, 15, 21, 28, 36, 45, 55, 66, 78, 91, 105, 120, 136, 153, 171, 190, 210}
```

Използвайте Column [ ] за да получите резултата като стълб

Чисти функции (Function, #, &)

Function[списък от *входна променливи, тяло*]

Тялото е израз на входните променливи

```
In[9]:= Function[x, x^2 + 2][a]
Out[9]= 2 + a^2

In[11]:= Map[Function[x, x^2 + 2], {a, b, c}]
Out[11]= {2 + a^2, 2 + b^2, 2 + c^2}
```

За по-лесно се пише само един аргумент, тялото, като променливата се замества с # **Function[#]**

```
In[12]:= Map[Function[#^2 + 2], {a, b, c}]
Out[12]= {2 + a^2, 2 + b^2, 2 + c^2}

In[17]:= Map[#^2 + 2 &, {a, b, c}]
Out[17]= {2 + a^2, 2 + b^2, 2 + c^2}
```

Символът & записан заедно с празно след израз показва, че този израз е тялото на чиста функция

Чисти функции с повече аргумента

```
In[14]:= Function[{x, y, z}, x^2 - y^3 + 2 z][a, b, c]
```

Out[14]= $a^2 - b^3 + 2\,c$

```
In[13]:= Function[{x, y, z}, x^2 - y^3 + 2 z][a, b]
```

Function::fpct : Too many parameters in {x, y, z} to be filled from Function[{x, y, z}, $x^2 - y^3 + 2\,z$][a, b]. >>

Out[13]= Function[{x, y, z}, $x^2 - y^3 + 2\,z$][a, b]

```
In[16]:= Function[#1^2 - #2^3 + 2 #3][a, b, c]
```

Out[16]= $a^2 - b^3 + 2\,c$

Пример. От даден списък с тримерни елементи да се отделят само първите елементи на всеки елемент на списъка

```
In[19]:= Map[Take[#, 2] &, {{1, 2, 3}, {2, 3, 4}, {3, 4, 5}, {4, 5, 6}}]

Out[19]= {{1, 2}, {2, 3}, {3, 4}, {4, 5}}
```

Пример. Напечатайте стойностите на числото π с 0, 1, 2, 3, 4, и т.н до 9 знака след десетичната точка

```
In[21]:= Map[N[Pi, #] &, Range[10]]

Out[21]= {3., 3.1, 3.14, 3.142, 3.1416, 3.14159,
          3.141593, 3.1415927, 3.14159265, 3.141592654}

In[22]:= Column[Map[N[Pi, #] &, Range[10]]]

Out[22]= 3.
         3.1
         3.14
         3.142
         3.1416
         3.14159
         3.141593
         3.1415927
         3.14159265
         3.141592654
```

А сега да го оформим като табличка

Пример: Намерете редицата от стойности на n-тите частични суми на хармоничния ред , н=1,2,3,…,10

$$1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n}$$

```
In[45]:= f[x_, y_] := x + 1 / y
         FoldList[f, 1, Range[2, 10]]
```

$$\text{Out[46]}= \left\{1, \frac{3}{2}, \frac{11}{6}, \frac{25}{12}, \frac{137}{60}, \frac{49}{20}, \frac{363}{140}, \frac{761}{280}, \frac{7129}{2520}, \frac{7381}{2520}\right\}$$

Или по кратко

```
In[47]:= FoldList[#1 + 1 / #2 &, 1, Range[2, 10]]
```

$$\text{Out[47]}= \left\{1, \frac{3}{2}, \frac{11}{6}, \frac{25}{12}, \frac{137}{60}, \frac{49}{20}, \frac{363}{140}, \frac{761}{280}, \frac{7129}{2520}, \frac{7381}{2520}\right\}$$

Или

```
In[48]:= FoldList[#1 + 1 / #2 &, 1., Range[2, 10]]
Out[48]= {1., 1.5, 1.83333, 2.08333, 2.28333, 2.45, 2.59286, 2.71786, 2.82897, 2.92897}
```

К р а й

Контролна работа 3

Задача 1. Да се намери най голямото отрицателно решение на уравнението $\cos x = \sin(3-x)$

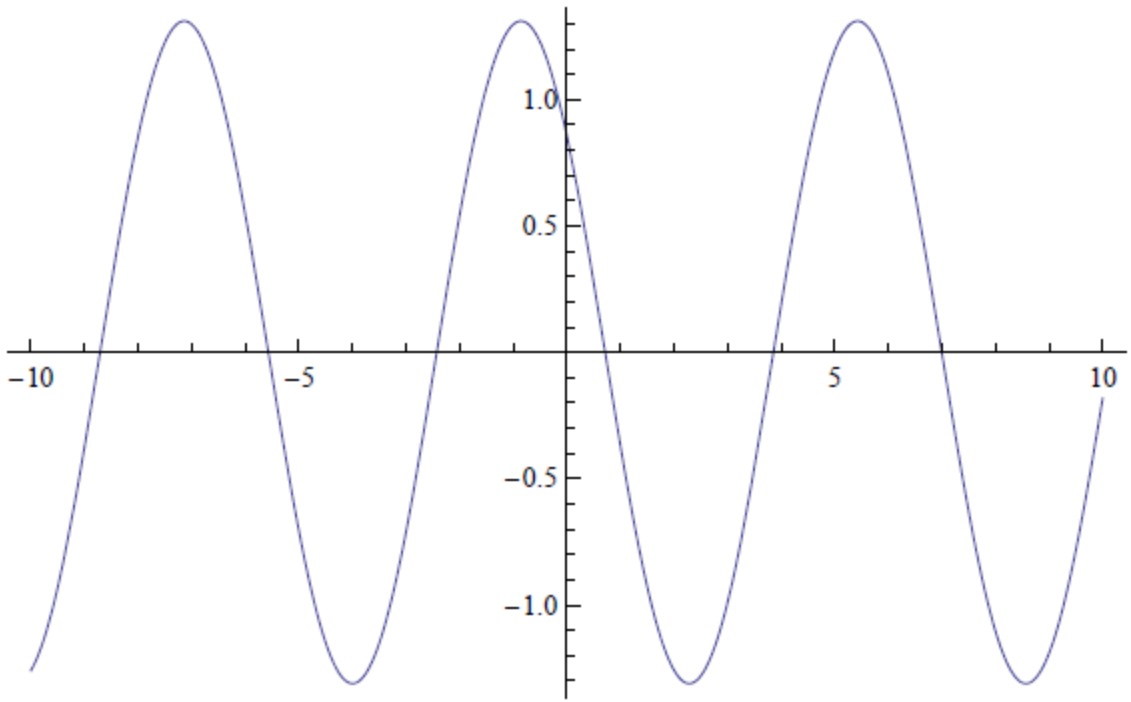
Задача 2. Намерете естествено число n, за което

$$\frac{1}{1*2}+\frac{1}{2*3}+\frac{1}{3*4}+\ldots+\frac{1}{n*(n+1)}=\frac{17}{18}$$

Задача 1.

```
In[68]:= Plot[Cos[x] - Sin[3 - x], {x, -10, 10}]
```

Out[68]=




```
In[70]:= FindRoot[Cos[x] - Sin[3 - x], {x, -2}]
Out[70]= {x → -2.42699}
```

Задача 2.

```
In[92]:= Solve[Sum[1 / (x (x + 1)), {x, 1, n}] == 17 / 18, n]
Out[92]= {{n → 17}}
```