





Цикъл For

Куала Лумпур, Малайзия

По време на пътуването ние определено напредваме в знанията си! Следващата ни дестинация е Куала Лумпур! Това е привлекателно място, разположено на хълмовете в южно-малайския полуостров. Имах неудоволствието да се изгубя там! След малко ще направите едно малко кръгче!



На този етап от пътуването ни ще тръгнем в съвсем различна посока. Вече знаете как да пишете код, който взима решения. Сега ще ви покажа как да пишете код, който се изпълнява циклично. Циклите се използват за многократно изпълнение на кода. Като програмисти вие трябва да знаете как да дефинирате цикли, тъй като са те са друга основно средство в езиците за програмиране.



Цикли

През живота си ние повтаряме изпълнението на много неща. Спомням си, когато боядисвах мотоциклета си. Нанесох три пласта боя. Свърших доста работа, но резултатът си заслужаваше. След това майка ми ми напомни да изхвърля боклуците поне пет пъти преди да го направя.

Когато пишете код се случват подобни неща. Понякога програмата ви трябва да повтаря определен брой пъти изпълнението на някои неща. Ето защо всички езици за програмиране притежават цикли. Кодът в цикъла се изпълнява многократно. Циклите са много удобни. Те намаляват кода, който трябва да пишете. Използвайки ги, вие пишете определен код само веднъж, след което той може да се изпълни няколко пъти. Това значително намалява шанса да допуснете грешка.

Ето ви пример, който показва колко полезни са циклите. Представете си, че искате да съберете всички числа между 1 и 100. Можете да направите това като напишете дълъг израз. Например,

```
Dim TotalCount As Integer
TotalCount = 0
TotalCount = 1 + 2 + 3 + 4 'etc.
```

Изглежда досадно! Какво ще кажете за това?

```
Dim TotalCount As Integer
TotalCount = 0
TotalCount = TotalCount + 1
TotalCount = TotalCount + 1
TotalCount = TotalCount + 1
TotalCount = TotalCount + 1
'и т.н.
```

Все още в кода се наблюдава повтораемост. Един из същ ред трябва да се повтори 100 пъти!

А ако трябва да съберете всички числа от 1 до 1,000? Как ще го направите?

For...Next



Тук се появяват циклите. Както споменахме по-рано всички езици за програмиране притежават цикли, които позволяват кодът да се изпълнява многократно. В повечето случаи знаете колко пъти трябва да се изпълни кода. В примера за събирането на числата от 1 до 1,000 кодът трябва да се изпълни 1,000 пъти. Когато знаете колко пъти трябва да се изпълни даден код, вие използвате т.нар. "определени цикли". Те са наречени така, защото броят на изпълненията е точно определен. С тях казвате на кода колко пъти да повтори изпълнението си.

Във Visual Basic определен цикъл се дефинира с конструкцията For...Next. Кодът, който трябва да се изпълни

многократно, се поставя в оператора For...Next. Ето и синтаксиса на цикъла във Visual Basic:

```
Dim LoopCounter As Integer
```

```
For LoopCounter = LowerValue To UpperValue  
    statement1  
    statement2  
    statement etc.
```

```
Next
```

Забележете, че "For", "Next" и "To" са ключови думи. Те са оцветени в синьо и започват с главна буква. LoopCounter е променлива, която трябва да декларирате от тип Integer, преди да бъде използвана в цикъла. (Не е задължително името ѝ да бъде LoopCounter.) Променливата LoopCounter съхранява броя на изпълненията в цикъла. Нейната стойност е достъпна в цикъла.



LowerValue и UpperValue трябва да бъдат целочислени константи, променливи от тип Integer или изрази, които дават целочислен резултат като $(3 + 1 \text{ or } X + 1)$. LowerValue е инициализираща стойност за променливата LoopCounter. UpperValue е крайната стойност за променливата LoopCounter. Всеки път, когато цикълът се изпълнява стойността на променливата LoopCounter се увеличава с единица. Когато тази стойност стане по-голяма от UpperValue, цикълът спира.

Нека да напишем някакъв код, който използва определен цикъл, конструиран с оператора For...Next. Като начало ще направим цикъл, който се изпълнява 2 пъти и показва стойността на брояча на всяка итерация. Създайте ново Windows приложение с име ForNextTwo. Върху формата добавете бутон. Щракнете двукратно върху бутона, за да създадете обработчик на събитието Button1_Click. Добавете следния код:

```
Dim LoopCounter As Integer
For LoopCounter = 1 To 2

    MessageBox.Show (LoopCounter)
Next
```

Полезен съвет

Най-често учениците използват цели числа за начало и край на цикъла. Те забравят, че числата 1 или 2 от примера могат да бъдат променливи като например X.

Изградете и стартирайте проекта. Натиснете бутона. Диалогов прозорец визуализира съобщение "1", което е текущата стойност на брояча LoopCounter. Натиснете бутона ОК в прозореца. Друг диалогов прозорец визуализира съобщение "2", което отново е текущата стойност на брояча LoopCounter. Натиснете бутона ОК. Цикълът приключва изпълнението си и повече съобщения не се показват.

Ето какво се случва. В кода декларирахте променлива с име LoopCounter от тип Integer. След ключовата дума For инициализирахте променливата LoopCounter с 1 и ограничихте нарастването ѝ до стойност 2. Изразът, който трябва да се изпълни поставихте между ключовите думи For и Next. Всеки път, когато цикълът се изпълнява, стойността на променливата LoopCounter нараства с единица. Диалоговият прозорец показва текущите стойности на променливата LoopCounter (1, 2). Когато стойността на променливата LoopCounter стане 3, цикълът приключва и диалоговият прозорец не се визуализира повече!

Полезен съвет

Когато цикълът For приключи, броячът има стойност, която е с 1 по-голяма от горната граница.



Дестинация
Куала Лумпур, Малайзия



Пейджър

По подразбиране броячът на цикъла се инкрементира с единица на всяка итерация. Можете да инкрементирате брояча на цикъла с по-голяма стойност като използвате ключовата дума **Step**, както е показано в кода по-долу:

```
Dim LoopCounter As Integer  
For LoopCounter = 0 To 100  
Step 5  
    MessageBox.Show  
    (LoopCounter)  
Next
```

Тук броячът LoopCounter се инкрементира с 5 на всяка итерация. Диалоговият прозорец показва 0, 5, 10, 15, и т.н.



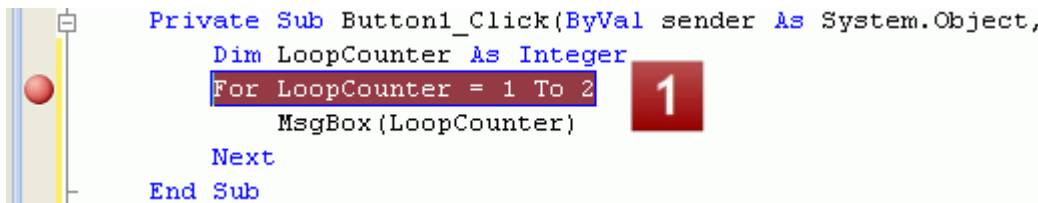
Постъпково изпълнение на цикъла For...Next

Нека да изпълним кода, който написахте като използваме средствата за дебъгване във Visual Studio. Изпълнението ще проследим стъпка по стъпка. Следвайте инструкциите:

Полезен съвет

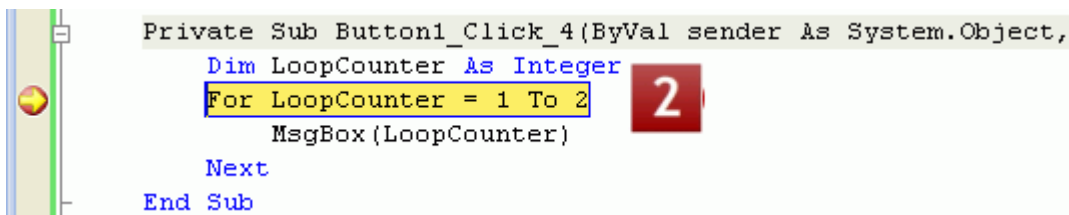
Накарайте учениците да направят това, а не само да го прочетат.

1. Поставете точка за спиране на реда, който започва с ключовата дума For. Изградете и стартирайте приложението.



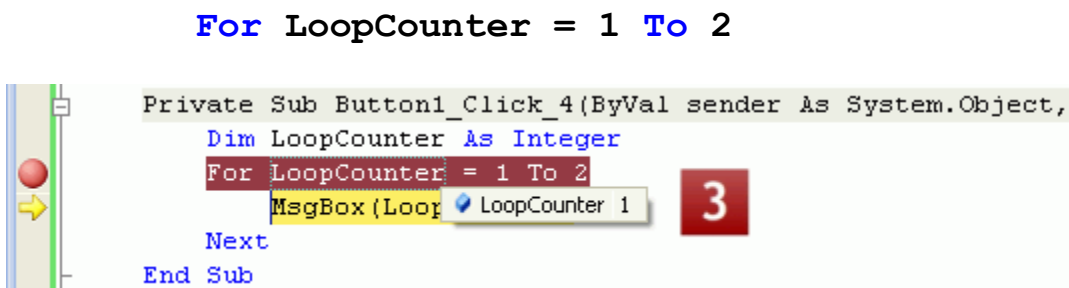
```
Private Sub Button1_Click(ByVal sender As System.Object,
    Dim LoopCounter As Integer
    For LoopCounter = 1 To 2
        MsgBox (LoopCounter)
    Next
End Sub
```

2. Формата се показва. Натиснете бутона. Редът с точката за спиране се маркира с жълт цвят.



```
Private Sub Button1_Click_4(ByVal sender As System.Object,
    Dim LoopCounter As Integer
    For LoopCounter = 1 To 2
        MsgBox (LoopCounter)
    Next
End Sub
```

3. Натиснете клавиша F11. Маркираният ред се изпълнява.

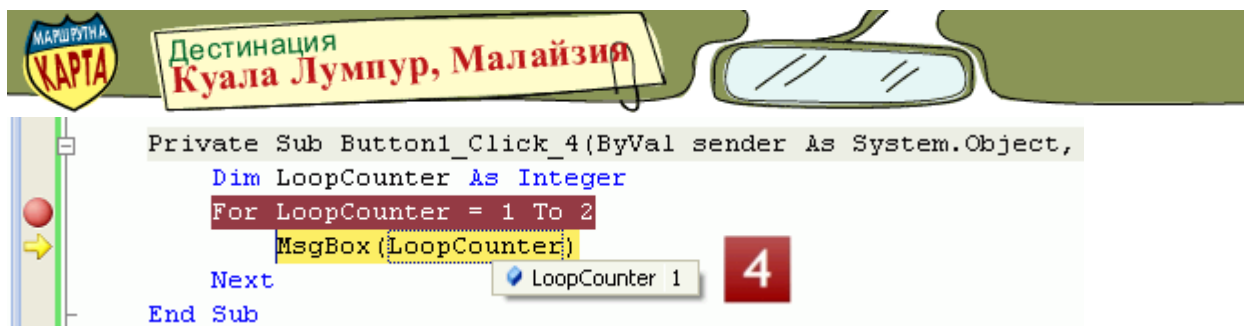


```
Private Sub Button1_Click_4(ByVal sender As System.Object,
    Dim LoopCounter As Integer
    For LoopCounter = 1 To 2
        MsgBox (LoopCounter)
    Next
End Sub
```

Преместете курсора на мишката над променливата LoopCounter. Стойността ѝ е 1.

4. Натиснете клавиша F11. Маркираният ред се изпълнява.

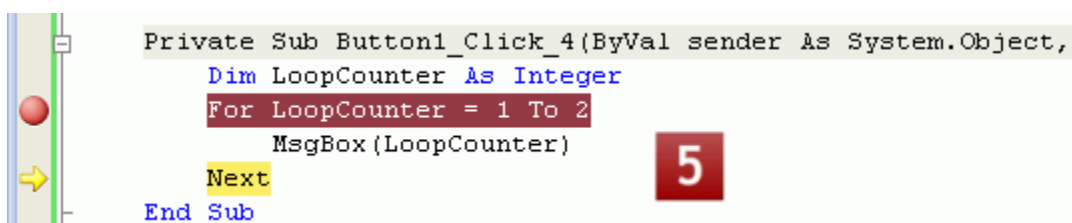
MessageBox.Show (LoopCounter)



Диалоговият прозорец показва 1. Натиснете бутона ОК, за да затворите прозореца.

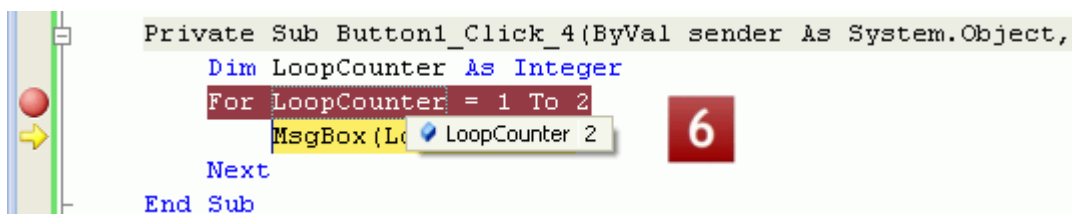
5. Натиснете клавиша F11. Маркираният ред се изпълнява.

Next



6. Изпълнява се и реда с ключовата дума For.

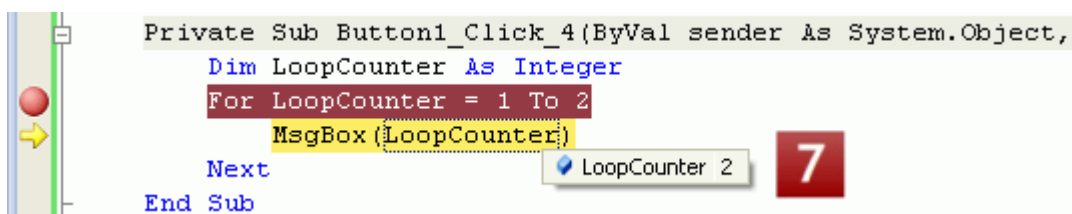
For LoopCounter = 1 To 2



Преместете курсора на мишката над променливата LoopCounter. Стойността ѝ е 2.

7. Натиснете клавиша F11. Маркираният ред се изпълнява.

MessageBox.Show(LoopCounter)

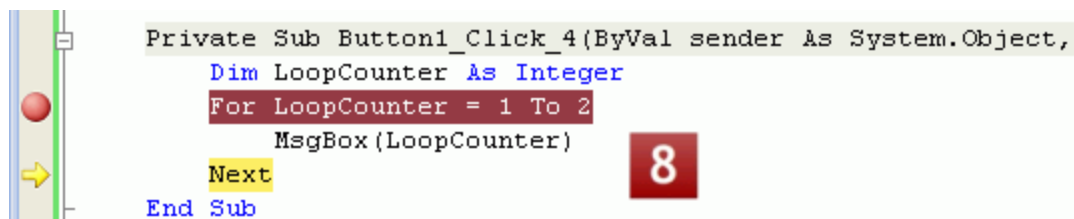




Диалоговият прозорец показва 2. Натиснете бутона ОК, за да затворите прозореца.

8. Натиснете клавиша F11. Маркираният ред се изпълнява.

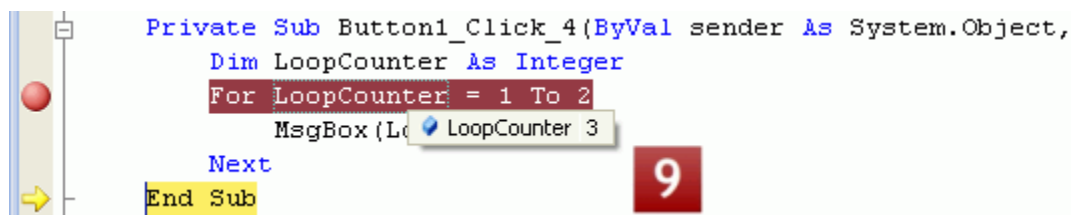
Next



```
Private Sub Button1_Click_4(ByVal sender As System.Object,  
    Dim LoopCounter As Integer  
    For LoopCounter = 1 To 2  
        MsgBox(LoopCounter)  
    Next  
End Sub
```

9. Изпълнява се и реда с ключовата дума.

For LoopCounter = 1 To 2

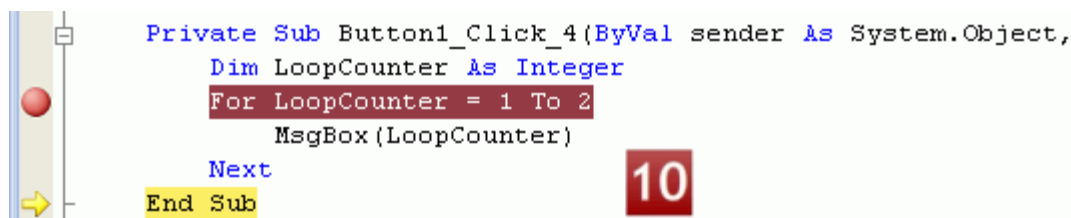


```
Private Sub Button1_Click_4(ByVal sender As System.Object,  
    Dim LoopCounter As Integer  
    For LoopCounter = 1 To 2  
        MsgBox(LoopCounter)  
    Next  
End Sub
```

Преместете курсора на мишката над променливата LoopCounter. Стойността ѝ е 3.

10. Натиснете клавиша F11. Маркираният ред се изпълнява.

End Sub



```
Private Sub Button1_Click_4(ByVal sender As System.Object,  
    Dim LoopCounter As Integer  
    For LoopCounter = 1 To 2  
        MsgBox(LoopCounter)  
    Next  
End Sub
```

Формата се появява отново.

Използвахте средствата за дебъгване в цикъла For...Next. Сега вече наистина видяхте как протича изпълнението на цикъла от гледна точка на програмата!



Сумиране на числата до 1000

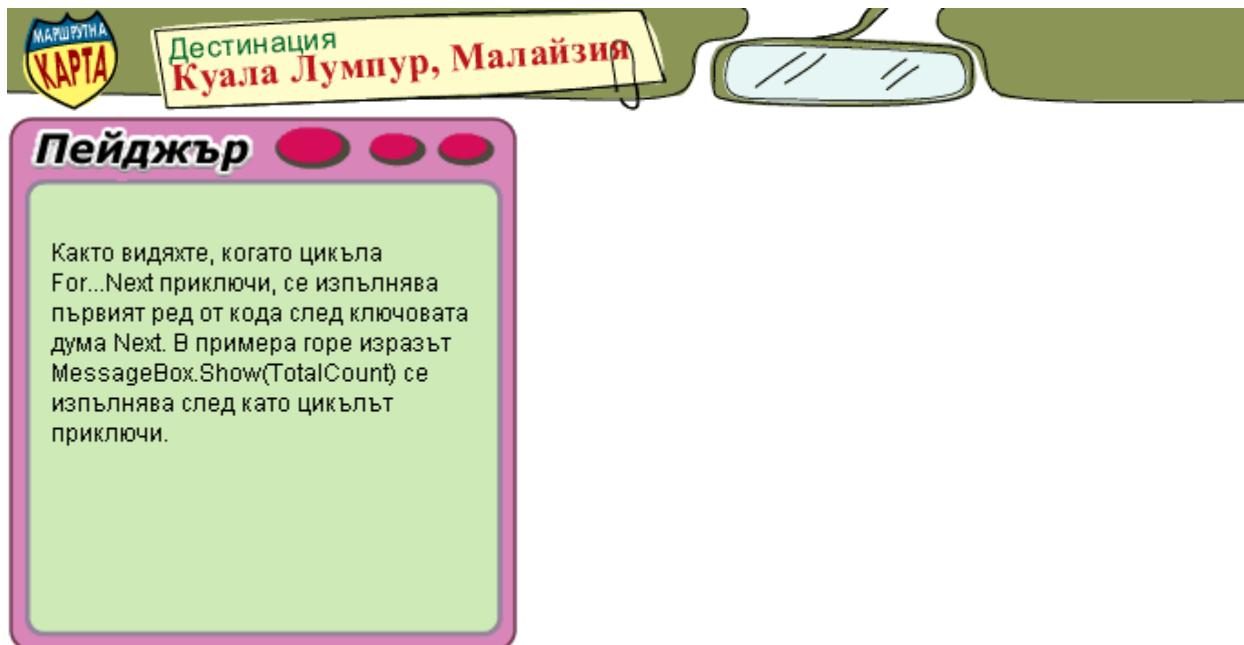
Имам идея! Нека да напишем код, който събира числата от 1 и 1,000! Готови ли сте? С цикъла For...Next е лесно. Създайте ново Windows приложение с име AddUp. Върху формата добавете бутон. Задайте стойност "Add" за свойството Text на бутона. Щракнете двукратно върху него, за да създадете обработчик на събитието Button1_Click. Добавете следния код:

```
Dim LoopCounter As Integer
Dim TotalCount As Integer = 0

For LoopCounter = 1 To 1000
    TotalCount = TotalCount + LoopCounter
Next
MessageBox.Show(TotalCount)
```

Изградете и стартирайте приложението. Натиснете бутона "Add". Диалоговият прозорец показва крайният резултат от събирането на числата от 1 до 1,000. Не беше ли бързо? Написахте кода, а програмата изчисли резултата по-бързо отколкото вие можете да го направите с калкулатор!

Как работи кода? Първо декларирахме две целочислени променливи: LoopCounter и TotalCount. Променливата LoopCounter определя броя на изпълненията на цикъла. Променливата TotalCount съхранява получената междина сума. След ключовата дума For променливата LoopCounter е инициализирана с 1, а нарастването на стойността ѝ е ограничено до 1000. Ето защо цикълът се изпълнява 1000 пъти, събирайки числата от 1 до 1000. Изразът, чието изчисление се повтаря, събира текущата стойност на променливата TotalCount с текущата стойност на променливата LoopCounter и съхранява резултата в променливата TotalCount. При всяко изпълнение на цикъла стойността на променливата LoopCounter се добавя към стойността на променливата TotalCount (0+1=1, 1+2=3, 3+3=6, 6+4=10, etc.) . Когато променливата LoopCounter стане 1000, цикълът приключва и се изпълнява редът след ключовата дума Next. Тогава диалоговият прозорец показва стойността на променливата TotalCount.





Изхвърли боклука

Сега нека да разгледаме още няколко примера, които използват определен цикъл, за да реализират нещо различно от математически изчисления. Спомняте ли си, че в цикъла For...Next е възможно да поставите всякакъв код. Можете автоматично да конструирате дълги символни низове, да показвате досадни съобщения многократно или да променяте свойства въз основа на брояча в цикъла. Пробвайте това различно приложение на циклите.



Стартирайте ново Windows приложение с име GarbageOut. Върху формата добавете бутон. Задайте стойност "Mom Says" за свойството Text на бутона. Щракнете двукратно върху него, за да създадете обработчик на събитието Button1_Click. Добавете следния код:

```
Dim i As Integer = 0
MomMessage = "Please, Take Out The Garbage."
For i = 1 To 5
    MessageBox.Show(MomMessage)
    MomMessage = MomMessage & vbCrLf & MomMessage
Next Dim MomMessage As String
```

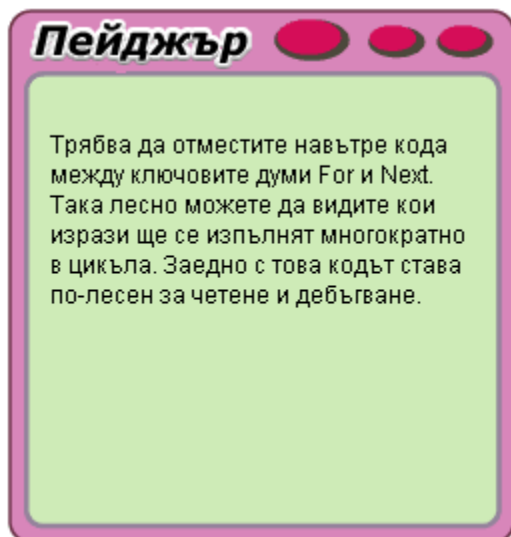
Изградете и стартирайте проекта. Натиснете бутона "Mom Says". Показва се съобщение с думите на майка ми. Натиснете отново бутона "Mom Says". Забележете как съобщението нараства. Добре, добре. Разбрах! Циклите предоставят механизъм за сливане на символни низове.



Ето друг интересен пример. Добавете върху формата друг бутон. Задайте стойност "Color Again" за свойството Text на бутона. Щракнете двукратно върху него, за да създадете обработчик на събитието Button2_Click. Добавете следния код:

```
Dim i As Integer = 0
For i = 2 To 6
    If i < 3 Or i > 5 Then
        MessageBox.Show(i)
        Form1.ActiveForm.BackColor = System.Drawing.Color.Red
    Else
        MessageBox.Show(i)
        Form1.ActiveForm.BackColor =
System.Drawing.Color.Blue
    End If
Next
```

Изградете и стартирайте проекта. Натиснете бутона "Color Again". Диалогов прозорец показва текущата стойност на брояча в цикъла. Забележете, че този път първоначалната стойност на брояча е 2, а крайната – 6. Операторът If...Then...Else в цикъла определя в зависимост от стойността на брояча как да се оцвети формата. Доста ловко, нали? Поставих диалогов прозорец, за да виждате стойността на брояча и за да проследите промяната на цвета на формата.





10 Пробвайте 01 Преброяване на зайци

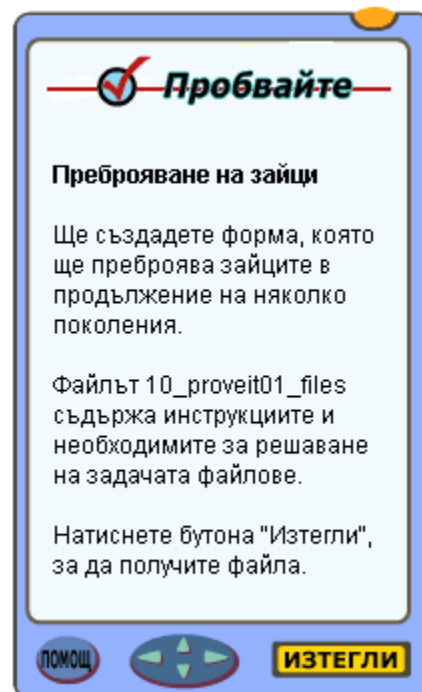
Погледнете навън! Прекосявайки пътя се натъкнахме на голям заек! Допускате ли, че заекът мисли, че може да живее вечно? Хрумва ми идея. Да предположим, че зайците живеят вечно и са на остров с много растителност и без хищници. Да предположим също така, че на острова първоначално има два заека и всяка двойка зайци има поколение от четири зайчета. След едно поколение зайците ще бъдат 6 ($2 + 4 = 6$), а след две – 18 ($6 + 12$).

Как ще нараства популацията на зайците с всяко поколение?

Създайте форма подобна на показаната по-долу:

Визуализирайте номера на поколението, броя на зайците, и броя на зайците на квадратен метър, имайки предвид, че островът е един квадратен километър.

Позволете на потребителя да избере броя на поколенията.





За да направите изходните данни лесни за четене, използвайте константите **vbTab** и **vbNewLine** на Visual Basic:

```
VariableX = Format(VariableX, "###, ###, ###, ###,  
##0")
```

или

```
VariableY = Format(VariableY, "###, ##0.0000")
```

Ако програмата работи правилно, я покажете на учителя си.



Вложени цикли

Да продължим с циклите! Знаете ли, че можете да поставите един цикъл в друг? Вътрешният цикъл е кодът, който се изпълнява на всяка итерация от външния цикъл. Тогава колко пъти се изпълнява вътрешния цикъл? Разгледайте примера, за да отговорите на въпроса.

Създайте ново Windows приложение с име LoopTheLoop. Добавете върху формата бутон. Щракнете двукратно върху бутона, за да създадете обработчик на събитието Button2_Click. Добавете следния код:

```
Dim OuterLimit As Integer =  
3  
Dim InnerLimit As Integer =  
4  
Dim OuterCounter As Integer  
Dim OnnerCounter As Integer  
Dim Total As Integer = 0
```

Полезен съвет

Попитайте предварително учениците каква ще бъде стойността на променливата Total, докато още не са започнали да програмират.

```
For OuterCounter = 1 To OuterLimit  
    For InnerCounter = 1 To InnerLimit  
        Total = Total + 1  
    Next  
Next  
MessageBox.Show("Total= " & Total)
```

Изградете и стартирайте проекта. Натиснете бутона. Диалоговият прозорец показва "Total= 12". Това е броят на изпълненията на вътрешния цикъл. Погледнете отново кода. Забелязахте ли, че 12 е равно на 3 (броя на изпълненията на външния цикъл) умножено по 4 (броя на изпълненията на вътрешния цикъл)? Точно така! Всеки път, когато външният цикъл се изпълнява, вътрешният цикъл се изпълнява 4 пъти. Тъй като външният цикъл се изпълнява 3 пъти, то вътрешният цикъл се изпълнява $4 \times 3 = 12$ пъти.

Забележете, че използвахме две променливи OuterLimit и InnerLimit, за да ограничим броя на изпълненията съответно на външния и вътрешния цикъл. Стойностите, които определят горната и долната граница на броячите, не е задължително да бъдат целочислени константи. Те могат да бъдат променливи или изрази, които дават целочислен резултат. Забележете също така, че вътрешният цикъл е отместен навътре в кода.



Знакът Exit



Време е да прекъснем за малко дългото си пътуване и да направим почивка. Трябва да стигнем до знак за изход и да се отклоним от главния път. Мисля, че това понякога се случва и в кода. Изпълнението му също може да се прекъсне. Представете си, че събирате числата от 1 до 1000! Аз бих се уморил също!

Понякога може да се наложи да спрете изпълнението на цикъла преди броячът да е достигнал крайната си стойност. Това става посредством израза "Exit For". Той се поставя в оператора If...Then в тялото на цикъла. Условието на оператора If...Then определя кога изпълнението на цикъла ще се

прекрати. Нека да напишем малко код, за да демонстрираме как става това.

Създайте ново Windows приложение с име ExitSign. Върху формата добавете бутон. Задайте стойност " ExitNow " за свойството Text на бутона. Щракнете двукратно върху него, за да създадете обработчик на събитието Button1_Click. Добавете следния код:

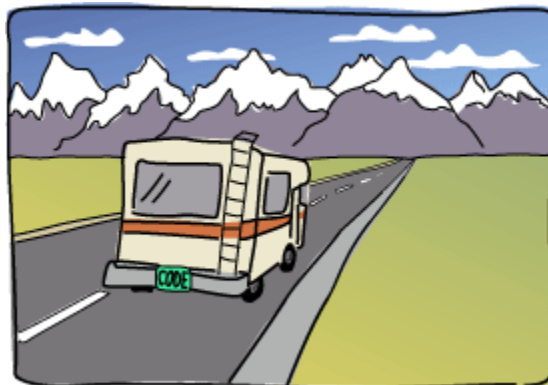
```
Dim i As Integer = 0
For i = 1 To 5
    MessageBox.Show("i inside= " & i)
    If i = 3 Then
        Exit For
    End If
Next
MessageBox.Show("i outside= " & i)
```

Полезен съвет

Операторът Exit се използва в много случаи: Exit For, Exit Do, Exit Sub.....



Изградете и стартирайте проекта. Натиснете бутона "Exit Now" няколко пъти. Диалогов прозорец показва стойността на брояча в цикъла. Когато тя стане 3, условието на оператора If...Then се удовлетворява и изразът Exit For се изпълнява. Цикълът се прекратява без да са изпълнени всички итерации. Забележете, че диалоговият прозорец никога не показва съобщение "i inside = 4". Когато цикълът приключи, кодът след ключовата дума Next се изпълнява. Тогава диалоговият прозорец показва съобщение "i outside = 3".



Ето друг пример. Ще добавим още малко функционалност към приложението Exit Sign. Върху формата добавете поле за отметка и втори бутон. Задайте стойност "ChexIt" за свойството Text на бутона. Щракнете двукратно върху него, за да създадете обработчик на събитието Button2_Click. Добавете следния код:

```
Dim LoopIndex As Integer
For LoopIndex = 1 To 5
    If LoopIndex = 4 Then
        CheckBox1.Checked = True
    End If
    If CheckBox1.Checked = True Then
        MessageBox.Show("Exit on: " & LoopIndex)
        Exit For
    End If
Next
```

Изградете и стартирайте проекта. Натиснете бутона "ChexIt". Цикълът For...Next се изпълнява докато променливата LoopIndex не стане равна на 4. След това се проверява стойността на свойството Checked на полето за отметка. Тъй като това поле е маркирано, диалогов прозорец показва стойността на променливата LoopIndex и се изпълнява реда Exit For. Цикълът приключва.

Натиснете отново бутона "ChexIt", докато полето за отметка все още е маркирано. Какво се случва? Редът Exit For се изпълнява, когато LoopIndex = 1.



Цикълът For...Next в C#

Както всички съвременни езици за програмиране C# също поддържа определени цикли! Може би Джин и Дърк могат да ни покажат как да използваме циклите в тези езици? Какво ще кажете за това?



C# има еквивалентен на Visual Basic оператор за реализиране на цикъл, който обаче изглежда малко по-различно. Ето как се създава определен цикъл в C#:

```
for (int LoopCounter = 1; LoopCounter < 5;
LoopCounter++)
{
    MessageBox.Show(LoopCounter.ToString());
}
```

Кодът изглежда различно в сравнение с разгледания преди това, но работи по същия начин.



В C# не се използва ключова дума Next, а ключовата дума for се пише с малки букви. Кодът, който трябва да се изпълнява в цикъла, се загражда с фигурни скоби. Кодът в кръглите скоби контролира изпълнението на цикъла:

```
int LoopCounter = 1;
```

Декларира се брояча на цикъла и го инициализира.

```
LoopCounter < 4;
```

Задава се крайната стойност на брояча, при достигането на която цикълът спира.

```
LoopCounter++
```

Инкрементира се броячът с 1.

Това може да бъде написано и по следния начин

```
LoopCounter = LoopCounter + 1
```

Всичко това е само в един ред от кода!





Ето и код, написан на C#. Изглежда по същия начин с изключение на това, че използва функцията `System.Convert.ToString`, за да преобразува стойността на брояча в символна стойност, за да я покаже в диалогов прозорец.

```
for (int LoopCounter = 1; LoopCounter < 5;
LoopCounter++)
{
    MessageBox.Show(System.Convert.ToString(LoopCounter));
}
```

Благодаря ти, Джин. Дори и да изглежда по различен начин в сравнение с Visual Basic, операторът за цикъл има същото приложение – многократно изпълнение на един и същ код.



Пристигане в Куала Лумпур, Малайзия

Като че ли пътуването беше кратко! Хубаво е, че не се загубихме в пустинята. Можехме още да се движим в кръг! По пътя ви показах как да изпълнявате многократно код като използвате оператора For...Next. Този оператор е подходящ, когато знаете броя на повторенията в тялото на цикъла. Показах ви също и как да използвате израза Exit For, за да прекъснете изпълнението на цикъла.

Сега вече усвоихте с още една програмна техника. Проверете дали можете да приложите това, което сте научили, в следващата задача. Сигурен съм, че Комисията по състезанията има за вас ново предизвикателство. Не забравяйте теста!



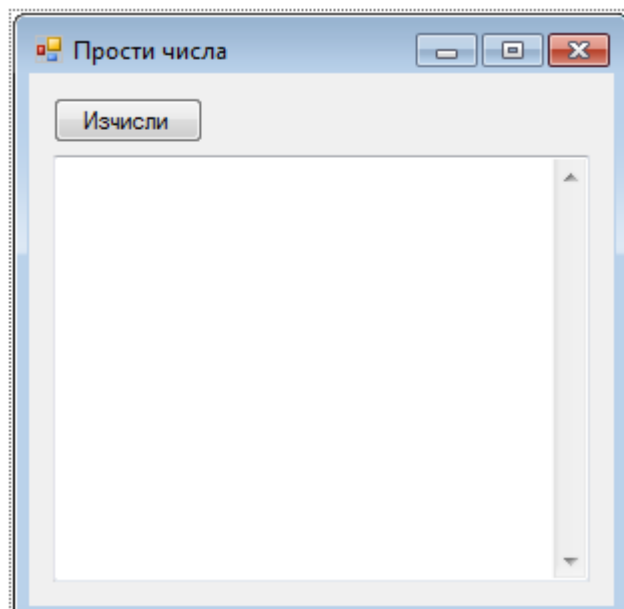
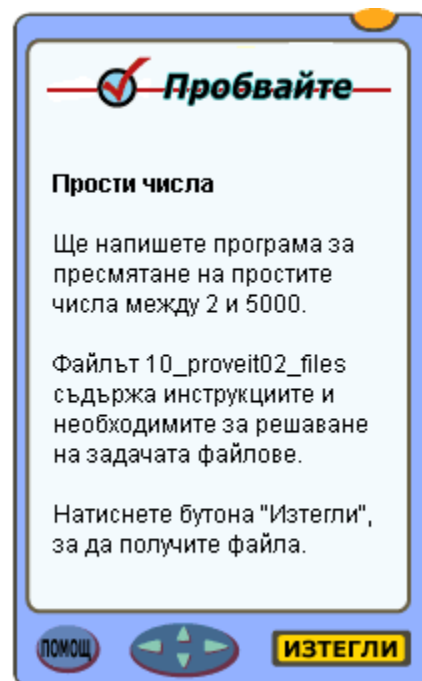
10 Пробвайте 02 Прости числа

Тази вечер очаквам с нетърпение крехките ребърца вместо обичайната пица. Крехки ребърца! Вкусно!

Говорейки за крехките, неделими ребърца, си спомних за простите числа, които изучавахме в клас. Просто число е това число, което се дели без остатък само на себе си и 1. Например: 2, 3, 5, 7. Числата 4, 6, 8, и 9 не са прости, защото се делят на две, а 9 респективно на 3.

Можете ли да напишете програма, която изчислява простите числа от 2 до 5000?

Създайте форма подобна на показаната по-долу:



Изчислете простите числа до 5000 при натискането на бутона "Изчисли". Добавете ги в текстово поле.



Ето и някои съвети, които могат да ви бъдат полезни.

Запознавайки се с материалите от курса по програмиране, вие научихте, че можете да влагате един в друг блокове с изходен код, например няколко условни оператора **If**. Също така можете да влагате и оператори за цикъл **For**, като единия цикъл е вътрешен, а другия външен. Важно е да внимавате със затварянето на циклите – първо се затваря вътрешния, а след това външния цикъл. Използвайте външен цикъл, който започва от 2 и приключва с 5000.

Променливата от външния цикъл е числото, което ще се проверява дали е просто. Вътрешния цикъл трябва да започва от 2 и да приключва с текущата стойност на външния цикъл намалена с 1. Защо това трябва да се направи така? Защото променливата от вътрешния цикъл не трябва да става по-голяма от променливата във външния цикъл. Защо минус 1? Защото простите числа могат да се делят сами на себе си и следователно не е нужно това да се проверява.

Ще ви бъде необходима променлива – флаг, която ще обозначава дали стойността на променливата от външния цикъл е просто число или не. Тя трябва да приеме стойност **True** преди влизането във вътрешния цикъл и да се установи в състояние **False**, ако условието за проверка дали дадено число е просто не е изпълнено.

Във втория цикъл трябва да проверите дали съществува остатък при деление на съответното число с променливата, управляваща цикъла.

Да разгледаме числата X и Y . Как ще установите, че Y дели X без остатък? Съществуват няколко начина. Тъй като проверяваме вашите умения в конструирането на цикли, а не знанията ви по математика, ще ви дадем идея. Най-лесният начин е да използвате оператора **Mod**. Предположете, че сте дефинирали променлива с име **Remainder**, която ще приема остатъка от делението на X и Y . Следователно:

$$\text{Remainder} = X \text{ Mod } Y$$

След като приключи вътрешния цикъл вие трябва да добавите съответното число в текстовото поле, ако стойността на променливата – флаг е **True**. Може би ще е необходимо по някакъв начин да отделяте визуално простите числа в текстовото поле със запетая, интервал или по някакъв друг, избран от вас начин..

Ако програмата работи правилно, я покажете на вашия учител.



Ето и предизвикателството на Комисията по състезанията.

10 Можете ли? Калкулатор на Рот

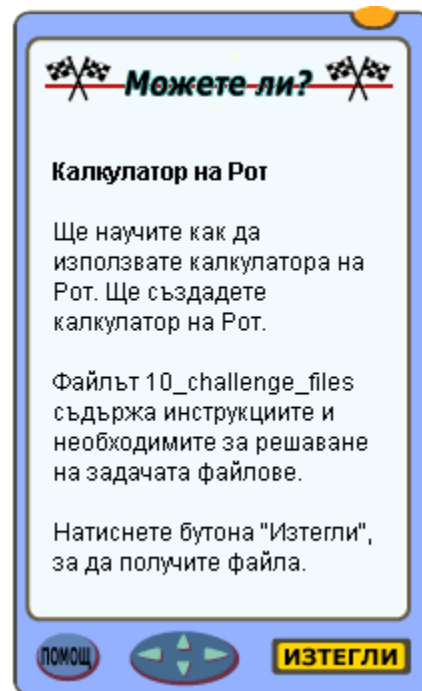
Моето желание е да работя дълго време като програмист, но един ден знам, че ще трябва да се пенсионирам. Мислех си за времето, когато това ще се случи, и знам, че ако започна да спестявам като млад, по-късно ще разполагам със значителни средства.

Полезен съвет

Вижте полезния съвет от последната страница.....

Чувал съм за личната пенсионна сметка на Рот. Рот е човекът, който се грижи за сметката на Конгреса. Той определя влоговете по отношение на по-особени данъци. Вие никога не сте плащали данък върху приходите си, т.е. запазвате всичко, което сте спечелили. Можете да спестявате пари докато навършите 59 години и половина или да използвате спестяванията си за да завършите образованието си или да закупите първия си дом.

Нека да създадем калкулатор на Рот.
Създайте форма подобна на показаната по-долу:





Във формата се въвеждат възраст, от която започвате да спестявате, възраст, от която спирате, годишен депозит, и процент на заинтересованост, който е цяло число.

След като вече изучихме циклите имаме и подходяща възможност да ги използваме. Стойностите за началната и крайната възраст могат да се използват за дефиниране на цикъла. Не забравяйте, че текстовите полета получават символни стойности така, че трябва да използвате функцията `Val` за да ги преобразувате в числа по следния начин

```
X = Val (TextBox1.Text) .
```

Визуализирайте възрастта, общата стойност на депозитите от началото на спестяването, баланса на сметката и печалбата, която е разлика между депозитите и баланса.

За да форматираме изхода, използвайте функцията `Format`.
Например:

```
VariableX = FormatCurrency(VariableX)
```

Съвет 1: Преди всяко изчисление добавяйте един депозит.

Съвет 2: Печалбата се определя от баланса, който е натрупания депозит умножен по $(1 + \text{процента})$, т.е. 9% са 1.09.

Ако програмата работи правилно, я покажете на учителя си.

Продължение / Обобщение

Какво ще стане, ако правите същите влогове в обикновена банкова сметка? При обикновените банкови сметки правителството получава част от приходите ви. Предположете, че се отдържат по 25% от приходите ви всяка година. С каква сума ще разполагате в края на периода, в който спестявате?



Дестинация
Куала Лумпур, Малайзия



Проверка на знанията

НАПРАВЕТЕ ТЕСТА ОТНОВО

- 1** Какво правят циклите?
- ☐ A. Връщат към началото на блок с код
 - ☐ B. Предизвикват изход от програмата
 - ☐ C. Изпълняват многократно код
- 2** Какво правят определените цикли?
- ☐ A. Взимат многократно решения
 - ☐ B. Изпълняват редове с код определен брой пъти
 - ☐ C. Дефинират променливи
- 3** Какъв тип променлива е броячът в цикъла For?
- ☐ A. Single
 - ☐ B. Integer
 - ☐ C. String
- 4** В израза:
`For LoopCounter = 1 To 2`, каква е стойността на LoopCounter, когато цикълът приключи?
- ☐ A. 1
 - ☐ B. 2
 - ☐ C. 3



Полезен съвет

Обикновено това е много труден проект за учениците, не от гледна точка на програмната реализация, а поради това, че лихвените проценти и данъците са нещо ново за тях. Ето един анализ, който е използван в други класове:

Ваша цел е да увеличите банковата си сметка, която първоначално е нулева.

Това е направено по два начина. Единият е вие да внесете пари в банковата сметка. Вторият е банката да добавя пари към банковата ви сметка като компенсация, че работи с парите ви.

В проекта предполагаме, че вие внасяте пари веднъж годишно и банката ви плаща също веднъж годишно. Ще използвате цикъл For, с който ще внасяте пари в банката и ще им начислявате лихва определен брой години.

В цикъла ще правите две неща.

Първо, ще добавяте пари.

Второ, банката ще добавя пари, увеличавайки банковата ви сметка. Увеличаването се прави чрез умножение на банковата сметка с 1 плюс лихвения процент. Ако лихвения процент е 3%, то банковата сметка трябва да се умножи по 1.03.

След като цикълът приключи трябва да форматирате и визуализирате резултата в текстово поле.

На следващата стъпка трябва да сумирате всички внесени от вас суми. Внасяте пари всяка година, включително и първата. Ако предположим, че започвате да внасяте пари от 16 до 20 годишна възраст, то сте внесли суми на 16, 17, 18, 19 и 20 години. Това означава, че горната граница на цикъла се изчислява като възрастта на спиране минус възрастта на започване плюс 1.

Лихвата, която банката плаща, се изчислява като от банковата сметка се извади внесената от вас сума.

А данъците? Правителството взима част от всички пари, които сте спечелили.

За внесените пари не се плащат данъци, защото те са вече платени, когато сте получавали заплата си от работодателя.

