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 повторение 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 < или = на imax.

Пример: Сумата от квадратите на първите 10 естествени числа

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

Занулляваме променливата x

Указваме да се отпечати стойността на променливата x

Оформяме цикъл за к от 1 до 10

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

```
x = 0; Do[x = x + k^2; Print["iteration ", k, ": ", x], {k, 1, 10}]
iteration 1: 1
iteration 2: 5
iteration 3: 14
iteration 4: 30
iteration 5: 55
iteration 6: 91
iteration 7: 140
iteration 8: 204
iteration 9: 285
iteration 10: 385
```

Пример. Да намерим всички прости числа по-малки от предварително зададено число.

Дефинираме функция

```
In[1]:= prime[n_] := Select[Range[n], PrimeQ]
```

Да намерим всички прости по-малки от 20

```
In[2]:= prime[20]
Out[2]= {2, 3, 5, 7, 11, 13, 17, 19}
```

Да намерим последователно, всички прости числа по-малки от 1,2,3, и т.н до 30, тогава организираме цикъл

```
In[4]:= Do[Print[i, ":", prime[i]], {i, 1, 30}]
```

Оператор за цикъл While

While[условие, тяло] изпълнява тялото докато условието е True
Тялото на цикъла може да се състои от няколко оператора, отделени с ;

Това е условието, и докато е верно се изпълнява тялото

```
s = 0; k = 1; While[k ≤ 10, s = s + k^2; k = k + 1]; s
385
```

Това е тялото, което се състои от два оператора за присвояване, отделени с ;

Забележка. Изразът m++ е еквивалентен на m=m+1, което добавя 1 към m.

Пример: Намерете най-малкото просто число>11, което е записано само с 1.

```
In[6]:= n = 111; While[! PrimeQ[n], n = 10 * n + 1]; Print[n]

1 111 111 111 111 111 111
```

!PrimeQ[n] е Булева функция, да припомним, че ! означава отрицание или Not, т.е. ако s е True, тогава !s е False. В случая, n=10n+1 е кода на това което искаме да се повтаря. Кодът n=10n+1 дава числата, записани само с 1. Цикълът While указва повторение на този код докато !PrimeQ[n] стане False, т.е. докато PrimeQ[n] е True, което се случва когато n е просто.

Френският математик Ферма (1601–1665) изказва твърдението, че всяко число от вида $2^{2^n}+1$ е просто. И чак един век по късно, Ойлер дава контрапример на това твърдение. Намерете най-малкото число, което е контрапример на това твърдение.

```
In[17]:= k = 0; While[m := 2^(2^k) + 1; PrimeQ[m], k++]; m
Out[17]= 4 294 967 297
```

Покажете, че полученото число е наистина съставно, т.е. намерете най-малкия делител >1 на полученото число.

```
In[18]:= n = 2; While[m = 4 294 967 297 / n; ! IntegerQ[m], n++]; n
Out[18]= 641
```

Пример. Намерете най-малкото естествено число *m, за което* $529^3 + 132^3 m$ *се дели на* 262417.

Започваме с *m = 1 и докато остатъка при делението на* $529^3 + 132^3 m$ *на* 262417 е различен от 0, на езика на *Mathematica*
While[Mod[529^ 3 + 132^ 3m, 262417] != 0
тогава добавяме 1 към *m,* т.е. *m++, и повтаряме това докато остатъка стане 0.*

```
In[7]:= m = 1; While[Mod[529^3 + 132^3 * m, 262 417] ≠ 0, m++]; m

Out[7]= 1984
```

Пример. Намерете най голямото просто число, което е по-малко от дадено число *n.*

```
In[8]:= n = Input["vyvedi 4islo"]

Out[8]= 100

In[9]:= While[!PrimeQ[n], n--]; Print[n]
        97
```

Тялото на цикъла се състои от един оператор. Операторът Input отваря кутийка и иска стойност. Това е много добър начин да укажем на потребителя да въведе данни. !PrimeQ[n] връща True и повтаря цикъла докато n е просто.
n - - е еквивалентно на n=n-1

Оператор за цикъл For

For[начало, условие, нарастване, тяло] изпълнява началото, после многократно изчислява тялото докато условието е True,като нарастването дава изменението на променливата

Началото е където инициализираме променливите използвани в тялото на цикъла. Втората част, условието е булев израз, който определя докато да се изпълнява цикъла. Всяка от тези части могат да съдържат няколко команди, отделени с ;

Да разгледаме отново получаването на сумата от квадратите на естествените числа по-малки от 10

Това е началото, което се състои от два оператора за присвояване, отделени с ;

тялото

```
In[4]:= For[s = 0; k = 1, k ≤ 10, k = k + 1, s = s + k^2]; s
Out[4]= 385
```

условие нарастването

За да проследим изпълнението стъпка по стъпка на цикъл,използваме **Trace**

```
In[7]:= x = 0; Do[x = x + k^2, {k, 1, 10}]; x
Out[7]= 385

In[8]:= x = 0; Trace[Do[x = x + k^2, {k, 1, 10}]]
Out[8]= {Do[x = x + k^2, {k, 1, 10}], {{{x, 0}, {{k, 1}, 1^2, 1}, 0 + 1, 1}, x = 1, 1},
  {{{x, 1}, {{k, 2}, 2^2, 4}, 1 + 4, 5}, x = 5, 5},
  {{{x, 5}, {{k, 3}, 3^2, 9}, 5 + 9, 14}, x = 14, 14},
  {{{x, 14}, {{k, 4}, 4^2, 16}, 14 + 16, 30}, x = 30, 30},
  {{{x, 30}, {{k, 5}, 5^2, 25}, 30 + 25, 55}, x = 55, 55},
  {{{x, 55}, {{k, 6}, 6^2, 36}, 55 + 36, 91}, x = 91, 91},
  {{{x, 91}, {{k, 7}, 7^2, 49}, 91 + 49, 140}, x = 140, 140},
  {{{x, 140}, {{k, 8}, 8^2, 64}, 140 + 64, 204}, x = 204, 204},
  {{{x, 204}, {{k, 9}, 9^2, 81}, 204 + 81, 285}, x = 285, 285},
  {{{x, 285}, {{k, 10}, 10^2, 100}, 285 + 100, 385}, x = 385, 385}, Null}
```

Вложени цикли

Пример. Намерете всички питагорови тройки по малки от 100, т.е. всички естествени числа m, n , такива че m^2+n^2 е точен квадрат.

```
In[10]:= Do[Do[If[Sqrt[n^2 + m^2] ∈ Integers, Print[n, ",", m]], {m, n, 100}], {n, 1, 100}]
```

Външния цикъл с брояч n започва от 1, докато тялото му е вътрешен цикъл с брояч m. В вътрешния цикъл *{j, i, 100}* прави брояча m да приема стойности от n до 100 и да проверява дали корен квадратен от m^2+n^2 е цяло число.

Функционално срещу процедурно програмиране

Да сравним двата основни метода за програмиране
в Mathematica с един пример

Намерете сумата от квадратите на всички естествени числа по-малки от n, където n приема последователно стойности 0, 1,2,3, и т.н до 10.

Ще дадем кода на решението, като използваме функционално програмиране, както и възможностите за процедурно програмиране. Заедно с това ще използваме и функцията **Timing[израз]**, която дава времето (в сек) за изпълнение и резултата. За да сравним, ще използваме сумите не до 10, а до 1000.

Функционално срещу процедурно програмиране

Чрез функционално програмиране

```
In[1]:= FoldList[#1 + #2^2 &, 0, Range[10]]

In[17]:= Timing[FoldList[#1 + #2^2 &, 0, Range[1000]]]
```

0

Чрез процедурно програмиране

- с оператор Do

```
In[8]:= Do[{s = 0; Do[s = s + k^2, {k, 1, n}],
    Print["pri  ", n, "=", "  suma", "=", s]}, {n, 1, 10}]

In[16]:= Timing[Do[{s = 0; Do[s = s + k^2, {k, 1, n}]}, {n, 1, 1000}]]
Out[16]= {0.982806, Null}
```

- с оператор While

```
In[9]:= n = 1; While[n ≤ 10, {s = 0; k = 1; While[k ≤ n, s = s + k^2; k++];
    Print["pri  ", n, "=", "  suma", "=", s], n++}]

In[15]:= Timing[n = 1; While[n ≤ 1000, {s = 0; k = 1; While[k ≤ n, s = s + k^2; k++], n++}]]
Out[15]= {1.825212, Null}
```

- с оператор For

```
In[11]:= For[n = 1, n ≤ 10, n++, For[s = 0; k = 1, k ≤ n, k++, s = s + k^2];
    Print["pri  ", n, "=", "  suma", "=", s]]

In[14]:= Timing[For[n = 1, n ≤ 1000, n++, For[s = 0; k = 1, k ≤ n, k++, s = s + k^2]]]
Out[14]= {1.700411, Null}
```

Функционално срещу процедурно програмиране

Може и чрез комбиниране с ***вградената функция*** **Sum**

```
In[3]:= Sum[k^2, {k, 1, 10}]

Out[3]= 385
```

Функционалното програмиране е по-бързо

Чрез процедурно програмиране

- с оператор Do

```
In[8]:= Do[{s = 0; Do[s = s + k^2, {k, 1, n}],
    Print["pri  ", n, "=", "  suma", "=", s]}, {n, 1, 10}]

In[19]:= Timing[Do[{s = Sum[k^2, {k, 1, n}]}, {n, 1, 1000}]]

Out[19]= {0.499203, Null}
```

- с оператор While

```
In[20]:= n = 1; While[n ≤ 10,
    {s = Sum[k^2, {k, 1, n}]; Print["pri  ", n, "=", "  suma", "=", s], n++}]

In[24]:= Timing[n = 1; While[n ≤ 1000, {s = Sum[k^2, {k, 1, n}], n++}]]

Out[24]= {0.468003, Null}
```

- с оператор For

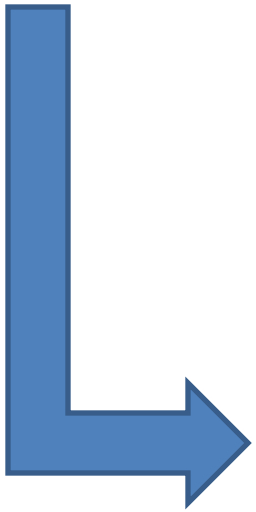
```
In[21]:= For[n = 1, n ≤ 10, n++, s = Sum[k^2, {k, 1, n}];
    Print["pri  ", n, "=", "  suma", "=", s]]

In[23]:= Timing[For[n = 1, n ≤ 1000, n++, s = Sum[k^2, {k, 1, n}]]]

Out[23]= {0.499203, Null}
```

Контролни структури

Най простата е If.

Условни разклонения (If)

If[условие, част1 , част2] ако условието е True, изпълнява се част1, в противен случай-изпълнява се част 2



Пример: Сумата на две естествени числа 5432, а тяхното най-малко общо кратно е 223020. Намерете числата.

```
In[5]:= Do[If[LCM[i, 5432 - i] == 223 020, Print[i, ", ", 5432 - i]], {i, 1, 2718}]
1652, 3780
```

Тялото на цикъла се състои от оператора If

Функцията LCM[n,m] дава най малкото общо кратно на двете числа n, m

Зареждане на пакети

Пакетите в *Mathematica са файл, който активира допълнителни команди които обикновено не са възможни. Когато се зареди пакета, тези команди стават възможни за използване*

За зареждането на пакет се използва командата **Needs**. Ако искате да използвате пакет, независимо дали е стандартен пакет или е свален от Интернет, той трябва да е в Applications. Например, има стандартен пакет с **Units,** който позволява лесно да се превръща от една мерна в друга мерна система. За да го заредим, въвеждаме

```
In[23]:= Needs["Units`"]
```

КРАЙ

Аргументът на **Needs** е стринг, който е ограден с кавички. **Забележете, че след името на пакета, преди кавичките има обратен акцент `.** След въвеждането на клетката, няма да има изход. Ако се появи съобщение за грешка, това означава, че не е напечатано правилно.