ЛЕКЦИИ ПО ПРОГРАМИРАЊЕ ВО VISUAL BASIC

Верзија 0.1

Никита Шекутковски **Ирена Стојковска**

СОДРЖИНА[1]

[1] Текстовите презентирани во оваа верзија се објавувани статии во математичкото списание Нумерус во годините 2006-2007 и 2007-2008

Лекција 1.
Стартување на програмскиот пакет Visual Basic 5.0

Вообичаен начин за стартување на програмскиот пакет Visual Basic 5.0 е со дупло кликање на истоимената икона на десктопот, доколку ја има.

Иконата на програмскиот пакет
Visual Basic 5.0 на десктопот

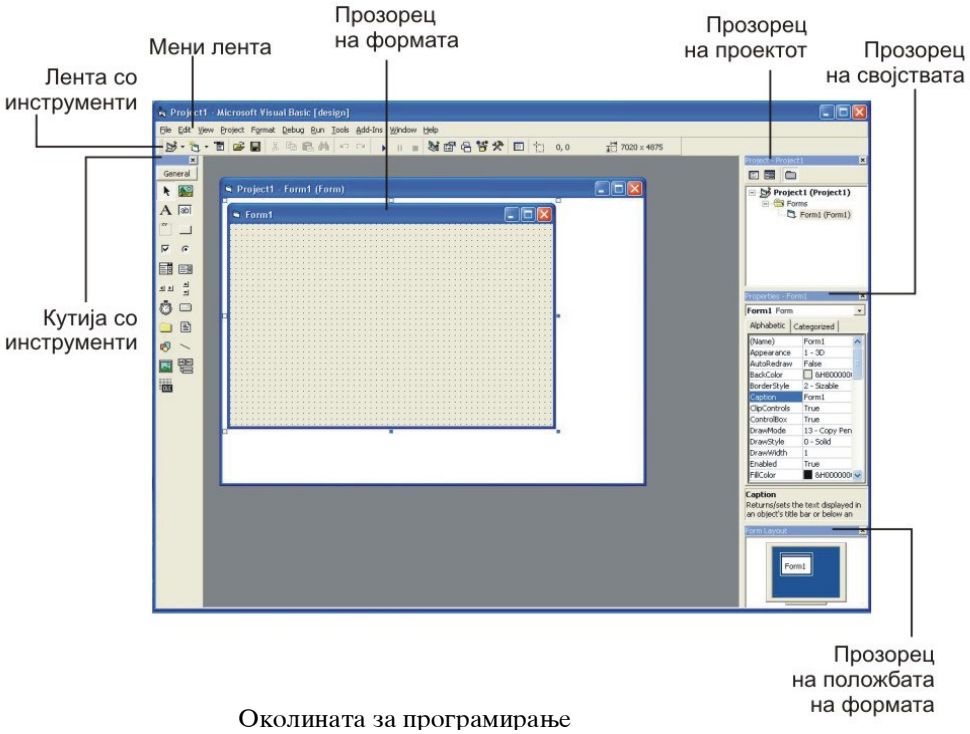
Но, исто така може да го стартувате следејќи ги следните чекори:

1 чекор. Стиснете на копчето Start 

во Microsoft Windows, одберете Programs и потоа одберете Microsoft Visual Basic 5.0 или Visual Basic 5.0 CCE. Иконите во таа папка (фолдер) ќе се појават во вид на список.

2 чекор. Кликнете на иконата Visual Basic 5.0 или Visual Basic 5.0 CCE. Програмата ќе се стартува и ќе се појави прозорецот New Project за креирање на нов ВБ-проект (така се нарекуваат програмите кои се прават со овој програмски пакет), кој прашува за типот на ВБ-проектот кој сакате да го создадете.

3 чекор. Селектирајте го стандардниот тип на ВБ-проект (Standard EXE) и потоа кликнете на копчето Open. Со тоа на екранот ќе се прикаже околината за програмирање во Visual Basic заедно со некои од прозорците и инструментите. Препорачлива е следната околина за програмирање:



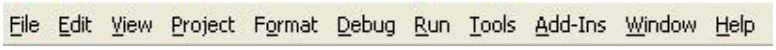
Околината за програмирање

1.1. Околината за програмирање во Visual Basic

Во овој дел ќе дадеме краток опис на околината за програмирање во Visual Basic, која се состои од сите потребни инстументи за брзо и ефикасно креирање на ВБ-проекти. Оние кои се нестпрливи да ги видат можностите на програмирањето во Visual Basic, може да преминат на следната точка, и да си ги испробаат своите програмерски квалитети. Но, доколку приметите дека потешко ви оди работата таму, тогаш сигурно ќе треба да се навратите повторно и повторно на овој дел.

Значи, составни делови на околината за програмирање се:

1. Мени лента (Menubar), која дава пристап до повеќе од командите кои помагаат при програмирањето. Менијата и командите работат на истиот начин како и во сите останати Windows-базирани програми и пристапот до нив се остварува со глушецот или преку тастатурата. Тука, ќе ги сретнете менијата File, Edit, View и.т.н.



Оваа лента секогаш е составен дел на околината за програмирање.

2. Лента со инструменти (Toolbar), која се наоѓа под мени лентата и се состои од копчиња со чија помош се извршуваат команди и се контролира изгледот на околината за програмирање.

Оваа лента не е задолжителна, затоа што ако сакате да пристапите кон некоја од командите од лентата со инструменти, тоа може да го направите и преку некое од менијата од мени лентата (ова ќе го увидите подолу). Сепак е препорачлива, па ако ја нема може да ја повикате така што од мени лентата ќе го одберете менито View, кликнете на иконата Toolbar, и одберете ја опцијата Standard.

3. Кутија со инструменти (Toolbox), која содржи копчиња со кои се вметнуваат објекти на формата, како што се слики (Image), етикети (Label), копчиња (Command Button), текстуални полиња (Text Box), ленти за скролирање (Scroll Bar) и слично.

Функцијата на секое од копчињата ќе се прикаже ако го поставите покажувачот на глушецот над соодветното копче.

За да се прикаже оваа кутија во околината за програмирање (доколку ја нема), се отвара преку лентата со инструменти со кликање на копчето Toolbox  или од мени лентата се одбира менито View и потоа се клика на иконата Toolbox.

4. ***Прозорец на формата*** (Form Window), каде се планира изгледот (интерфејсот) на ВБ-проектот со вметнување на разни објекти. Затоа, се нарекува уште и прозорец на објектите и се повикува со кликање на иконата Object од View менито.

5. ***Прозорец на проектот*** (Project Explorer Window), кој помага за пристапување кон компонентите од кои е составен еден ВБ-проект, кои се користат во процесот на програмирање, односно кон нивните делови (код и/или изглед) со копчињата View Code  и View Object  соодветно.

Еден ВБ-проект се состои од повеќе датотеки (фајлови): *датотеката на ВБ-проектот* (со наставка .vbp) која го содржи во себе списокот на сите останати датотеки во ВБ-проектот, *датотеката на форма* (со наставка .frm) која се состои од изгледот и програмскиот код, придружен на нејзините објекти и која е најзначајна да се зачува, *стандарден модул* (со наставка .bas) кој содржи декларации на типови, константи, променливи и процедури кои се заеднички за сите делови на ВБ-проектот и кој нема да го користиме во првиве лекции и други.

За да се прикаже овој прозорец во околината за програмирање (доколку го нема), се отвара преку лентата со инструменти со кликање на копчето Project Explorer  или од мени лентата се одбира менито View и потоа се клика на иконата Project Explorer.

6. *Прозорец на својствата* (Properties Window), кој се користи за менување на својствата на формата и објектите сместени на неа. За да се прикаже овој прозорец во околината за програмирање (доколку го нема), се отвара преку лентата со инструменти со кликање на копчето Properties Window  или од мени лентата се одбира менито View и потоа се клика на иконата Properties Window.

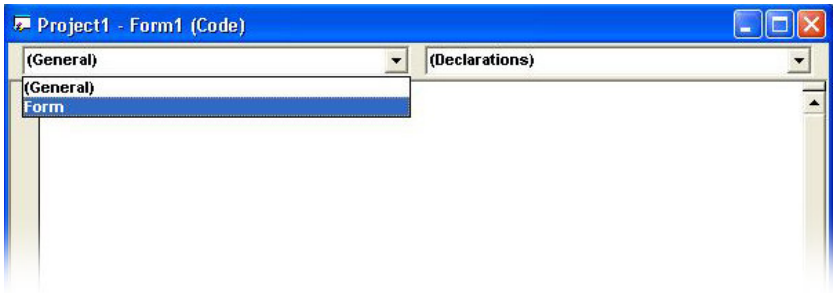
7. *Прозорец на положбата на формата* (Form Layout Window), кој ја дава местоположбата на формата на екранот за време на извршување на ВБ-проектот. За да се прикаже овој прозорец во околината за програмирање (доколку го нема), се отвара преку лентата со инструменти со кликање на копчето Form Layout Window  или од мени лентата се одбира менито View и потоа се клика на иконата Form Layout Window.

1.2. Во Visual Basic може веднаш да се програмира

Откако пред вас е препорачливата околина за програмирање, време е и да ги испробаме програмерските можности на овој програмски пакет. За таа цел ви изложуваме два ВБ-проекта, чекор по чекор, без да се задржуваме на многу детални објаснувања. За основите на програмирањето во Visual Basic (променливи, константи, операции, функции, наредби) ќе читате во некои од наредните лекции. Сега проследете ги чекориве и ве уверуваме дека ќе добиете интересни резултати:

1 чекор. Прво прикажете го прозорецот на кодот, така што ќе коликнете на копчето View Code  кое се наоѓа во горниот лев дел од прозорецот на проектот, а истото копче го има и на лентата со инструменти или со кликање на иконата Code од View менито.

2 чекор. Откако ќе се појави прозорецот на кодот, отворете го списокот на објекти (Object) со кликање на полето на кое пишува General и одберете го објектот Form т.е. самата форма, како што е прикажано на сликата:

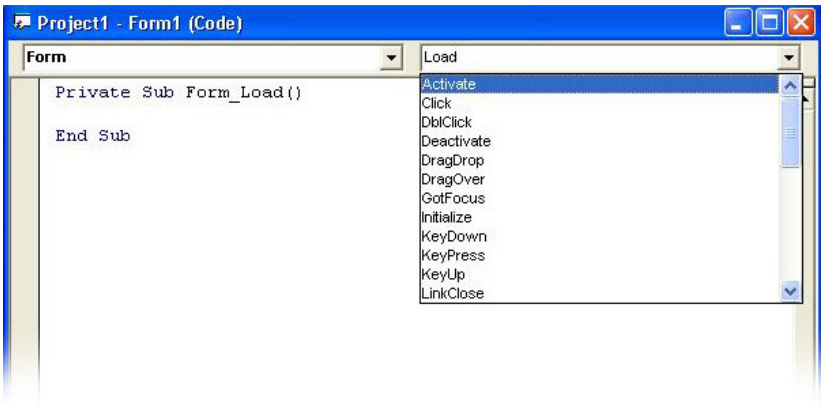


3 чекор. Тогаш, автоматски од списокот со настани (Procedure) се одбира настанот Load и во прозорецот на кодот автоматски се испишува почетокот и крајот на процедурата Form_Load т.е. следните програмски редови

```
Private Sub Form_Load()

End Sub
```

Меѓу почетниот и крајниот програмски ред на една процедура се вметнуваат програмски редови кои го сочинуваат *телото на процедурата*.

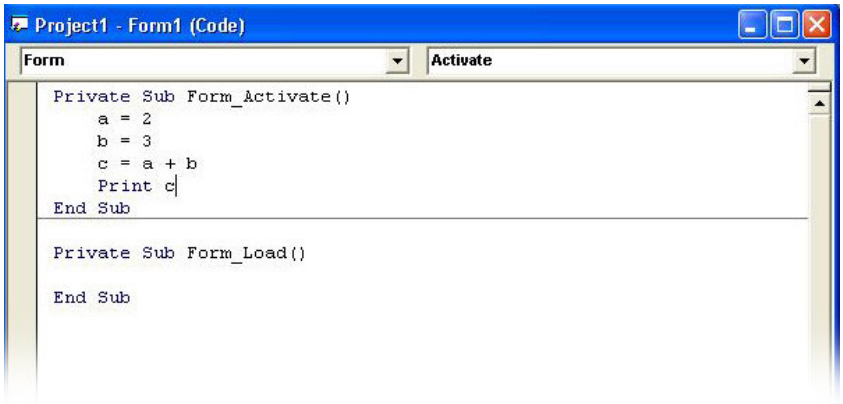
Но, за нашите први ВБ-проекти ќе користиме друг настан на формата. Затоа, треба да кликнете на полето со настани и од списокот со настани да го одберете настанот Activate, како што е прикажано на сликата:



4 чекор. Во самото тело на процедурата Form_Activate која се извршува кога прозорецот на формата ќе стане активен ("во центарот на вниманието"), вметнете ги следните програмски редови:

```
a = 2
b = 3
c = a + b
Print c
```

По овој чекор, прозорецот на кодот треба да изгледа вака:



5 чекор. За да го стартувате ВБ-проектот, кликнете на копчето Start  од лентата со инструменти или од менито Run кликнете на иконата Start или кликнете на копчето F5 од тастатурата.

Тогаш, на екранот ќе се појави формата (Form1) на која ќе биде испишан збирот на броевите 2 и 3, т.е. бројот 5. Како што гледате,

Наредбата Print служи за печатење на екранот. Бојата на наредбата Print, како и на сите други наредби, во прозорецот на кодот е сина.

6 чекор. За да го затворите ВБ-проектот, кликнете во горниот десен агол од формата на копчето Close  или кликнете на копчето End  од лентата со инструменти или од менито Run кликнете на иконата End.

Пробајте да внесете делумни измени во програмските редови, за да го натерате Visual Basic да ви ги прикаже резултатите од следните операции:

$$23-9,\ 3\cdot 7,\ 45:5,\ 2^5,$$

а може да ги испробате вашите програмерски квалитети правејќи програма за решавање на следните изрази:

$$16-(2+5),\ (12-7)\cdot(4+3),\ 12-72:3\cdot 2,\ 4^3\cdot 2-3^4\cdot 5.$$

Мала помош од наша страна е да ви ги кажеме знаците за некои од основните операции кои се користат при пишувањето на програмските кодови во Visual Basic. Тоа се, + за собирање, - за одземање, * за множење, / за делење и ^ за степенување (на пример 2^5, во кодот на Visual Basic ќе се напише 2^5). Нели, станува интересно.

Првиов, ВБ-проект нема да го снимиме сега, затоа што тоа ќе го направиме со следниот. Доколку сакате да го зачувате, прочитајте ја следната точка за тоа како се снима еден ВБ-проект.

Сега, ќе преминеме на правење на вториот ВБ-проект, кој не би можеле така лесно да го менувате, сега за сега. Се работи за програма за пресметување на сумата (збирот)

$$1+2+2^2+2^3+\ldots+2^{19}.$$

7 чекор. Кликнете на File менито и од списокот со икони кликнете на иконата New Project. Ќе бидете запрашани дали сакате да ги снимите измените во датотеките на проектот на кој до сега работевте. Одговорете со кликање на копчето No. Потоа, треба да го одберете типот на ВБ-проектот, слично како на почетокот при стартувањето на програмскиот пакет. Одберете го стандардниот тип на ВБ-проект (Standard EXE) и кликнете на копчето OK.

8 чекор. Применете ги инструкциите од 1, 2 и 3 чекор од оваа точка за да го отворите прозорецот на кодот и на него да се прикаже почетокот и крајот на процедурата Form_Activate т.е. следните програмски редови

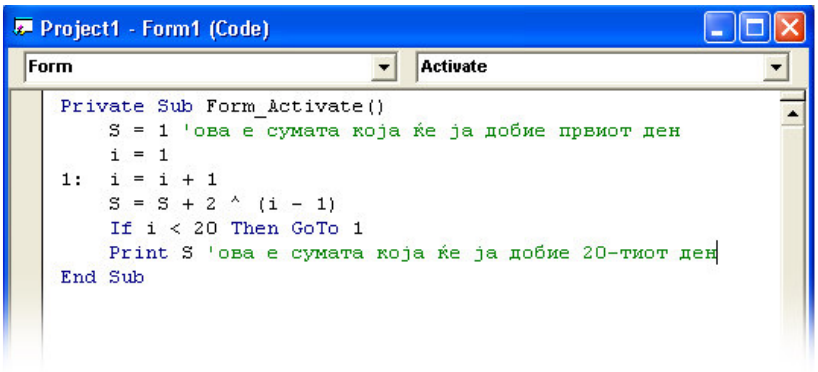
```
Private Sub Form_Activate()

End Sub
```

меѓу кои треба да ги вметнете следните програмски редови

```
    S = 1 'ова е сумата која ќе ја добие првиот ден
    i = 1
1:  i = i + 1
    S = S + 2 ^ (i - 1)
    If i < 20 Then GoTo 1
    Print S 'ова е сумата која ќе ја добие 20-тиот ден
```

На крајот од било кој програмски ред, може да се вметнуваат *коментари*, после апостров. Тие коментари, Visual Basic не зи зема во предвид при извршувањето на програмските редови. Нивната боја во прозорецот на кодот е зелена.

Сега, прозорецот на кодот, треба да изгледа вака:



9 чекор. Стартувајте го ВБ-проектот, според инструкциите во 5 чекор и на формата ќе се појави бројот 1048575.

Погледнете го сега повторно кодот кој го внесовте. Да ја забележиме основната разлика мегу математичкиот јазик и јазикот на програмирањето. Во програмирањето можно е да се изврши програмскиот ред

$$S = S + 2\text{^}(i-1)$$

за секоја вредност на i од 2 до 20, додека во математичкиот јазик ваквиот вид на израз е апсурден. Меѓутоа, во програмирањето се работи за чекори на програмата. Имено,

кога $i = 2$, тогаш $S = 1 + 2\text{^}1 = 3$,
кога $i = 3$, тогаш $S = 3 + 2\text{^}3 = 7$,
...
кога $i = 20$, тогаш $S = 524287 + 2\text{^}19 = 1048575$.

10 чекор. Ајде сега да го снимаме овој ВБ-проект. Бидејќи, овој ВБ-проект го креиравме како нов проект, затоа снимањето ќе биде според вториот начин (види ја следната точка), т.е. кликнете на иконата Save Project As… од File менито и ќе бидете прашани за името на датотеките и местото каде сакате да ги снимите. Препорачуваме, да креирате нова папка со помош на копчето Create New Folder  , во која ќе ги снимите сите датотеки од овој ВБ-проект според редоследот кој ќе го диктира самиот програмски пакет. Со кликање на копчето Save, одобрете го снимањето на датотеките Form1.frm и Project1.vbp во папката Suma што самите ја креиравте.

11 чекор. Време е да излеземе, да го отстраниме активниот ВБ-проект. Тоа се извршува со кликање на иконата Remove Project од File менито.

Така, чекор по чекор, минавме заедно низ дел од основните функции на овој програмски пакет. Се надеваме дека се заинтересиравте. За следните лекции ви припремаме нови програмчиња, како што се наоѓање на сите прости броеви до 1000, барање на најголем заеднички делител на два броја и други.

1.3. Снимање на ВБ-проекти. Отворање на снимени ВБ-проекти. Излегување од Visual Basic

Кога ќе завршите со работата во Visual Basic, снимајте го отворениот ВБ-проект (за почеток препорачуваме да не работите истовремено на повеќе ВБ-проекти, а ако ненамерно сте успеале да отворите истовремено и друг ВБ-проект, тогаш тој се отстранува со кликање на иконата Remove Project од File менито), а потоа затворете го програмскиот пакет.

Снимањето на ВБ-проектите може се извршува на три начина во зависност од претходниот вид на работа со ВБ-проектот:

I начин. Ако ВБ-проектот е веќе снимен и вие сте внесувале само измени во рамките на веќе постоечките датотеки (измени и дополнувања во кодот, својствата на објектите или сте додавале нови објекти на постоечките форми) без да биде додадена нова датотека (.frm, .bas или друга датотека), тогаш снимањето се изведува само со кликање на копчето Save Project  од лентата со инструменти или со кликање на истоимената икона од File менито.

II начин. Ако сте работеле на нов ВБ-проект, тогаш со кликање на иконата Save Project As… од File менито, ќе бидете прашани за името и местото на датотеките каде сакате да го снимите ВБ-проектот. Ви препорачуваме да сите датотеки во рамките на еден ВБ-проект ги снимите во иста папка (фолдер) која ќе биде само за тој ВБ-проект. Во овој случај треба да го забележите редоследот на снимањето на датотеките, одреден од страна на самиот програмски пакет т.е треба да увидите дека последна се снима датотеката на ВБ-проектот (.vbp).

III начин. Ако ВБ-проектот е веќе снимен и вие сте внесувале измени во рамките на веќе постоечките датотеки и сте додавале нови датотеки (.frm, .bas или други датотеки) или едноставно сакате внесените измени само во постоечките датотеки да ги зачувате во нов ВБ-проект, тогаш ќе треба самите да го водите снимањето датотека по датотека, така што датотеката што сакате да ја снимате, ја селектирате преку прозорецот на проектот, потоа со кликање на иконата Save .frm As… или Save .bas As… (во зависност од видот на датотеката) од File менито, ги одбирате новото име и новото место на соодветната датотека. Откако ќе завршите со снимањето на сите подредени датотеки (сите датотеки освен .vbp датотеката), снимањето на .vbp датотеката се извршува со кликање на иконата Save Project As… од File менито, со што повторно ќе бидете прашани за името и местото.

Како повторно да ги отворите сниманиот ВБ-проект, во папката Suma? Во File менито побарајте ја иконата Open Project ... и кликнете на неа или кликнете на копчето Open Project  од лентата со инструменти. Влезете во папката Suma, селектирајте ја датотеката Project1.vbp и кликнете на копчето Open или со дупло кликање на датотеката Project1.vbp ќе го отворите ВБ-проектот. Со помош на копчињата View Code и View Object можете да ги видите кодот или прозорецот на формата. Понатаму продолжете самите да "експериментирате", само внимателно, чекор по чекор за да ја увидите промената која сте ја направиле и да научите и нешто ново самите.

Затворањето на програмскиот пакет Visual Basic се изведува со избирање на командата Exit од File менито.

1.4 Барање на помош

Можеби ова до сега ви изгледаше и малку тешко за разбирање. Но, како за почеток се обидовме да ве снабдиме со основните познавања околу овој програмски пакет, доволни да можете и самите да почете да ги дознавате неговите можности. За таа цел, во меѓувреме, може да ви помогне вграденото упатство со инструкции за користење на овој програмски пакет. Така, одбирајќи го менито Help, со кликање на иконата Microsoft Visual Basic Help Topics, се отвора прозорецот на помошта. И пребарувањето може да започне, било преку Contents или преку Index. На пример, со внесување на зборот Project Explorer во слободното поле во Index одделот, а потоа со

кликање на копчето Display, ќе можете да прочитате нешто повеќе за карактеристиките на прозорецот на проектот.

Ви препорачуваме да одделите извесно време за проучување на помошта што се нуди, пред да преминете на следната лекција. Пријатна работа.

Превод на некои зборови	
Start	почеток, почни
Project	проект
New Project	нов проект
Open	отвори
View Code	види го кодот
View Object	види ги објектите
Close	затвори
End	крај
Create New Folder	креирај нова папка
Save	снимај, зачувај, меморирај
Remove	отстрани
Exit	излез
Help	помош

Лекција 2.

Повторување. Програмирање во Visual Basic. Задача: наоѓање на сите прости броеви до даден природен број

9 чекор. Применете ги инструкциите од 1, 2 и 3 чекор од точката 1.2. за да го отворите прозорецот на кодот и на него да се прикаже почетокот и крајот на процедурата Form_Activate т.е. следните програмски редови

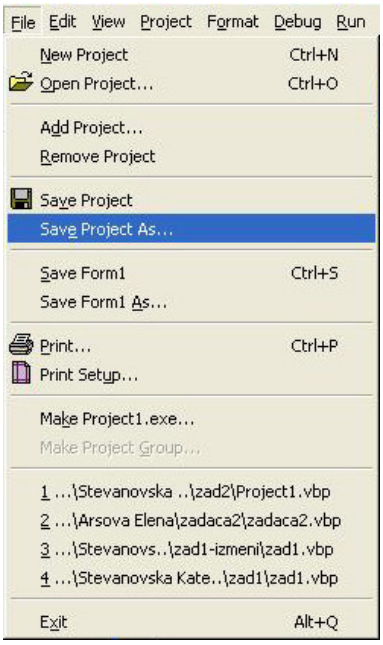
```
Private Sub Form_Activate()


End Sub
```

меѓу кои треба да ги вметнете следните програмски редови

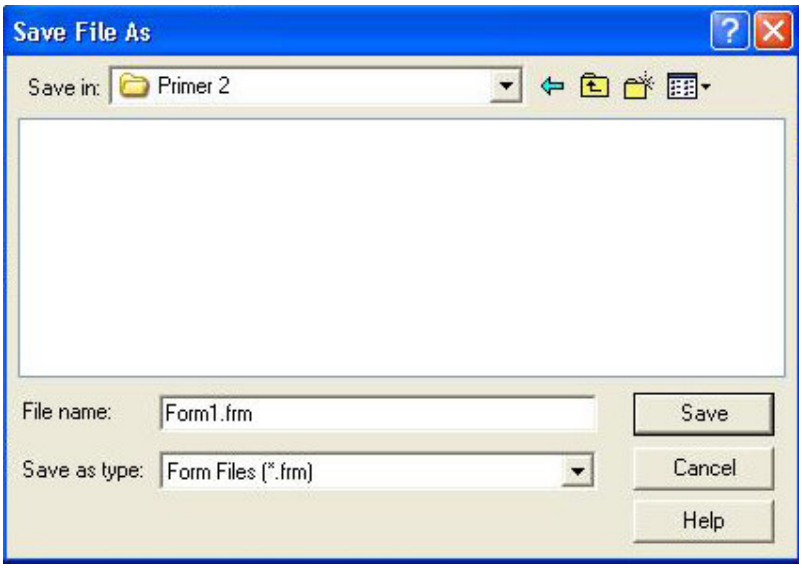
```
n = Val(InputBox("Vnesi priroden broj n, n="))
Print "Site prosti broevi do brojot "; n
For m = 1 To n
        b = 0
        For j = 1 To m
                If m Mod j = 0 Then b = b + 1
        Next j
        If b = 2 Then Print m
Next m
```

10 чекор. Стартувајте го VB-проектот, според инструкциите во 5 чекор и ќе бидете запрашани да го внесете природен број n. Внесете, на пример 50 и кликнете на копчето OK, и ќе видите дека на формата ќе ви се испишат сите простите броеви до 50 т.е. броевите 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47 во колона. Затворете го VB-проектот, како што ви е напишано во 6 чекор од точката 1.2.

11 чекор. Ајде сега да го снимаме овој VB-проект. Бидејќи, овој VB-проект го креиравме како нов проект, затоа снимањето ќе биде според вториот начин, т.е. кликнете на иконата Save Project As… од File менито (како што е прикажано на сликата),



и ќе бидете прашани за името и местото на датотеките каде сакате да го снимите VB-проектот. Препорачуваме, да креирате нова папка, таму каде што сакате да биде, со помош на копчето Create New Folder  , во која ќе ги снимите сите датотеки од овој VB-проект според редоследот кој ќе го диктира самиот програмски пакет. Со кликање на копчето Save, одобрете го снимањето на датотеките Form1.frm и Project1.vbp во папката Primer 2 што самите ја креиравте.

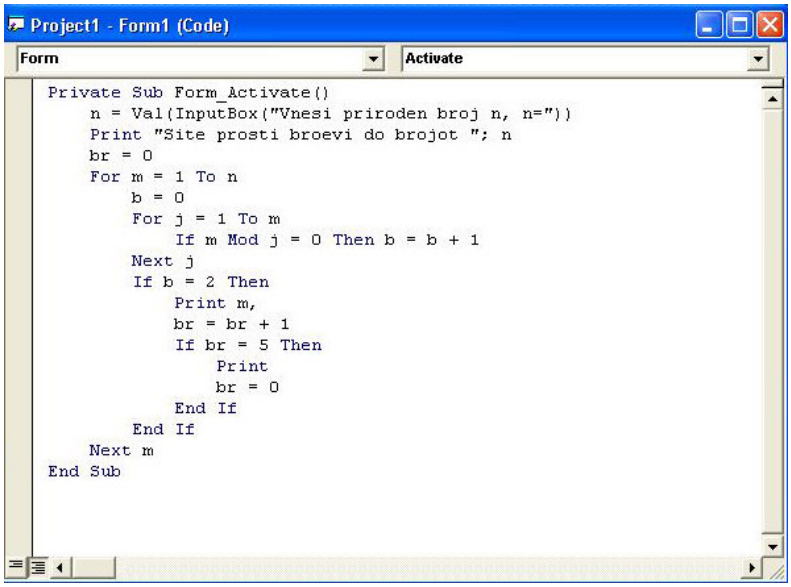


Откако го снимивте VB-проектот, може повторно и повторно да го стартувате, при што секој пат внесувајте нови вредности за природниот број n. Ќе забележите дека ако n е голем тогаш на формата нема да има место да се испишат сите прости броеви до него. Затоа, ќе ги направиме следните измени со цел испишувањето на простите броеви да биде во редици, на пример по 5 броја во ред.

11 чекор. Во телото на процедурата Form_Activate сместете ги следните програмски редови

```
n = Val(InputBox("Vnesi priroden broj n, n="))
Print "Site prosti broevi do brojot "; n
br = 0
For m = 1 To n
        b = 0
        For j = 1 To m
                If m Mod j = 0 Then b = b + 1
        Next j
        If b = 2 Then
                Print m,
                br = br + 1
                If br = 5 Then
                        Print
                        br = 0
                End If
        End If
Next m
```

Внимавајте на запирката на крајот од десетиот програмски ред која овозможува секој наредено прост број да се испише во продолжение на претходниот. Но, ако веќе се испишани пет прости броеви тогш следниот прост број се испишува во нов ред (ова е "кажано" од 12-тиот до 15-тиот програмски ред). Така, сега изгледот на прозорецот на кодот треба да биде следниот



12 чекор. Ајде сега да ги снимиме измените во овој VB-проект, под ново име. Тоа е третиот начин на сминање (види ја следната точка). Прво, одберете ја опцијата Save Form1.frm As… од File менито и ќе бидете запрашани за името и местото каде да ја снимите на датотеката на формата. Креирајте нова папка со име Primer 2_1, влезете во неа и со кликање на копчето Save, снимете ја датотеката Form1.frm во новата папка. Потоа, одберете ја опцијата Save Project As… од File менито и ќе бидете прашани за името и местото каде да ја снимите на датотеката на проектот. Снимајте ја датотеката Project1.vbp во папката Primer 2_1.

13 чекор. Време е да излеземе, да го отстраниме активниот VB-проект. Тоа се извршува со кликање на иконата Remove Project од File менито.

Како повторно да ги отворите снимените VB-проекти, во папките Primer 2, Primer 2_1 и Primer 3? Во File мениото побарајте ја иконата Open Project ... и кликнете на неа или кликнете на копчето Open Project  од лентата со инструменти. Влезете во папката на VB-проектот кој сакате да го отворите, селектирајте ја датотеката Project1.vbp и кликнете на копчето Open или со дупло кликање на датотеката Project1.vbp ќе го отворите саканиот VB-проект. Со помош на копчињата View Code и View Object можете да ги видите кодот или прозорецот на формата. Понатаму продолжете самите да "експериментирате", само внимателно, чекор по чекор за да ја увидите промената која сте ја направиле и да научите и нешто ново самите.

До сега, ги поминавме инструкциите за стартување на пакетот Visual Basic 5.0, објаснувањата за составните елементи на околината за програмирање, снимањето на ВБ-проекти и барањето помош, и **делови од овие инструкции и објаснувања ги повториме и во втората лекција**. Заедно со инструкциите и објаснувањата дадовме решенија на две задачи, во едната се бараше да се пресмета сумата (збирот) $1+2+2^2+2^3+...+2^{19}$, а втората задача бараше да се испишат сите прости броеви до одреден природен број n, при тоа дадовме само делумни објаснувања на програмските редови од овие задачи. Имено, не сакавме уште на самиот почеток да ве оптовариме со многу податоци за овој програмски јазик.

Се надеваме дека се заинтересиравте за овој програмски јазик, иако ви изложивме само две задачи, и дека сигурно веќе се прашувате како би можеле и самите да решите некоја нова конкретна математичка задача. Ќе се обидеме да ви помогнеме во тоа, затоа што **ние сме тука за вас**.

За таа цел, ќе треба да почнеме да ве запознаваме со **основите на програмирањето**.

Лекција 3.
Променливи и операции во Visual Basic

Користењето на променливи и операции при правењето на програма во Visual Basic е скоро неизбежно. Нив ги користевме и при решавањето на претходните задачи. Еве, да се потсетиме уште еднаш на програмските редови на тие задачи:

```
      S = 1
      i = 1
1:    i = i + 1
      S = S + 2 ^ (i - 1)
      If i < 20 Then GoTo 1
      Print S
```

Задача 1.

```
      n = Val(InputBox("Vnesi priroden broj n, n="))
      Print "Site prosti broevi do brojot "; n
      br = 0
      For m = 1 To n
            b = 0
            For j = 1 To m
                  If m Mod j = 0 Then b = b + 1
            Next j
            If b = 2 Then
                  Print m,
                  br = br + 1
                  If br = 5 Then
                        Print
                        br = 0
                  End If
            End If
      Next m
```

Задача 2.

Така, во задачата за пресметување на сумата (првата задача) променливи беа **S** и **i**, а во задачата за прости броеви (втората задача) тоа беа **n**, **br**, **m**, **j** и **b** (за полесно воочување на променливите во претходните две задачи, ви ги прикажуваме со масни, задебелени букви). Имено, тие претставуваа временско место за складирање на одредени информации во одреден момент.

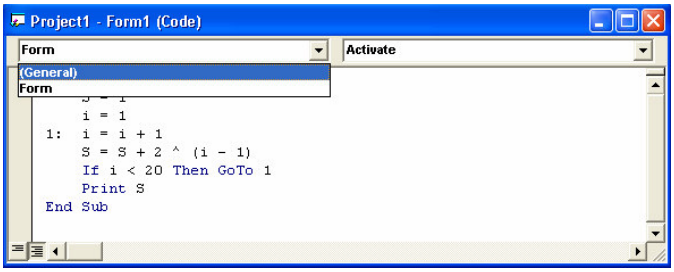
За именување на променливите треба да се користат кратки, интуитивни и лесни за памтење имиња. Така на пример S ја означуваше сумата, n го означуваше природниот број до кој ги баравме простите броеви, m беше природниот број меѓу 1 и n чиј број на делители го баравме, а со b го означувавме бројот на делители на природниот број m. Потоа, **името на секоја променлива треба да започнува со буква** (ова е барање од самиот програмски јазик Visual Basic), **името на променливата треба да содржи најмногу 256 знаци и не смее да содржи празни места** (исто така барања од самиот програмски јазик Visual Basic).

Користењето на променливи во Visual Basic е исто како добивање на маса во некој ексклузивен, модерен ресторан. Ако ја имате масата, ќе може да ја посетувате во секое време, кога ќе посакате, но најдобро (за да биде успешно раководењето, менаџментот на ресторанот) ќе биде ако претходно направите резервација. Да ги погледнеме повторно програмите на првите две задачи. И во првата и во втората задача се користат променливи (читај: "маси во ресторанот") за кои Visual Basic (читај: "раководството, менаџментот на ресторанот") не знае каков тип на информации (читај: "луѓе") ќе складираат (читај: "седнат"), дали ќе бидат тоа нумерички податоци (цели броеви, децимални броеви) или пак ќе бидат текстуални податоци (зборови, реченици, текстови). Затоа, тој автоматски одредува дека променливите ќе складираат информации од типот **Variant**, што значи дека податоците складирани во нив може да се од секаков тип. При тоа, од начинот на работа со променливите, Visual Basic одредува дали ке бидат тоа нумерички или текстуални променливи. Но, во тој случај Visual Basic резервира поголемо место во меморијата за променливата од типот Variant, што не е добро за ефикасноста на програмата (читај: "успешното раководење на ресторанот").

Затоа, **препорачливо е однапред да се каже каков тип на информации ќе складираат променливите**, односно променливите однапред треба да се декларираат, однапред треба да се резервира место за нив во меморијата. Покрај типот Variant, ќе се запознаеме и со типовите **Integer, Long, Single, Double и String**. Во следната табела се дадени кој вид податоци, информации складира променлива од секој од погорните типови поединечно:

Integer	цели броеви од -32768 до 32767
Long	цели броеви од -2147483648 до 2147483647
Single	децимални броеви од $-3{,}402823\cdot10^{38}$ до $3{,}402823\cdot10^{38}$
Double	децимални броеви од $-1.79769313486232\cdot10^{308}$ до $1.79769313486232\cdot10^{308}$
String	текстови од 0 до 65535 знака (било кој знак, дури и празно место)
Variant	сите типови на податоци

Декларирањето на една променлива е со зборчето **Dim** (кратенка од dimension), во делот **(General) (Declarations)**, во кој се пристапува откако од списокот на објекти се одбира, се клика на (General), тогаш автоматски од списокот со настани се одбира, се појавува (Declarations). Да го илустрираме ова на првата задача:

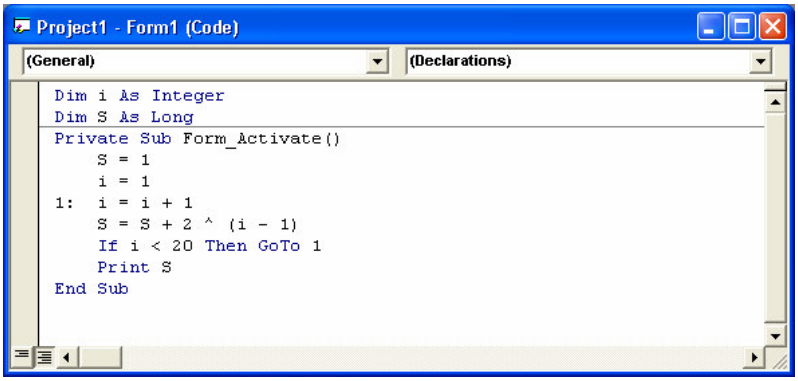


Во оваа задача, променливите S и i примаат целобројни вредности, при тоа i прима вредности од 1 до 20, а најголемата вредност која ја прима S е ктајниот резултат, односно бројот 1048575. Па, според горната табела на вредности, најоптимално е променливата i да ја декларираме како Integer променлива, а S како Long променлива. За таа цел, во делот (General) (Declarations) ги запишуваме следните програмски редови:

Dim i As Integer

Dim S As Long

и тогаш, крајниот облик на првата задача е следниот:



Задача за вежби:
Обидете се самите да ги декларирате променливите во втората задача.
Упатство: сите променливи примаат вредности цели броеви.

Време е да преминеме на решавање на нова конкретна математичка задача, при што ќе ги декларираме променливите кои ќе ги користиме и ќе ги увидиме **правилата за користење на операциите во Visual Basic**.

Задача 3. *Пресметај ја вредноста на изразот*

$$a^2 + b^2 + \frac{a+b}{a-b} - \frac{b+a}{a \cdot b},$$

за a *и* b *цели броеви.*

Бидејќи е напоменато дека a и b се цели броеви, затоа истоимените променливи во програмскиот код можеме истовремено да ги декларираме како:

Dim a, b As Integer

и при тоа, за практична примена на оваа задача, доволно е да обезбедиме да тие примаат вредности од помалиот обсег на цели броеви т.е. Integer, а не Long.

Потоа, ќе и го доделиме името izraz на променливата која ќе ја содржи вредноста на изразот кој треба да го пресметаме (кратко, јасно и лесно за паметење име на променлива). Јасно е дека izraz може да прима и децимални вредности, па затоа таа променлива ќе ја декларираме како:

Dim izraz As Double

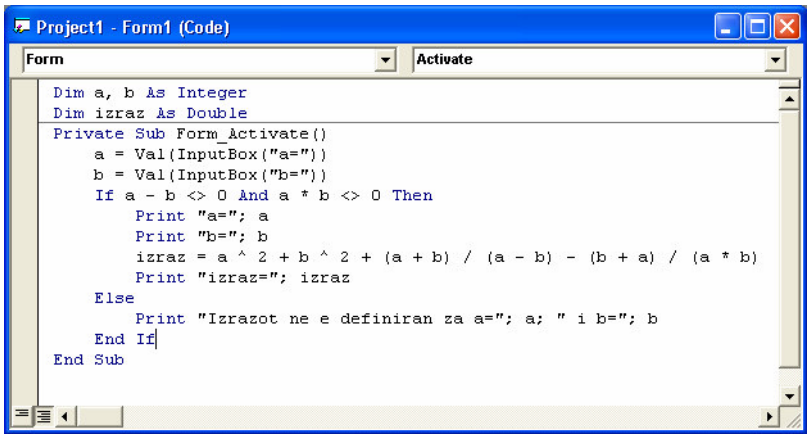
и избираме да доколку izraz прими децимална вредност, таа да биде пресметана и прикажана со поголема прецизност, на повеќе децимали, и затоа ја декларираме да биде од типот Double, а не од типот Single.

Да се потсетиме дека програмите ги пишуваме во процедурата Form_Activate, па за оваа задача, телото на таа процедура е следното:

```
a = Val(InputBox("a="))
b = Val(InputBox("b="))
If a - b <> 0 And a * b <> 0 Then
   Print "a="; a
   Print "b="; b
   izraz = a ^ 2 + b ^ 2 + (a + b) / (a - b) - (b + a) / (a * b)
   Print "izraz="; izraz
Else
   Print "Izrazot ne e definiran za a="; a; " i b="; b
End If
```

Задача 3.

Тогаш, декларирањето заедно со телото на процедурата Form_Activate, за оваа задача изгледа вака:



Овој пат ќе се задржме на тоа како е пресметана вредноста на изразот, односно кои операции и како се користени.

Иако во првата лекција ви укажавме на некои од операциите кои се користат во Visual Basic, тука ќе ги повториме истите и ќе научите и нешто повеќе за нив и за останатите операции. Значи,

Основни операции	
+	собирање
-	одземање
*	множење
/	делење

Посложени операции	
^	степенување
\	целобројно делење
Mod	остаток при делење
&	слепување

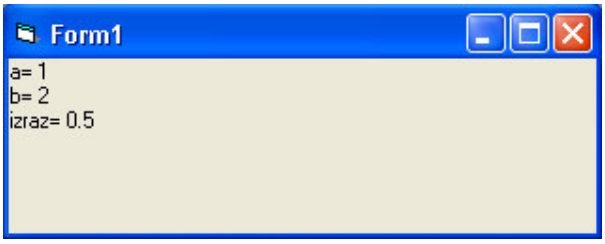
И имајќи го во предвид **приоритетот, првринството на извршување на операциите**, дадено со следната табела:

Приоритет на извршување на операциите	
( )	вредноста во загради секогаш прва се предметува
^	дигањето на степен е второ по приоритет
-	создавањето на негативен број е трето
* /	множењето и делењето се четврти
\	целобројното делење е петто
Mod	остатокот при делење е шести
+ -	собирањето и одземањето се последни

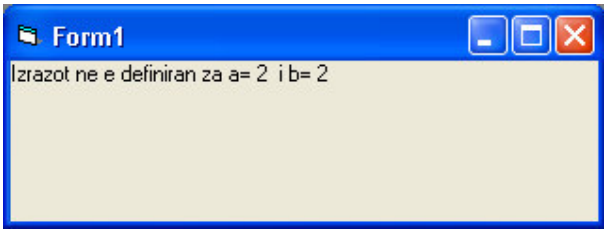
јасно е дека изразот $a^2+b^2+\frac{a+b}{a-b}-\frac{b+a}{a\cdot b}$ мора во кодот на Visual Basic да се запише како a ^ 2 + b ^ 2 + (a + b) / (a - b) - (b + a) / (a * b), што значи дека, на пример, за a = 1 и b = 2, операциите при пресметување на вредноста на променливата izraz, Visual Basic ќе ги извршува во следните етапи:

izraz = 1 ^ 2 + 2 ^ 2 + (1 + 2) / (1 - 2) - (2 + 1) / (1 * 2)
izraz = 1 ^ 2 + 2 ^ 2 + 3 / (-1) - 3 / 2
izraz = 1 + 4 + 3 / (-1) - 3 / 2
izraz = 1 + 4 + (-3) - 1.5
izraz = 0.5

а тоа што ќе се прикаже на формата откако сте внеле вредности 1 за a и 2 за b е следното:



Доколку внесете вредности за a и b такви да a - b = 0 или a * b = 0, тогаш пресметувањето на изразот ќе нема смисла, па на формата, на пример ако внесете 2 за a и 2 за b ќе се прикаже следното:



Ќе ве запознаеме со уште две (многу важни) правила при користењето на операциите во Visual Basic.

Правило 1. Извршувањето на операциите * и / (бидејќи се со ист прироритет), доколку нема загради, е последователно. Така на пример,

2 * 6 / 4 * 2 = 12 / 4 * 2 = 3 * 2 = 6,

додека

(2 * 6) / (4 * 2) = 12 / 8 = 1.5.

Истото важи и при извршувањето на операциите + и - (бидејќи и тие се со ист приоритет). Па, во тој случај имаме,

2 + 6 - 4 + 2 = 8 - 4 + 2 = 4 + 2 = 6

и

(2 + 6) - (4 + 2) = 8 - 6 = 2.

Правило 2. (важи само за пакетот Visual Basic 5.0, инаку овој недостаток е отклонет во посовршените верзии на овој програмски јазик, на пример Visual Basic 6.0) Ако во првите два програмски реда го отстраниме користењето на функцијата **Val**, тогаш тие ќе изгледаат на следниот начин

```
a = InputBox("a=")
b = InputBox("b=")
```

и ако сега ја стартуваме програмата, и при тоа повторо внесеме 1 за a и 2 за b, може да се случи да го добиеме резултатот izraz = -17.5. Што се случило?

Имено, иако ги имаме дефинирано a и b како променливи кои ќе складираат нумерички вредности, поточно цели броеви, сепак кога ги задаваме нивните вредности со помош на функцијата InputBox, при извршување на операцијата собирање (+), настанува забуна. Тогаш, Visual Basic работи со нив како да се текстови, па операцијата собирање (+) делува исто како **операцијата слепување (&)**, ги слепува вредностите на тие две променливи. Додека, останатите операции ги извршува исто како претходно. Па еве како Visual Basic, во тој случај доаѓа до резултатот -17.5,

```
izraz = 1 ^ 2 + 2 ^ 2 + (1 + 2) / (1 - 2) - (2 + 1) / (1 * 2)
izraz = 1 ^ 2 + 2 ^ 2 + 12 / (-1) - 21 / 2
izraz = 1 + 4 + 12 / (-1) - 21 / 2
izraz = 1 + 4 + (-12) - 10.5
izraz = -17.5
```

што воопшто **не е математички исправно!!**

Затоа, **не заборавајте кога доделувате вредности на променливите со помош на InputBox, да ја користите функцијата Val**.

Задача за вежби:

За a, b и с произволни броеви, пресметај ја вредноста на изразот

$$\frac{ab(c^2+1)+c(a^2+b^2)}{ab(c^2-1)+c(a^2-b^2)}.$$

Остана уште да објасниме малку повеќе за **операциите целобројно делење (\) и остаток при делење (Mod)**. Последнава операција, веќе ја употребивме во втората задача, кока во еден од програмските редови испитувавме дали j е делител на m, односно дали остатокот при делење на m со j е нула (If m Mod j = 0 …).

Овие две операции се некако меѓусебно поврзани. Имено, едната ја дава вредноста на целобројниот количник при делење на два цели броја, а втората остатокот при то делење. Така на пример,

```
17 \ 3 = 5,
17 Mod 3 = 2,
```

затоа што 17 : 3 = 5 (остаток 2). (**Внимавајте!** Лесно може да дојде до забуна при користење на знакот за делење (/) и знакот за целобројно делење (\), затоа што и двете се коси црти само со обратни правци)

Практичната употреба на овие операции, ќе ја согледате при решавањето на следната задача.

Задача 4. *Најди ги сите трицифрени броеви кои се 7 пати поголеми од збирот на квадратите на своите цифри.*

Како ли сега да ја направиме програмата за оваа задача? Ајде да почнеме од крајот!

На пример, еден од тие броеви е бројот 917, затоа што

7 * (9 ^ 2 + 1 ^ 2 + 7 ^ 2) = 7 * (81 + 1 + 49) = 7 * 131 = 917.

Па, благодарение на брзината на денешните сметачи, можеме многу брзо да ги испитаме сите трицифрени броеви (од 100 до 999), за секој поединечно да провериме дали го задоволува бараниот услов (да е 7 пати поголем од збирот на квадратите на своите цифри) и доколку најдеме таков број да го прикажеме на формата. Но, најголемиот проблем е како знаејќи ја вредноста на еден трицифрен број да ги добиеме вредностите на неговите цифри?

Знаејќи како функционираат операциите целобројно делење (\) и остаток при делење (Mod), не е тешко да заклучиме дека цифрите 9, 1, 7 на бројот 917, може да бидат издвоени на следниот начин,

9 = 917 \ 100,

1 = (917 Mod 100) \ 10 или 1 = (917 \ 10) Mod 10,

7 = (917 Mod 100) Mod 10 или 7 = 917 Mod 10,

Навистина, од 917 : 100 = 9 (остаток 17), заклучуваме дека 917 \ 100 = 9 и 917 Mod 100 = 17, па издвојувањето на цифрата на стотките 9 е во ред. Понатаму, од 17 : 10 = 1 (остаток 7), заклучуваме дека 17 \ 10 = 1 и 17 Mod 10 = 7, па добивањето на цифрата на десетките 1 и цифрата на единиците 7 е исто така во ред.

Сега веќе е полесно. За променливата која времено ќе ја складира вредноста на некој трицифрен број, ќе и го доделиме името n, со a ќе ја означиме цифрата на стотки, со b цифрата на десетки и со c цифрата на единици на трицифрениот број n, а со zbir го означуваме името на променливата која ќе ја складира вредноста на збирот на квадратите на цифрите a, b и c. И како сите овие променливи ќе примаат целобројни вредности, нив можеме истовремено да ги декларираме во делот (General) (Declarations) како:

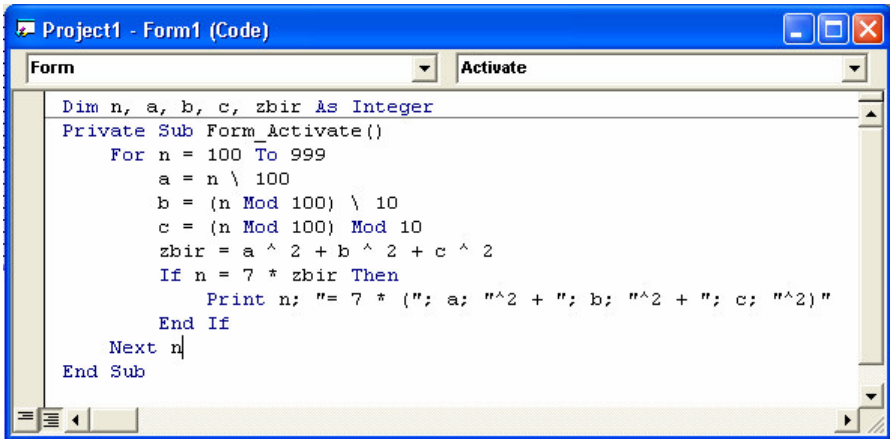
Dim n, a, b, c, zbir As Integer

а во телото на процедурата Form_Activate ќе ги вметнеме следните програмски редови:

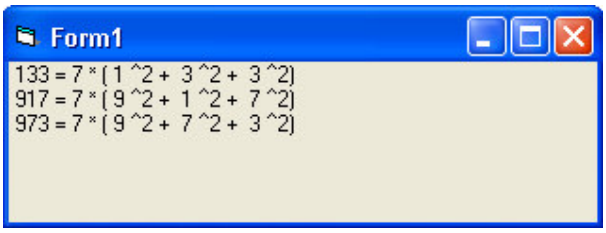
```
For n = 100 To 999
   a = n \ 100
   b = (n Mod 100) \ 10
   c = (n Mod 100) Mod 10
   zbir = a ^ 2 + b ^ 2 + c ^ 2
   If n = 7 * zbir Then
      Print n; "= 7 * ("; a; "^2 + "; b; "^2 + "; c; "^2)"
   End If
Next n
```

Залача 4.

Тогаш, декларирањето заедно со телото на процедурата Form_Activate, за оваа задача изгледа вака:



Стартувајте ја програмата (да се потсетиме, на пример, со копчето F5 од тастатурата), и на формата ќе ви се прикаже следното:



односно, одговорот на задачата е дека броевите 133, 917 и 973 се единствените трицифрени броеви кои се 7 пати поголеми од збирот на квадратите на своите цифри.

Пробајте сега, оваа задача да ја решите без помош на Visual Basic. ☺

И за крај, доколку при решавањето на математичките задачи сте дошле до **некоја задача која ви предизвикала тешкотии** (на такви задачи секој "налетува", инаку математиката нема да биде интересна без тешкотии), јавете ни се, пишете ни (numerus@mt.net.mk), а ние ќе видиме **како Visual Basic може да помогне**.

Продолжуваме да ве запознаваме со **основите на програмирањето**. Следното наше излагање ќе биде за наредбите For…Next, If…Then и SelectCase, и секако поткрепено со решавање на нови конкретни математички задачи.

Лекција 4.
Циклусот For…Next и условните структури If…Then и SelectCase

Првите две наредби For…Next и If…Then скоро и да се неизбежни при програмирањето. Така, ние до сега ги употребивме скоро во секоја од програмите што ви ги изложивме. За да можете и самите да ги користите во креирањето на нови програми, ќе се обидеме да ви го доближиме нивното значење објаснувајќи за нивната синтакса (описот на наредбата) и употребата.

Синтаксата на циклусот **For…Next** е следната:

For *бројач=почеток* **To** *крај* [**Step** *чекор*]
[*наредби*]
[**Exit For**]
[*наредби*]
Next [*бројач*]

Објаснување на деловите од циклусот For…Next:

1) *бројач* е нумеричка променлива која се користи како бројач на циклусите, односно колку пати ќе се повторат наредбите меѓу For и Next
2) *почеток* е почетната вредност на бројачот
3) *крај* е крајната вредност на бројачот
4) *чекор* покажува за колку во секој циклус се менува вредноста на бројачот, ако недостасува чекорот се подразбира дека е 1
5) наредбата Exit For може да се постави било каде меѓу For и Next и означива алтернативен начин за излез од циклусот For…Next, односно продолжување на извршување на наедбите после Next

Да забележиме дека, деловите кои се во средни загради не се задолжителни, односно и без нив ќе биде фуционален циклусот For…Next. Но, сепак се препорачува веднаш до Next да биде напишана променливата која се користи како бројач во тој циклус, за да не настане забуна при користење на повеќе циклуси кои имаат различни бројачи и се сместени еден во друг.

Условната структура **If…Then** се среќава во проста и сложена форма. Синтаксата на простата условна структура If…Then е:

If *услов* **Then** *наредба1* [**Else** *наредба2*]

а нејзиното и значењето на нејзините делови е во следното: *услов* е нумерички или текстуален израз кој прима вредности True (точно) или False (неточно) кога условот *услов* е исполнет или не, и доколку тој е исполнет се извршува *наредба1*, а ако не е исполнет тогаш се извршува *наредба2*.

Синтаксата пак на сложената условна структура If…Then е:

If *услов* **Then**
[*наредби*]
[**ElseIf** *услов-1* **Then**
[*наредби-1*]]
[**ElseIf** *услов-2* **Then**
[*наредби-2*]]
...
[**ElseIf** *услов-n* **Then**
[*наредби-n*]]
[**Else**
[*останатинаредби*]]
End If

Нејзиното значење се објаснува на следниот начин: доколку е исполнет условот *услов*, се преминува на извршување на наредбите *наредби*, а потоа на наредбите после End If. Ако не е исполнет условот *услов*, последователно се проверува дали е исполнет некој од условите *услов-1*, *услов-2*, се до *услов-n*, и ако се добие да некој од тие услови е исполнет (но само по тој редослед се врши проверката), тогаш се извршуваат наредбите после соодветното Then, па се преминува на наредбите после End If. И ако ниеден од условите *услов, услов-1*, *услов-2*, се до *услов-n*, не е исполнет, тогаш се преминува на извршување на наредбите после Else.

За формирање на изразите кои стојат на местото од *услов, услов-1*, *услов-2*, се до *услов-n*, се користат **логичките релации и операции**. Тоа се следните:

Логички релации	
=	еднакво на
<>	различно од
<	помало од
>	поголемо од
<=	помало или еднакво на
>=	поголемо или еднакво на

Логички операции	
Not	не
And	и
Or	или
Xor	или...или

Да се обидеме повторно да ги примениме овие две наредби во решавањето на следните задачи:

Задача 5. *Одреди ги сите можни вредности на цифрите* a *и* b*, за кои производот на броевите* $\overline{54a}$ *и* $\overline{63b1}$ *е делив со* 12.

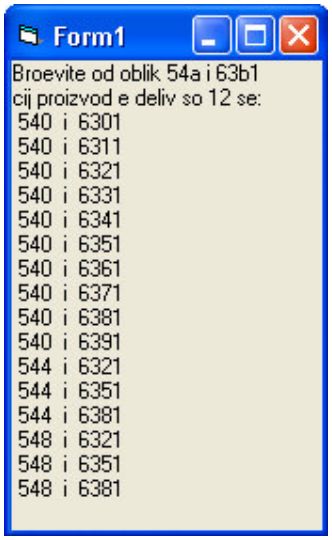
Навидум не така едноставна задача (оваа задача се падна на овогодишниот Регионален натпревар по математика за учениците од основното образование), со помошта од Visual Basic ќе ја претвориме во најлесната задача што сте ја сретнале воопшто. Но, најнапред да ја искоментираме задачата и да предложиме начин за нејзино решавање со Visual Basic, а потоа ќе ви ја презентираме и програмата.

Бидејќи a и b се цифри, значи дека $a,b\in\{0,1,2,3,4,5,6,7,8,9\}$. Со Visual Basic е едноставно да се испробаат сите можни начини за формирање на броевите $\overline{54a}$ и $\overline{63b1}$ (а нив ги има $2^{10}=1024$) и при секој од нив да се провери дали нивниот производ е делив со 12. Со помош на два циклуса For…Next еден за a, а вториот за b, кои ќе бидат еден во друг, ќе ги испитаме сите можни случаи, и при секој од тие случаи со If…Then ќе правиме проверка на условот за деливост на производот со 12. Така, ја добиваме следната програма:

```
Dim a, b, prvbroj, vtorbroj As Integer
Dim proizvod As Long
Private Sub Form_Activate()
   Print "Broevite od oblik 54a i 63b1"
   Print "cij proizvod e deliv so 12 se:"
   For a = 0 To 9
      For b = 0 To 9
         prvbroj = 540 + a
         vtorbroj = 6301 + 10 * b
         proizvod = prvbroj * vtorbroj
         If proizvod Mod 12 = 0 Then
            Print prvbroj; " i "; vtorbroj
         End If
      Next b
   Next a
End Sub
```

Задача 5.

Така, по стрaртувањето на програмата на формата ќе се прикаже следниот резултат:



Задача 6. *Пресметај ја вредноста на* y, *ако*

$$y = \begin{cases} x_1 + x_2, & ако\ x_1 > x_2 \\ x_1 \cdot x_2, & ако\ x_1 = x_2 \\ x_1 / x_2, & ако\ x_1 < x_2\ и\ x_2 \neq 0 \end{cases}$$

каде x_1 *и* x_2 *се однапред познати броеви.*

Оваа задача е добра за илустрирање на сложената условна структура If…Then. Имено, имаме дека во зависност од релацијата меѓу x_1 и x_2, вредноста на y се пресметува различно. Да забележиме дека во случајот кога $x_1 < x_2$ и $x_2 = 0$, вредноста на y не е дефинирана. Така, ја формираме следната програма:

```
Dim x1, x2, y As Double
Private Sub Form_Activate()
   x1 = Val(InputBox("x1="))
   x2 = Val(InputBox("x2="))
   Print "Za x1="; x1; " i x2="; x2; ", ";
   If x1 > x2 Then
      y = x1 + x2
      Print "y=x1+x2="; y
   ElseIf x1 = x2 Then
      y = x1 * x2
      Print "y=x1*x2="; y
   ElseIf x1 < x2 And x2 <> 0 Then
      y = x1 / x2
      Print "y=x1/x2="; y
   Else
      Print "y ne e definiran."
   End If
End Sub
```

Задача 6.

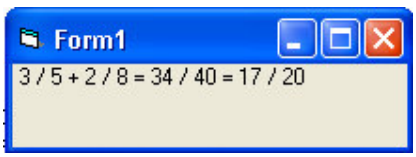
Задача 7. *Собери ги дропките* $\frac{a}{b}$ *и* $\frac{c}{d}$, *каде* a,b,c,d *се цели броеви при што* $b,d \neq 0$. *Резултатот прикажи го во облик на нескратлива дропка.*

Најнапред проверуваме дали при внесувањето на броевите a,b,c,d, внесено е за b или за d нула. Ако е така, да се појави порака (со функцијата MsgBox) и да сопре програмата (со наредбата End). Понатаму, користејќи го равенството $\frac{a}{b}+\frac{c}{d}=\frac{ad+bc}{bd}$, би ги собрале првично дадените дропки, резултатот би бил дропка, но ништо не ни гарантира дека ќе биде нескратлива. Затоа, ќе треба потоа броителот $br=ad+bc$ и именителот $im=bd$ да ги поделиме со нивниот најголем заеднички делител за да резултатот биде нескратлива дропка. Начинот на кој ќе бараме НЗД на два броја е тој да почнувајќи од помалиот од нив, и движејќи се на назад кон 1, би проверувале дали имаме делител и на двата броја, односно заеднички делител (zd), ако да, излегуваме од циклусот, а ако не тогаш го намалуваме чекорот за 1. Така, ја добиваме следната програма:

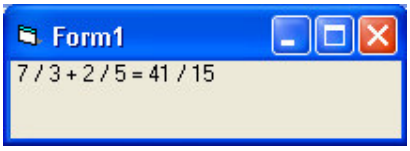
```
Dim a, b, c, d, br, im, nzd, zd, br1, im1 As Integer
Private Sub Form_Activate()
   a = Val(InputBox("a="))
   b = Val(InputBox("b<>0, b="))
   c = Val(InputBox("c="))
   d = Val(InputBox("d<>0, d="))
   If b = 0 Or d = 0 Then
      MsgBox ("Ne moze imenitelot na dropka da e nula")
      End
   End If
   Print a; "/"; b; "+"; c; "/"; d; "=";
   br = a * d + b * c
   im = b * d
   Print br; "/"; im;
   If br < im Then nzd = br Else nzd = im
   For zd = nzd To 1 Step -1
      If br Mod zd = 0 And im Mod zd = 0 Then Exit For
   Next zd
   nzd = zd
   If zd <> 1 Then
      br1 = br / nzd
      im1 = im / nzd
      Print "="; br1; "/"; im1
   End If
End Sub
```

Задача 7.

Ако по стартувањето на програмата внесеме a=3, b=5, c=2, d=8, го добиваме следниот излез:



доколку внесеме a=7, b=3, c=2, d=5, тогаш по собирањето на дропките според горното равенство се добива нескратлива дропка, односно го имаме следниот излез:



А сега, тестирај се кој побрзо ќе собере две дропки, ТИ или VISUAL BASIC.

Условната структура **SelectCase** се користи кога во зависност од вредноста на некој израз би се извршувале различни наредби. Нејзината синтакса е:

Select Case *тестизраз*
  [**Case** *вредност(и) на изразот-1*
    [*наредби-1*]]
  ...
  [**Case** *вредност(и) на изразот-n*
    [*наредби-n*]]
  [**Case Else**
    [*наредби*]]
End Select

Објаснувањето на нејзите делови:

1) *тестизраз*-от може да биде нумерички или текстуален

2) *вредност(и) на изразот* може да биде само една конкретна вредност (на пример, Case 5) или повеќе вредности (на пример, Case 2, 6, 8 или Case Is<5 или Case 3 To 7)

Работата на оваа условна структура се состои во тоа што се проверува најнапред дали *тестизраз*-от прима вредност еднаква на *вредност на изразот-1* или неговата вредност е некоја од *вредности на изразот-1* и ако е така се извршуваат *наредби-1* и потоа продолжува извршувањето на наредбите после End Select. Ако не, настапува проверка дали *тестизраз*-от прима вредност еднаква на *вредност на изразот-2* или неговата вредност е некоја од *вредности на изразот-2*, и така натаму. Ако *тестизраз*-от нема вредност еднаква на некоја од вредностите наведени по секое Case, тогаш се извршуваат наредбите после Case Else.

Ќе ја илуструраме оваа наредба на едноставен пример. Сакаме да го дознаеме остатокот при делење на внесен природен број *n* со 3. За таа цел ја пишуваме следната програма:

```
Dim n, ostatok As Integer
Private Sub Form_Activate()
   n = Val(InputBox("n="))
   ostatok = n Mod 3
   Select Case ostatok
      Case 1
         Print n; " dava ostatok 1 pri delenje so 3"
      Case 2
         Print n; " dava ostatok 2 pri delenje so 3"
      Case Else
         Print n; " e deliv so 3"
   End Select
End Sub
```

Остаток при делење со 3

Се надеваме дека и овој пат успеавме да го задржиме вашето внимание, решавајќи некои математички задачи. После ова, би можеле и вие самите да се обидете во некој математички проблем. СРЕЌНО!

Продолжуваме и оваа учебна година да се дружиме со програмскиот јазик Visual Basic и со негова помош да решаваме некои математички задачи. Покрај воведните лекции за стартувањето на пакетот и запознавањето со околината на програмирање, ве запознавме со променливите и операциите во Visual Basic, циклусот For…Next и условните структури If…Then и Select Case. Во овој број ќе ве запознаеме со циклусот Do…Loop.

Лекција 5.
Циклусот Do…Loop

За разлика од претходно објаснетиот циклус For..Next каде блокот наредби се извршува конечен број пати, циклусот **Do…Loop** може да го извршува блокот наредби како конечно така и бесконечно многу пати. Постојат неколку облици на циклусот Do…Loop, но секој од нив функционира така што ја проверува точноста на некој услов и во зависност од тоа продолжува или не да го извршува блокот наредби.

Синтаксата на најчесто користениот облик на циклусот Do…Loop е:

Do While *услов*
[*наредби*]
Loop

При извршувањето на овој облик на циклусот Do…Loop, најнапред се проверува дали е исполнет условот *услов* и ако тој **не е исполнет** тогаш се прескокнува блокот *наредби* и се продолжува со извршување на програмата после програмскиот ред Loop, доколку пак условот *услов* **е исполнет** тогаш се извршува блокот *наредби* и програмата повторно се враќа на програмскиот ред Do While и повторно ја проверува точноста на условот *услов*.

Токму заради последно кажаното можно е блокот *наредби* да се извршува бесконечно многу пати и таквиот циклус се нарекува **бескраен циклус**. Но, **бескрајните цилуси треба да се избегнат**. Тривијален пример за бескраен циклус е испишувањето на "сите" природни броеви, што "може" да се направи со следните програмски редови:

```
Dim n As Long
Private Sub Form_Activate()
   n = 1
   Do While n >= 1
      Print n
      n = n + 1
   Loop
End Sub
```

Сите природни броеви

Доколку се обидете да ја стартувате оваа програма (со копчето Start од лентата за инструменти или со претискање на F5 од тастатурата) ќе забележите дека на формата Form1 многу брзо во колона се испишале неколку први природни броеви (затоа што толку имало место на формата), а програмата никако да заврши со работата. Тоа е затоа што сте пуштиле да се изврши бескраен цилкус. И како понатаму? Изгледа дека Visual Basic се закочил, дека нема излез од новонастанатата ситуација, и дека можеби ќе треба да го рестартирате сметачот? Не, вие едноставно треба да го прекинете извршувањето на програмата. **Извршувањето на програмата се прекинува со претискање на Ctrl+Break од тастатурата, а потоа со притискање на копчето End од лентата со инструменти.** Сега треба да се внесат измени во кодот за да се отстрани бескрајниот циклус. Знаејќи дека постојат бесконечно многу природни броеви, потребно е барањето на задачата да го преформулираме, односно да побарме да се испишат сите природни броеви до бројот k. Односно,

```
Dim k, n As Long
Private Sub Form_Activate()
   k = Val(InputBox("Vnesi priroden broj, k="))
   n = 1
   Do While n <= k
      Print n
      n = n + 1
   Loop
End Sub
```

Сите природни броеви до k (прв облик)

На овој начин, во зависност од внесената вредност за бројот k и големината на формата Form1, би можело или не да се видат испишани сите природни броеви до бројот k, односно броевите 1, 2, 3, …, k, но најважно е тоа што сега циклусот е конечен, па програмата нема да се закочи (освен ако не внесете за k број надвор од обсегот на типот Long).

Овој облик на циклусот Do…Loop овозможува блокот *наредби* воопшто да не се изврши или да се извршува повеќе пати. Иста функција има и следниот облик на циклусот Do…Loop чија синтакса е:

Do Until *услов*
[*наредби*]
Loop

За разлика од претходниот облик, овде доколку **е исполнет** условот *услов* се прескокнува блокот *наредби* и се продолжува со извршување на програмата после програмскиот ред Loop, доколку пак условот *услов* **не е исполнет** тогаш се извршува блокот *наредби* и програмата повторно се враќа на програмскиот ред Do Until за да ја провери точноста на условот *услов*.

Така последната наша задача запишана со помош на вториов облик на циклусот Do…Loop изгледа вака:

```
Dim k, n As Long
Private Sub Form_Activate()
   k = Val(InputBox("Vnesi priroden broj, k="))
   n = 1
   Do Until n > k
      Print n
      n = n + 1
   Loop
End Sub
```

Сите природни броеви до k (втор облик)

Секако дека има разлика. **Забележете ја промената во условот *услов*!** Имено, **новиот услов е негација на претходниот**.

Следните два облика на циклусот Do…Loop овозможуваат блокот *наредби* да се изврши најмалку еднаш. Синтаксата на третиот облик е:

Do
[*наредби*]
Loop While *услов*

Во овој случај прво се извршува блокот *наредби*, а потоа се проверува дали е исполнет условот *услов*, ако тој **не е исполнет** тогаш се продолжува со извршување на програмата после програмскиот ред Loop While, доколку пак условот *услов* **е исполнет** тогаш се извршува блокот *наредби* и програмата повторно се враќа на програмскиот ред Loop While и повторно ја проверува точноста на условот *услов*.

Нашата задача запишана со помош на третиот облик на циклусот Do…Loop е следната:

```
Dim k, n As Long
Private Sub Form_Activate()
   k = Val(InputBox("Vnesi priroden broj, k="))
   n = 1
   Do
      Print n
      n = n + 1
   Loop While n<=k
End Sub
```

Сите природни броеви до k (трет облик)

Доколку ги споредиме програмите запишани со помош на првиот и третиот облик на циклусот Do…Loop може да се дојде до грешен заклучок дека нема разлика во употребата на првиот и третиот облик, односно дека не е важно каде ќе се стави While *услов*, дали после Do или после Loop. Грешен заклучок! Имено доколку по грешка за k внесете број кој не е природен, на пример 0 или некој негативен број, во секој од тие случаи на формата ќе се прикаже бројот 1, што не е точно решение на задачата. Затоа, овој облик не е погоден за решавање на оваа задача или ако се користи ќе мора да се дополни на следниот начин:

```
Dim k, n As Long
Private Sub Form_Activate()
   k = Val(InputBox("Vnesi priroden broj, k="))
   n = 1
   If k >= 1 Then
      Do
         Print n
         n = n + 1
      Loop While n <= k
   End If
End Sub
```

Сите природни броеви до k
(трет облик - дополнет)

И синтаксата на четвртиот облик е:

Do
[*наредби*]
Loop Until *услов*

И при овој облик прво се извршува блокот *наредби*, а потоа се проверува дали е исполнет условот *услов*, ако тој **е исполнет** тогаш се продолжува со извршување на програмата после програмскиот ред Loop Until, доколку пак условот *услов* **не е исполнет** тогаш се извршува блокот *наредби* и програмата повторно се враќа на програмскиот ред Loop Until за да ја провери точноста на условот *услов.*

Направете ја сами програмата за задачата преку четвртиот облик на циклусот Do..Loop имајќи ги во предвид сите претходни дискусии. Како заклучок на досегашното излагање е дека **обликот на циклусот Do…Loop кој ќе се користи во програмата зависи од природата на задачата која сакаме да ја решиме**.

Откако ви ги објаснивме синтаксата и употребата на циклусот Do…Loop време е да ја согледаме **неговата примена во решавањето на некои математички задачи**. Најчестата примена на овој циклус е при пресметувањето на бројот на цифри на некој број и наоѓање на најголем заеднички делител на два броја. Проследете ги следните задачи.

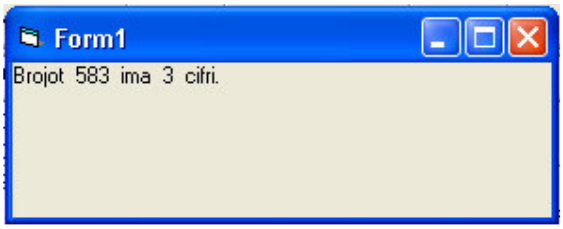
Задача 8. *Одреди од колку цифри е составен бројот* n.

Идејата на решението на оваа задача е цифрите да се бројат од назад, односно почнувајќи од цифрата на единици. Тоа може да се направи така што бројот n целобројно се дели (\) со 10 се додека не се добие за резултат 0. Но, за да променливата n си ја задржи својата почетна вредност, се воведува нова променлива m која се менува при секое извршување на блокот наредби во циклусот Do…Loop. Така ја добиваме следната програма:

```
Dim n, m As Long
Dim br As Integer
Private Sub Form_Activate()
   n = Val(InputBox("Vnesi priroden broj, n="))
   br = 0
   m = n
   Do While m <> 0
      m = m \ 10
      br = br + 1
   Loop
   Print "Brojot "; n; " ima "; br; " cifri."
End Sub
```

Задача 8.

Да се потсетиме уште еднаш која беше операцијата целобројно делење? Оваа операција за резултат го дава количникот при делењето на два броја. Како ќе работи програмата? На пример, ако $n=583$, тогаш почетните вредности за m и br се $m=583$ и $br=0$, и при првото извршување на блокот наредби во циклусот ќе се добијат следните вредности $m=583\backslash 10=58$ и $br=0+1=1$. Потоа се проверува условот $m<>0$, односно дали m е различно од 0? Бидејќи тој услов е исполнет се преминува на повторо извршување на блокот наредби во циклусот, и сега се добиваат следните вредности $m=58\backslash 10=5$ и $br=1+1=2$. Бидејќи повторно е исполнет условот $m<>0$, пак се извршува блокот наредби во циклусот и се добиваат вредностите $m=5\backslash 10=0$ и $br=2+1=3$. Се преминува на повторна проверка на условот $m<>0$. Но, овој пат тој не е исполнет и затоа се прескокнуваат блокот наредби во циклусот и се извршува наредбата после Loop, а тоа е печатење на формата дека бројот n, т.е. бројот 583 има br, т.е. 3 цифри. Се добива следниот излез:



Пребројувањето на цифрите на овој начин ни овозможува истите и да ги издвојуваме. Издвојувањето на цифрите ни е потребно за следната задача.

Задача 9. *Еден број е* ***палиндром****, ако е еднаков на бројот запишан со истите цифри но во обратен редослед. Провери дали внесен број n е палиндром?*

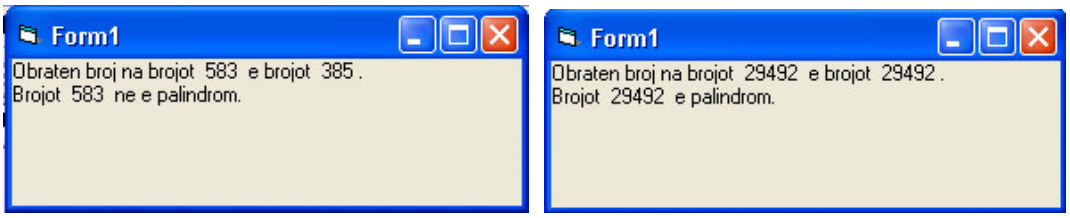
На кратко објаснета идејата на решението е следната. Ги издвојуваме цифрите на бројот n почнувајќи од назад како остатоци при делење (Mod) со 10, и при тоа обратниот број на бројот n го формираме така да при секое извршување на блокот наредби во циклусот обратниот број го множиме со 10 и му ја додаваме издвоената цифра. Програмата што се добива е следната:

```
Dim n, m, obraten As Long
Dim cifra As Integer
Private Sub Form_Activate()
   n = Val(InputBox("Vnesi priroden broj"))
   obraten = 0
   m = n
   Do
      cifra = m Mod 10
      obraten = obraten * 10 + cifra
      m = m \ 10
   Loop While m <> 0
   Print "Obraten broj na brojot "; n; " e brojot "; obraten; "."
   If n = obraten Then
      Print "Brojot "; n; " e palindrom."
   Else
      Print "Brojot "; n; " ne e palindrom."
   End If
End Sub
```

Задача 9.

Во оваа програма се користевме со третиот облик на циклусот Do…Loop. Што мислите зошто? Потребно беше и последно издвоената цифра да се додаде на обратниот број пред променливата m да добие вредност 0 со што се става и крај на циклусот.

А ова е тоа што би го добиле за излез доколку внесете за n еднаш бројот 583, а втор пат бројот 29492:



Забелешка за крај на задачите 8 и 9 е дека во програмите на овие задачи, на местото од While $m <> 0$ може да го користите и обликот Until $m = 0$.

Како што ви навестивме на почетокот, втората најчеста примена на циклусот Do…Loop е при наоѓањето на најголемиот заеднички делител на два броја.

Задача 10. *Најди го најголемиот заеднички делител на броевите* a *и* b.

Сигурно ви се познати некои од начините за барање на најголем заеднички делител на два броја, било да е тоа преку заедничките делители или Евклидовиот алгоритам. Секој од овие начини кои се покажале за рачно пресметување брзи и ефикасни, програмите базирани на нив би биле комплицирани. **Едноставните програми се базираат на едноставни идеи.** За оваа задача, едноставна е идејата да се испитуваат сите броеви на назад почнувајќи од помалиот од броевите a и b се додека не се стигне до број кој ги дели истовремено и двата броја a и b. На пример, ако $a = 45$ и $b = 18$, тогаш се проверуваат по редослед броевите 18, 17, 16, 15, ... се додека не се дојде до бројот 9 кој е првиот број во таа низа броеви кој ги дели истовремено броевите 45 и 18. Програмата изгледа вака:

```
Dim a, b, nzd As Integer
Private Sub Form_Activate()
   a = Val(InputBox("Vnesi go prviot broj, a="))
   b = Val(InputBox("Vnesi go vtoriot broj, b="))
   If a <= b Then nzd = a Else nzd = b
   Do While a Mod nzd <> 0 Or b Mod nzd <> 0
      nzd = nzd - 1
   Loop
   Print "NZD("; a; ","; b; ")="; nzd
End Sub
```

Задача 10.

Обидете се самите да направите слична на оваа програма, но овој пат за наоѓање на најмалиот заеднички содржател. Упатство: За почетна вредност за nzs земете го поголемиот од броевите и потоа наголемувајте го за 1 се додека не добиете дека тој се дели истовремено и со двата броја. Или пак откако сте го нашле најголемиот заедночки делител исористете го математичкото равенство дека $nzd(a,b)\cdot nzs(a,b)=a\cdot b$. Среќно!

Како за крај на овој дел погледнете ја и следната задача.

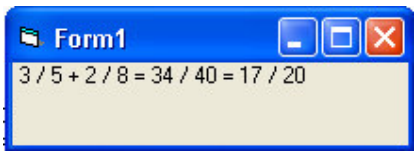
Задача 11. *Собери ги дропките* $\frac{a}{b}$ *и* $\frac{c}{d}$, *каде* a,b,c,d *се цели броеви при што* $b,d\neq 0$. *Резултатот прикажи го во облик на нескратлива дропка.*

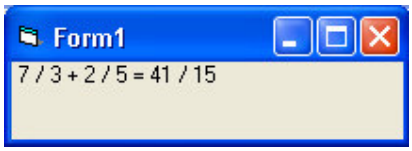
Најнапред проверуваме дали при внесувањето на броевите a,b,c,d, внесено е за b или за d нула. Ако е така, да се појави порака (со функцијата MsgBox) и да сопре програмата (со наредбата End). Понатаму, користејќи го равенството $\frac{a}{b}+\frac{c}{d}=\frac{a\cdot d+b\cdot c}{b\cdot d}$, би ги собрале првично дадените дропки, резултатот би бил дропка, но ништо не ни гарантира дека ќе биде нескратлива. Затоа, ќе треба потоа броителот $br=a\cdot d+b\cdot c$ и именителот $im=b\cdot d$ да ги поделиме со нивниот најголем заеднички делител за да резултатот биде нескратлива дропка. Така, ја добиваме следната програма:

```
Dim a, b, c, d, br, im, nzd, zd, br1, im1 As Integer
Private Sub Form_Activate()
   a = Val(InputBox("a="))
   b = Val(InputBox("b<>0, b="))
   c = Val(InputBox("c="))
   d = Val(InputBox("d<>0, d="))
   If b = 0 Or d = 0 Then
      MsgBox ("Ne moze imenitelot na dropka da e nula")
      End
   End If
   Print a; "/"; b; "+"; c; "/"; d; "=";
   br = a * d + b * c : im = b * d
   Print br; "/"; im;
   If br <= im Then nzd = br Else nzd = im
   Do While br Mod nzd <> 0 Or im Mod nzd <> 0
      nzd = nzd - 1
   Loop
   If nzd <> 1 Then
      br1 = br / nzd
      im1 = im / nzd
      Print "="; br1; "/"; im1
   End If
End Sub
```

Задача 11.

Ако по првото стартување на програмата внесеме a=3, b=5, c=2, d=8, а при повторно стартување на програмата внесеме a=7, b=3, c=2, d=5, ќе ги добиеме следните излези:





А сега, тестирај се кој побрзо ќе собере две дропки, ТИ или VISUAL BASIC. ☺

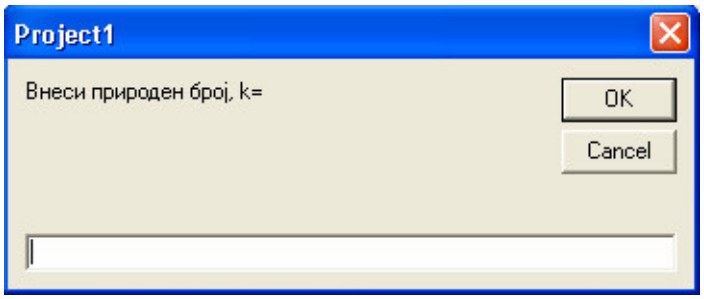
Во третата лекција "*Променливи и операции во Visual Basic*", која ја објавивме во третиот број на Нумерус од минатата учебна година (Нумерус XXXII-3), го споменавме типот на променливи **String**, кој се однесува на променливи кои складираат текстови, односно текстуални низи. Во овој број ќе се запознаеме токму со овој тип на променливи, со операциите и со некои од функциите кои делуваат на нив.

Лекција 6.
Текстуални низи (Strings)

Со текстовите, односно текстуалните низи, не е прв пат да се среќаваме. Сетете се дека при користење на функцијата за влез InputBox, на пример во следниот облик

k = Val(InputBox("Внеси природен број, k="))

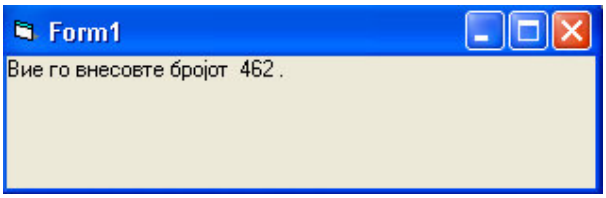
бараме во комуникацискиот прозорец на оваа функција да се појави текстот во наводниците, односно следниот прозорец



Понатаму, сетете се дека често при излез на резултатите од програмата, имавме потреба да со наредбата Print испишеме нешто на формата, како на пример

Print "Вие го внесовте бројот "; k; "."

Па, ако претходните два програмски реда се наоѓаат едно по друго, и ако по стартувањето го внесеме бројот 462, ќе го добиеме следниот излез на формата



што значи дека на формата се појавил текстот во наводниците (под **текст** или **текстуална низа** подразбираме низа од знаци меѓу кои букви, празни места, интерпукциски и специјални знаци) заедно со вредноста на променливата k. Истиот излез може да се добие и ако наместо последниот програмски ред ги запишеме следните програмски редови

```
izlez = "Вие го внесовте бројот " & Str(k) & "."
Print izlez
```

Што се случило? Прво, да забележиме дека употребивме нова функција **Str**, која е обратна на функцијата **Val** (функција која најчесто се користи заедно со функцијата за влез InputBox, за да текстуалната вредност внесена во текстуалното бело поле на комуникацискиот прозорец на функцијата InputBox, се претвори во нумеричка вредност). Обратниот процес го обезбедува функцијата Str, односно тука во нашиот пример, нумеричката вредност на променливата k ја претвара во текстуална вредност. Новововедената променлива izlez е формирана така што трите текстуални вредности "Вие го внесовте бројот ", Str(k) и "." се слепуваат една за друга со помош на **операцијата слепување (&)**. Всушност, од основните операции, наведени во третата лекција, освен оваа операција за слепување, и **операцијата собирање (+)** исто така делува на текстуалните вредности и променливи и при тоа го дава истиот резултат како операцијата за слепување.

Да забележиме и тоа дека ако сакаме излезот на формата да биде на македонска кирилица, ќе треба пред пишувањето на програмските редови кои содржат текст во наводници да ја смениме англиската латиница (EN) во македонска кирилица (MK). Ако овие две подршки се инсталирани на сметачот, тогаш на едноставен начин со ALT+SHIFT (прво се претиска копчето ALT од тастатурата и додека тоа е претиснато се претиска копчето SHIFT од тастатурата) се преминува од една во друга исталирана подршка. Која подршка е активна во моментот е прикажано во десниот крај од лентата на задачи на Windows.

Во истиот број на Нумерус (XXXII-3), кој го спомнавме погоре, пишувавме и за начини на декларирање на променливите, со помош на зборчето **Dim** во делот General Declarations. Текстуалните променливи се декларираат на два начина:

1) **Dim** *именапроменлива* **As String *** *n*

2) **Dim** *именапроменлива* **As String**

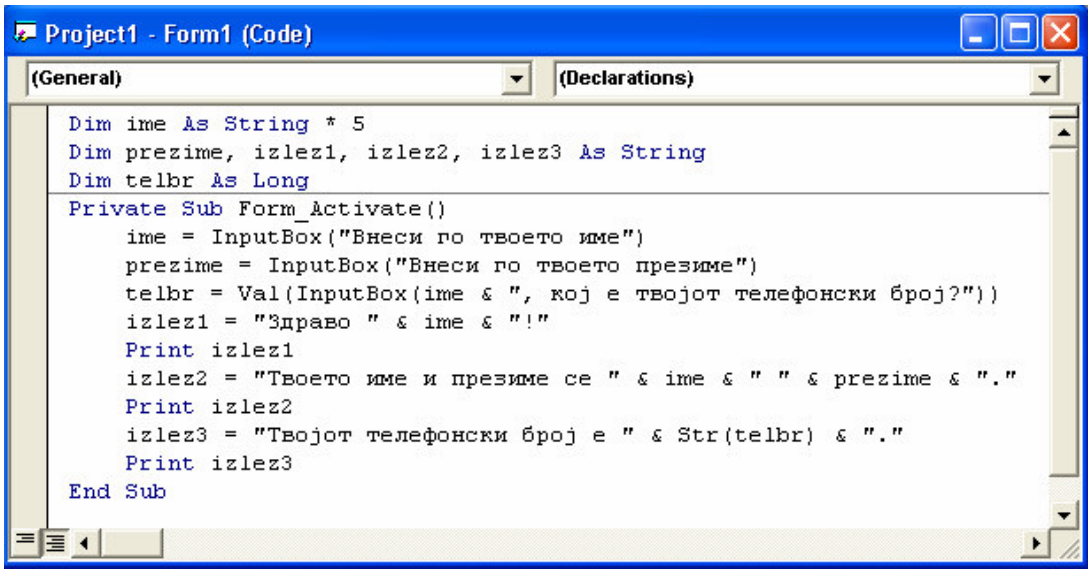
Првиот начин на декларирање подразбира дека променливата има фиксна должина *n* одредена од страна на програмерот, на пример ако за променливата ime сакаме да обезбедиме места за 5 знаци, ќе напишеме

Dim ime **As String *** 5

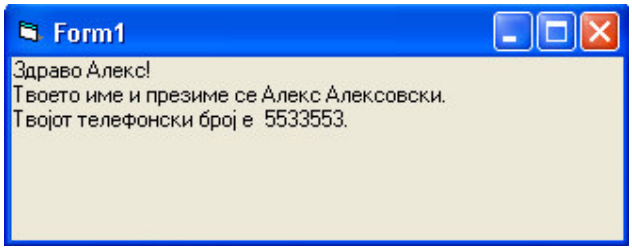
Ова значи дека, при внесување на било кои вредности за променливите ime, ќе се складираат само првите 5 знака. Доколку се внесат помалку од 5 знака, тогаш останатите места ќе се пополнат со празни места (што исто така претставува знак, како дел од текст, односно текстуална низа). Да забележиме дека максималната должина која може програмерот да ја обезбеди при декларирање на текстуалните променливи со фиксна должина е 2^{16} знака.

Кај вториот начин на декларирање нема ограничувања од страна на програмерот за бројот на знаци, односно должината на текстот, освен основното програмско ограничување од приближно 2 билиона (поточно 2^{31}) знака.

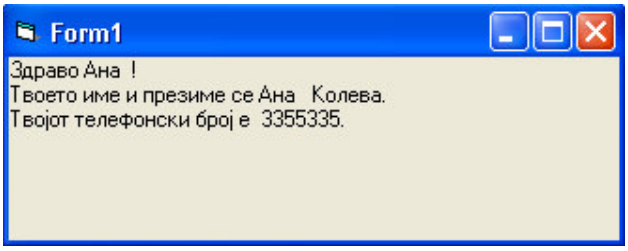
Со следниот пример ќе ги согледаме разликите во декларирањето на текстуалните променливи, а воедно и ќе ја примениме операцијата слепување (&), при испишување на текст во комуникацискиот прозорец на функцијата InputBox и текст на формата. Напишете ги следните програмски редови во делот за кодот



По стратувањето на ВБ-проектот, внесете за име Александар, за презиме Алексовски, за телефонски број 5533553. Треба да го добиете следниот излез:



Ако пак внесете за име Ана, за презиме Колева, за телефонски број 3355335, треба да го добиете следниот излез:



Дали ја согледавте разликата во декларирањето на променливата ime и променливата prezime преку излезот на формата при првиот и вториот влез?

Препорака: Кога не сте сигурни за должината на текстуалните променливи, декларирајте ги текстуалните променливи со вториот начин, односно не ставајте ограничување на бројот на знаци за вредноста на текстуалната променлива!

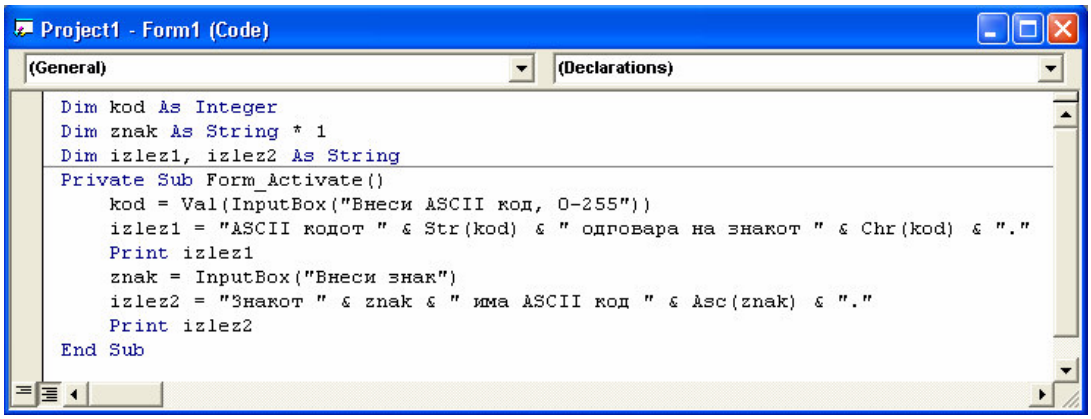
6.1. ASCII карактерното множество, фунциите Chr и Asc

Кога зборуваме за текстови, односно текстуални низи, како што рековме погоре, подразбираме низа од знаци. Секој знак (буква, интерпукциски знак, па дури и празно место)

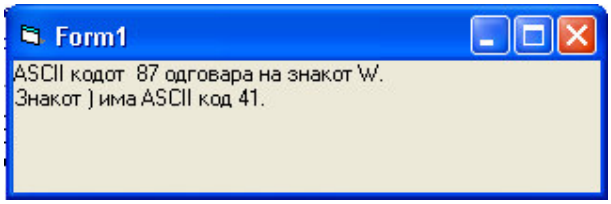
има свој **ASCII код** (American Standard Code for Information Interchange), т.е. број од 0 до 255. Така на пример, буквата "A" има ASCII код 65, додека буквата "a" има ASCII код 97, празното место " " има ASCII код 32. ASCII кодовите на сите знаци, односно **ASCII карактерното множество**, се наоѓа во Help-от на Visual Basic.

Функциите **Chr** и **Asc** ни овозможуваат премин од ASCII код во знак и обратно. Така на пример, Chr(65) прима вредност "A", Chr(97) прима вредност "a", Chr(32) прима вредност " ", додека Asc("A") прима вредност 65, Asc("a") прима вредност 97, Asc(" ") прима вредност 32.

Внесете ги следните програмски редови во прозорецот на кодот, за да направите ВБ-проект за конвертирање на ASCII кодови во знаци и обратно:



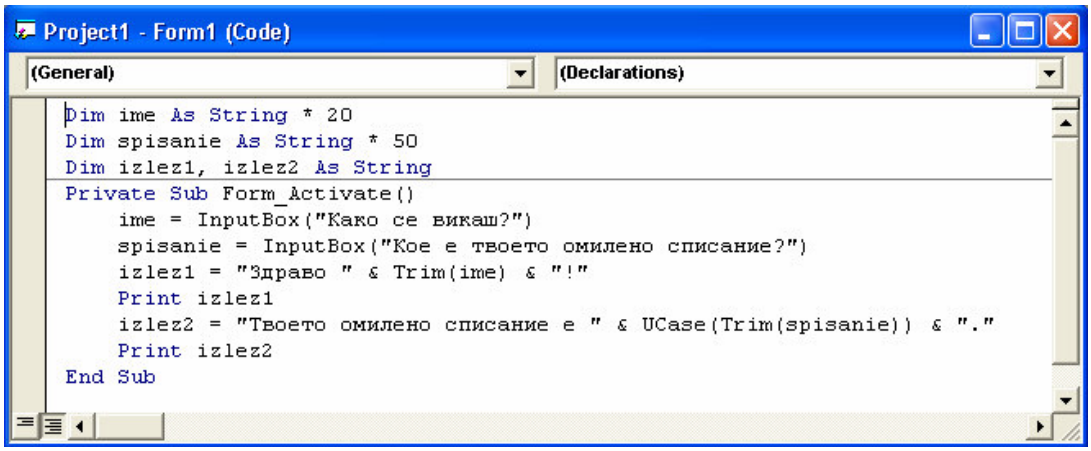
Откако ќе внесете вредност 87 за kod и вредност ")" за знак (само без наводниците, инаки за знак ќе бидат земени само левите наводници), треба да го добиете следниот излез:



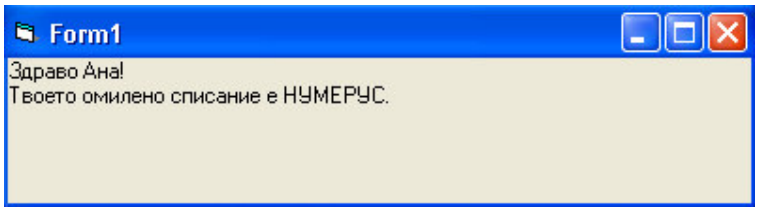
Тестирајте ја програмата и за останатите ASCII кодови и знаци.

6.2. Функциите UCase, LCase, LTrim, RTrim и Trim

Понекогаш иако за текстуалната променлива имаме обезбедено фиксен број на знаци, што најчесто се прави заради помало место во меморијата, вредностите која таа најчесто би ги примала би можело да бидат со многу помал број на знаци. Така на пример, ако за променливата ime обезбедиме место за 20 знаци, сепак најчесто имињата се со помалку од 10 знака. Во овој случај при понатамошна употреба на вредностите од ваквите променливи, препорачливо е да се отстрануваат празните места на почетокот, на крајот или и на почетокот и на крајот од вредноста на текстуалната променлива. Имено, непотребни празни места на почетокот може да се јават при внесување на вредноста во текстуалното поле од комуникацискиот позорец на функцијата InputBox. Отстранувањето на празните места на почетокот, на крајот или и на почетокот и на крајот се прави со функциите **LTrim**, **RTrim** и **Trim** соодветно. Ќе ја увидиме нивната примена со следниот ВБ-проект. Внесете ги следните програмски редови во прозорецот на кодот:



Ако по стартувањето на ВБ-проектот внесете како одговор на првото прашање Ана, а како одговор на второто прашање што друго ако не Нимерус, тогаш треба да го добиете следниот излез:



Сигурно забележавте дека при испишувањето на името Ана иако за него беа обезбедени 20 знака, а тоа содржи само 3 знака и би требало останатите знаци да бидат празни места, сепак при испишувањето на името на формата, веднаш по него се јавува знакот "!" (само без наводници секако), што значи дека функцијата Trim во програмскиот ред

izlez1 = "Здраво " & Trim(ime) & "!"

ги избришала празните места и од лево и од десно и името се појавило такво какво што сме го внеле.

За разлика од него, името на омиленото списание го внесовме со голема почетна буква и останатите мали, а на формата тоа се појави со сите големи букви. *Што се случило?* Погледенте го програмскиот ред

izlez2 = "Твоето омилено списание е " & UCase(Trim(spisanie)) & "."

Овој пат, покрај отстранувањето на празните места со Trim(spisanie), со помош на функцијата **UCase**, поточно со UCase(Trim(spisanie)) сите букви се претвориле во големи букви. На сличен начин делува и функцијата **LCase**, која сите букви ги претвара во мали букви.

Забелешка: Празните места кои се наоѓаат во внатрешноста **не може** да се отстранат со ниедна од функциите LTrim, RTrim и Trim, на пример

LTrim(" a b c ") дава вредност "a b c ",

RTrim(" a b c ") дава вредност " a b c",

Trim(" a b c ") дава вредност "a b c".

Функциите UCase (од англискиот збор "upper case" што значи "голема буква") и LCase (од англискиот збор "lower case" што значи "мала буква") делуваат само на буквите, што значи дека останатите знаци остануваат непроменети, на пример

UCase("Abc*Def+ k") дава вредност "ABC*DEF+ K",

LCase("Abc*Def+ k") дава вредност "abc*def+ k".

6.3. Функциите Len, Left, Right и Mid

Најкористени функции со текстуални низи за решевање на посложени проблеми се функциите Len, Left, Right и Mid. Најнапред ќе ја објасниме употребата на овие функции, а потоа ќе ги примениме истите во конкретни задачи.

Со помош на функцијата **Len** (од англискиот збор "length" што значи "должина") се пресметува вкупниот број на знаци во текстуалната низа од знаци. Нејзината синтакса е

Len (*текст*),

каде *текст* е име на некоја текстуална променлива или некој текст.

Со функциите **Left** (од англискиот збор "left" што значи "лево") и **Right** (од англискиот збор "right" што значи "десно") се издвојуваат *k* последователни знаци од почетокот, односно од крајот на текстуалната променлива или текстот *текст*, синтаксите се

Left (*текст*, *k*),

Right (*текст*, *k*).

Додека пак, со функцијата **Mid** (од англискиот збор "middle" што значи "средина") се издвојуваат *k* последователни знаци од текстуалната променлива или текстот *текст* почнувајќи од *i*-тиот знак, синтаксата е

Mid (*текст*, *i*, *k*).

Задача 12. *Одреди колку пати дадена буква се среќава во даден збор.*

Кога пребројуваме колку пати се среќава некоја буква, не е важно дали е таа мала или голема буква. На пример, во името Александар буквата "а" се среќава три пати, додека знакот "а" се среќава само два пати. За да ги изедначиме по важност малите и големите букви во внесен збор, ќе ја употребиме функцијата Ucase, веднаш по внесувањето на зборот, односно програмата ќе изгледа вака:

```
Dim zbor, bukva, zbor1 As String
Dim bukva1 As String * 1
Dim br, n, i As Integer
Private Sub Form_Activate()
    zbor = InputBox("Vnesi zbor")
    bukva = InputBox("Vnesi bukva")
    zbor1 = UCase(Trim(zbor))
    bukva1 = UCase(Trim(bukva))
    br = 0
    n = Len(zbor1)
    For i = 1 To n
        If Mid(zbor1, i, 1) = bukva1 Then
            br = br + 1
        End If
    Next i
    Print "Во зборот "; zbor; " буквата "; _
        bukva; " се среќава "; br; " пати."
End Sub
```

Задача 12.

Забележете го следниот програмските редови

```
Print "Во зборот "; zbor; " буквата "; _
    bukva; " се среќава "; br; " пати."
```

кој од сите до сега се разликува во тоа што го содржи знакот "_" (секако без наводниците) на крајот од првиот ред. Имено, знакот "_" ги спојува двата реда и програмата нив ги прифаќа како еден програмски ред. Овој начин на пишување на кодот е погоден при долги програмски редови. Но, треба да се има во предвид дека знакот "_" се користи само после завршена целина, односно не може да се користи внатре во некој текст во наводници за негово пренесување во друг ред, или не може да се користи во рамките на некој израз или формула.

Задача 13. *Одреди колку збора има во една реченица.*

Прво откако ќе ги отстраниме празните места од левата и од десната страна на реченицата, со функцијата Trim, ќе преминеме на пребројување на останатите празни места, но не поединечни, туку во групи (една група празни места подразбира последователни знаци за празно место), затоа што знаеме дека зборовите во една реченица се одделени со празни места. Да се потсетиме уште на тоа дека ASCII кодот на празното место е 32, па Chr(32) ни го означува карактерот празно место. Имајќи го во предвид последново, групите празни места ги пребројуваме така што го наоѓаме бројот на последователни знаци празно место и непразно место (`Mid(rec,i,1)=Chr(32) And Mid(rec,i+1,1)<>Chr(32)`), а потоа бројот на зборови е за еден поголем (`br + 1`) од бројот на групи од празни места. Така, го добиваме следниот код:

```
Dim rec As String
Dim br, n, i As Integer
Private Sub Form_Activate()
    rec = InputBox("Внеси реченица")
    rec = Trim(rec)
    br = 0
    n = Len(rec)
    For i = 1 To n - 1
        If Mid(rec, i, 1) = Chr(32) And _
            Mid(rec, i + 1, 1) <> Chr(32) Then br = br + 1
    Next i
    Print "Во реченицата:"
    Print rec
    Print "има "; br + 1; " зборови."
End Sub
```

Задача 13.

Тестирајте ја програмата така да ќе внесете “неидеална” реченица која има празни места и на почеток и на крај, а меѓу два последователни збора има повеќе од едно празно место, на пример реченицата “ Jac ja сакам математиката. ”.

Задача 14. *Провери дали еден збор е палиндром, односно дали е еднаков на зборот напишан со истите букви но во обратен редослед.*

Обратниот збор ќе го формираме така што од дадениот збор ќе ги издвојуваме буквите една по една од крајот и ќе ги слепуваме на обратниот збор. Односно, се добива програмата:

```
Dim zbor, obraten As String
Dim n, i As Integer
Private Sub Form_Activate()
    zbor = InputBox("Внеси збор")
    zbor = UCase(Trim(zbor))
    obraten = ""
    n = Len(zbor)
    For i = n To 1 Step -1
        obraten = obraten & Mid(zbor, i, 1)
    Next i
    If obraten = zbor Then
        Print "Зборот "; zbor; " е палиндром."
    Else
        Print "Зборот "; zbor; " не е палиндром."
    End If
End Sub
```

Задача 14.

Visual Basic дава големи можности за работа со текстови. Ние ви ги издвоивме најосновните и најкористените функции, што не значи дека останатите се помалку корисни. "Чепкајќи" по Help-от може да се најдат и останатите функции, што ако се разработат убаво може да доведат до пократки решенија на задачите. Така на пример, постои функција **StrReverse** со синтакса

StrReverse (*текст*),

која директно го дава обратниот запис на текстуалната низа или текстот *текст*. Замислете, колку кратко и едноставно! Така, последната "класична" задача во работата со текстуални низи, на најкраткиот можен начин изгледа вака:

```
Dim zbor As String
Private Sub Form_Activate()
    zbor = InputBox("Внеси збор")
    zbor = UCase(Trim(zbor))
    If StrReverse(zbor) = zbor Then
        Print "Зборот "; zbor; " е палиндром."
    Else
        Print "Зборот "; zbor; " не е палиндром."
    End If
End Sub
```

Задача 14'.

Сепак, и Help-от бил од некаква корист! ☺

Од лекција во лекција, се обидуваме вашето познавање на програмскиот јазик Visual Basic да го збогатиме со по некоја нова содржина. Секој пат ве запознаваме со нова алатка, било тоа да е некоја нова наредба или некоја нова тематика, со која се решаваат нови типови на задачи или веќе познатите задачи се решаваат на поинаков начин. Така, овој пат ќе се дружиме со низите од променливи и со нивна помош успешно ќе решиме неколку нови задачи.

Лекција 7.
Низи од променливи (Arrays)

Често пати при решавањето на задачите имаме потреба да работиме со поголем број на променливи и тогаш најефикасно е тие променливи да ги сместиме во низа.

Низите од променливи, исто како и нумеричките и текстуалните променливи, се декларираат во делот General Declarations од кодот на ВБ-проектот. Нивниот начин на декларирање се разликува во зависност од видот на низата, односно дали се тоа **низи со фиксна големина** или **динамички низи**.

Кога уште на почетокот го знаеме *името на низата*, *типот на нејзините елементи*, *нејзината димензија* и *бројот на променливи во секоја од димензиите*, тогаш таквата низа ја нарекуваме **низа со фиксна големина** и може да ја декларираме веднаш. На пример, со декларацијата

Dim broj(1 To 4) As Integer

во меморијата обезбедуваме место за низата broj (тоа е нејзиното име) чии елементи broj(1), broj(2), broj(3) и broj(4) се од типот Integer, односно цели броеви, нејзината димензија е еден (елементите зависат само од еден индекс, тоа е бројот во заградата) и бројот на елементи во рамките на таа димензија е 4 (со индекси од 1 до 4).

Забелешка. Во оваа лекција ќе работиме само со еднодимензионалните низи заради типот на задачите кои сакаме да ви ги изложиме.

Низите со фиксна големина во пракса поретко се користат затоа што не секогаш од самиот почеток знаеме со колкав број на променливи треба да работиме, тоа најчесто се остава на самиот корисник на програмата да го внесе. За да ја направиме програмата поприлагодлива на корисникот треба да користиме **динамички низи** кај кои на почетокот во General Declarations ги декларираме само *името и типот на променливите*, на пример

Dim broj() As Integer

а подоцна во телото на процедурата откако ќе ги "дознаеме" *нејзината димензија и бројот на променливи во секоја од димензиите*, динамичката низа ја додекларираме. На пример, откако сме го "прашале" корисникот колку природни броеви сака да внесе и неговиот одговор сме го складирале во променливата *n*, ја додекларираме низата broj на следниот начин

ReDim broj(1 To n) As Integer.

Ќе ја разгледаме употребата на динамичките низи и воопшто низите од променливи за почеток преку следната задача.

Задача 15. *Најди ги најмалиот и најголемиот број меѓу n реални броеви.*

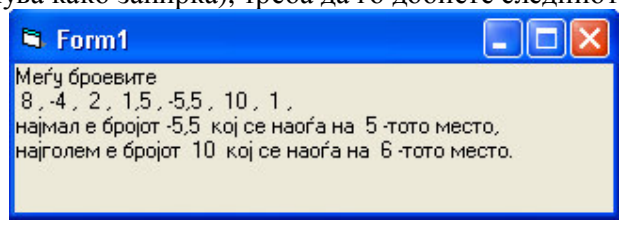
Реалните броеви ги сместуваме во низата од променливи a(1), a(2), …, a(n). Променливите min и max ќе ги означуваат соодветно најмалиот и најголемиот елемент од низата, а indexmin и indexmax се променливи кои ќе го изначуваат индексот на најмалиот односно најголемиот елемент од низата. Идејата е да по внесувањето на елементите во низата, претпоставиме дека првиот внесен елемент е најмал, односно најголем, а потоа со споредување на min, односно max со останатите елементи го најдеме вистинскиот најмал, односно најголем елемент. Така ја добиваме следната програма:

```
Dim a() As Double
Dim n, i, indexmin, indexmax As Integer
Dim min, max As Double
Private Sub Form_Activate()
    n = Val(InputBox("Внеси го бројот на реални броеви, n="))
    ReDim a(1 To n) As Double
    Print "Меѓу броевите"
    For i = 1 To n
        a(i) = Val(InputBox("a(" & i & ")="))
        Print a(i); ", ";
    Next i
    Print
    min = a(1): indexmin = 1
    max = a(1): indexmax = 1
    For i = 2 To n
        If a(i) < min Then
            min = a(i)
            indexmin = i
        End If
        If a(i) > max Then
            max = a(i)
            indexmax = i
        End If
    Next i
    Print "најмал е бројот "; min; " кој се наоѓа на "; _
      indexmin; "-тото место,"
    Print "најголем е бројот "; max; " кој се наоѓа на "; _
      indexmax; "-тото место."
End Sub
```

Задача 15.

Ако по стартувањето на програмата внесете за бројот на реални броеви 7, а потоа ги внесете еден по еден 7-те реални броеви 8, -4, 2, 1.5, -5.5, 10, 1 по тој редослед (внимавајте на

тоа да децималните броеви ги внесувате со точка на местото од децималната запирка иако подоцна таа се прикажува како запирка), треба да го добиете следниот излез:



Оваа задача сама за себе нема толку големо значење, но барањето на намал и најголем елемент од некоја низа има најголеме примена во алгоритмот за подредување на броеви по големина што е предмет за размислување во следната задача.

Задача 16. *Подреди ги по големина во растечки редослед (од најмалиот кон најголемиот) внесените реални броеви.*

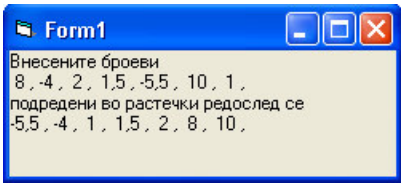
Алгоритмот кој ќе го користиме е следниот. Во првиот чекор се бара најмалиот број меѓу сите броеви и тој број го ставаме на прво место во низата, а на неговото место го ставаме бројот што бил на првото место. Потоа, во вториот чекор го бараме најмалиот број почнувајќи од вториот па до последниот и повторно ги заменуваме местата сега на вториот и најмалиот број. Оваа постапка ја повторуваме се до последниот број. Ајде да се обидеме претходните два чекори да ги искажеме во термин на *i*-ти чекор, тоа ќе ни помогне за пишување на кодот на програмата. Значи, во *i*-тиот чекор го бараме најмалиот број почнувајќи од *i*-тиот па до последниот *n*-ти број (ако променливата *n* ни го означува вкупниот број на реални проеви кои сакаме да ги подредиме), на начин како во помошната задача 15, и потоа ги заменуваме местата на *i*-тиот број со најмалиот број min најден во тој чекор. Програмските редови на овој алгоритам се следните:

```
Dim a() As Double
Dim b() As Double
Dim n, i, j, indexmin As Integer
Dim min, pb As Double
Private Sub Form_Activate()
    n = Val(InputBox("Внеси го бројот на реални броеви, n="))
    ReDim a(1 To n) As Double
    ReDim b(1 To n) As Double
    Print "Внесените броеви"
    For i = 1 To n
        a(i) = Val(InputBox("a(" & i & ")="))
        b(i) = a(i)
        Print a(i); ", ";
    Next i
    Print
    For i = 1 To n
        min = b(i): indexmin = i
        For j = i + 1 To n
            If b(j) < min Then
                min = b(j)
                indexmin = j
            End If
        Next j
        pb = b(i): b(i) = min: b(indexmin) = pb
    Next i
    Print "подредени во растечки редослед се"
    For i = 1 To n
        Print b(i); ", ";
    Next i
End Sub
```

Задача 16.

Употребата на втора низа од променливи b е за да се зачува внесената низа a, додека низата b постојано се менува за да на крајот кај неа броевите се подредени по големина и нејзината содржина е таа што ја печатиме. Променливата pb е помошна променлива која привремено во себе ја складира вредноста на *i*-тиот елемент b(i) за да потоа таа вредност ја предаде на елементот во низата b кој се наоѓа на местото каде е најден најмалиот елемент, односно елементот b(indexmin) ја превзема вредноста од pb, и секако на *i*-тото место b(i) доаѓа вредноста на најмалиот елемент min или b(indexmin).

Тестирајте ја програмата на 7-те броеви од тестирањето на претходната задача и треба да го добиете следниот излез:



А сега, обидете се самите да ја прилагодите оваа програма за подредување по големина *n* внесени реални броеви, но во опаѓачки редослед (почнувајќи од најголемиот кон најмалиот). Ако го разбравте претходно изложениот алгоритам, а се надеваме дека е така, нема да ви биде тешко.

Ќе преминеме на решавање на конкретни математички задачи со помош на алатката – низи од променливи.

Задача 17. *Најди го најголемиот заеднички делител (НЗД) на n природни броеви.*

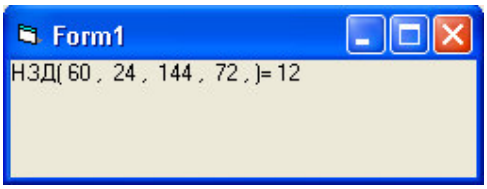
Во задачата 10 од Лекцијата 5 во Нумеруасот XXXIII-1 ја изложивме идејата за барање на најголемиот заеднички делител на два броја. Истата идеја ќе ја користиме и овде за броевите a(1), a(2), …, a(n). Најнапред го наоѓаме најмалиот од овие броеви и тој го земаме за почетна вредност на променливата nzd. Понатаму вредноста на nzd ја намалуваме во секој циклус за еден се додека не добиеме дека секој од броевите a(1), a(2), …, a(n) е делив со nzd. Така сме го добиле најголемиот заеднички делител. Програмата е следната:

```
Dim a() As Integer
Dim n, i, p, nzd, min As Integer
Private Sub Form_Activate()
    n = Val(InputBox("Внеси го бројот на природни броеви, n="))
    ReDim a(1 To n) As Integer
    Print "НЗД(";
    For i = 1 To n
        a(i) = Val(InputBox("a(" & i & ")="))
        Print a(i); ", ";
    Next i
    min = a(1)
    For i = 2 To n
        If a(i) < min Then min = a(i)
    Next i
    nzd = min
    p = 1
    Do While p = 1
        p = 0
        For i = 1 To n
            If a(i) Mod nzd <> 0 Then p = 1
        Next i
        If p = 1 Then
            nzd = nzd - 1
        End If
    Loop
    Print ")="; nzd
End Sub
```

Задача 17.

Во програмата променливата p служи како "покажувач". Имено ако таа има вредност 1 тогаш се дозволува намалувањето на вредноста на nzd затоа што се нашол барем еден број од низата a(1), a(2), …, a(n) кој не е делив со nzd. Ако вредноста на "покажувачот" p остане 0 на крајот на циклусот Do…Loop тогаш тоа значи дека најголемиот заеднички делител е најден.

Ако ја тестирате програмата за 4-те природни броеви 60, 24, 144 и 72, треба да го добиете следниот излез:



Ние би ви препорачале да ја направите сами програмата за наоѓање на најмалиот заеднички содржател (НЗС) на *n* природни броеви. На сличен начин, со тоа што сега ќе почнете од најголемиот од броевите a(1), a(2), …, a(n) и ќе го зголемувате за 1 се додека не добиете број кој е делив со секој од броевите a(1), a(2), …, a(n). Лесно, нели?

Во Нумерусот XXXII-2 ви ја изложиваме задачата за наоѓање на сите прости броеви до природниот број *n* на тој начин што ги броевме делителите на секој број од 1 до *n* и ако бројот има точно 2 делители, тогаш тој е прост и го печатевме на формата. Овој пат ќе ја решиме истата задача, но на поинаков начин.

Задача 18. *Најди ги сите прости броеви до природниот број n со помош на Ератостеново сито.*

Ератостен (276 п.н.е – 194 п.н.е) бил грчки математичар, поет, атлетичар, географ и астроном. Меѓу своите современици бил познат под надимакот "бета" (грчкиот број два) затоа што во многу области го сметале за втор човек на Медитеранот. Нему му се препишува системот на земјините координати со географските ширини и должини, а воедно тој е и првиот познат научник кој ја пресметал обиколката на Земјата на многу досетлив начин.

Меѓу останатите откритија Ератостен го смислил е методот за издвојување на простите броеви меѓу природните броеви, познат како **Ератостеново сито**. Овој метод се состои во тоа што прво се испишуваат сите броеви од 2 до бројот *n* (не се пишува бројот 1 затоа што тој не е ниту прост ниту сложен). Во првиот чекор се означува бројот 2 дека е прост и се прецртуваат сите броеви поголеми од 2 и деливи со 2. Во вториот чекор се означива бројот 3 дека е прост и се прецртуваат сите броеви поголеми од 3 и деливи со 3. Следниот непрецртан број е 5, се означува дека е прост и се прецртуваат сите броеви поголеми од 5 и деливи со 5. Оваа постапка се повторува заклучно до бројот $[\sqrt{n}]$ (цел дел од $\sqrt{n}$), затоа што после него сите броеви до бројот *n* кои останале непрешкртани сигурно се прости.

На пример, да ги најдеме сите прости броеви до природниот број 50. Постапката со прешкртување на број може да ја замениме и со запишување 0 на неговото место. Бидејќи $[\sqrt{50}] = [7{,}07106781...] = 7$, ќе имаме 4 чекори на "прешктрување" во однос на деливост со броевите 2, 3, 5 и 7 соодветно, а во последниот 5-ти чекор ги означуваме и преостанатите "непрешкртани" броеви дека се прости. Така, добиваме дека сите прости броеви до бројот 50 се: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 35, 27, 41, 43 и 47.

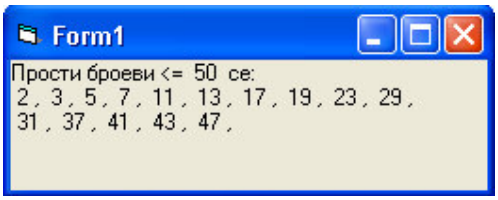
"Прешкртувањето" во програмата ќе го изведеме токму на овој начин, со доделување на 0 на членот на низата кој е делив со бројот со кој испитуваме деливост во тековниот чекор (односно со следниот непрецртан број). Така ја добиваме следната програма:

```
Dim a() As Integer
Dim n, i, j, br As Integer
Private Sub Form_Activate()
    n = Val(InputBox("Внеси природен број n="))
    ReDim a(2 To n) As Integer
    For i = 2 To n
        a(i) = i
    Next i
    For i = 2 To Int(Sqr(n))
        If a(i) <> 0 Then
            For j = i + 1 To n
                If a(j) Mod a(i) = 0 Then a(j) = 0
            Next j
        End If
    Next i
    Print "Прости броеви <= "; n; " се:"
    br = 0
    For i = 2 To n
        If a(i) <> 0 Then
            Print a(i); ", ";
            br = br + 1
            If br = 10 Then
                br = 0
                Print
            End If
        End If
    Next i
End Sub
```

Задача 18.

Да ги дообјасниме најнапред некои од функциите и променливите кои ги користевме во програмата. Функцијата **Sqr** е функција за квадратен корен, а функцијата **Int** е функција за цел дел. Така $[\sqrt{n}]$ во кодот на Visual Basic е Int(Sqr(n)). Променливата br ја користиме да ги брои испишаните прости броеви во еден ист ред на формата, така што ако достигне вредност 10, односно ако сме испишале веќе 10 броја во една редица, тогаш преминуваме на нов ред со празна наредба Print, а на бројачот br му доделуваме вредност 0.

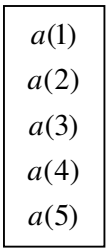
Тестирајте ја програмата за $n = 50$ и ќе го добиете следниот излез:



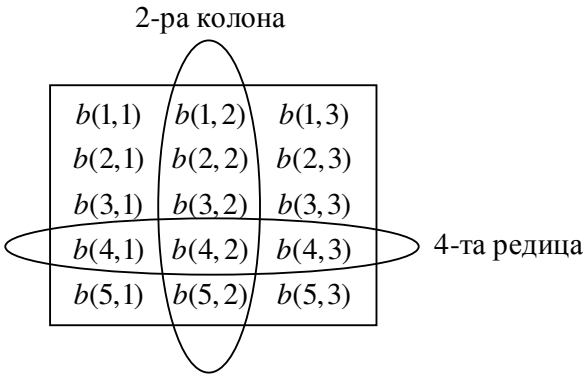
Досетлив начин за барање на простите броеви, нели? А каков ли е тогаш досетливиот начин на кој Ератостен ја пресметал обиколката на земјата? Што мислите вие? Сигурно не тргнал на пат околу светот. ☺

Во претходната лекција 7 ви го изложивме работењето со низи од променливи (Arrays), и се задржавме само на еднодимензионалните низи. Повеќе димензионалните низи даваат простор за складирање на повеќе вредности и начин на нивно подредување. На Скица 1 прикажана е еднодимензионалната низа a која има 5 членови, а на Скица 2 прикажана е дводимензионалната низа b која има $5 \cdot 3 = 15$ членови распоредени според вредностите на нивните индекси, на пример елементот $b(4,2)$ се наоѓа во 4-тата редица во 2-рата колона. Заедничко за низите од променливи е тоа што нивните елементи складират само податоци од ист тип, на пример или сите елементи на низата b се броеви, односно од некој од типовите податоци Integer, Long, Double, Single, или пак сите елементи се зборови, односто од типот на податоци String, и слично.

Не ретко се јавува потребата одредени податоци од различен тип да се сместат во некоја дводимензионална низа. На пример, низата од Скица 2 сакаме да ја искористиме да ги



Скица 1



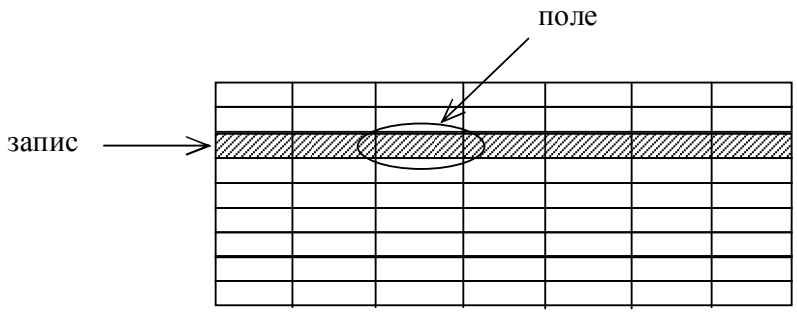
Скица 2

зачуваме имињата, оценките по математика и просечниот успех на 5 ученици. Тоа ќе го направиме ако податоците за секој ученик ги запишеме во една редица, така што во првата колона ќе бидат имињата, во втората оценките по математика и во последната трета колона просечниот успех на ученикот. Кој податок тогаш ќе го содржи елементот $b(4,2)$? Не е тешко да се заклучи дека тоа ќе биде оценката по матемтика на 4-тиот ученик. При запишувањето на податоците на овој начин податоците во првата колона треба да бидат од типот String, во втората од типот Integer, а во третата од типот Double. Кај низите од променливи не е дозволено ова, па доколку ги користиме нив за овој проблем ќе треба да ги дефинираме сите податоци да се од типот Variant, но во тој случај за нив ќе треба да се обезбеди поголемо место во меморијата, а и читањето на нивните вредности според внесени податоци за еден ученик нема да биде толку практично.

Поедноставен и попрегледен начин за правењето на ваквите "списоци" од податоци е работата со датотеки за читање и запишување на податоци во Visual Basic.

Лекција 8.
Работа со датотеки за читање и запишување на податоци

Датотеките за читање и запишување на податоци користат за зачувување на податоци кои подоцна може да се искористат. Запишувањето на податоците во датотеката е слично како кај дводимензионалната низа прикажана на Скица 2. Имено, податоците се запишуваат во редици наречени **записи** или **рекорди**, и при тоа секоја редица содржи еднаков број на **полиња**, како што е прикажано на Скица 3.



Скица 3

Полињата од една иста колона содржат податоци од ист тип. На тој начин секој запис во датотеката е од ист нов **кориснички дефиниран тип на податоци**. Новиот кориснички дефиниран тип на податоци се дефинира со **наредбата Type**, најчесто во нова датотека наречена **стандарден модул** која е со наставка .bas и која се додава на ВБ-проектот преку менито Project кога ќе се одбере наредбата Add Module. За стандардните модули како составен дел од еден ВБ-проект пишувавме во Лекциите 1 и 2. Синтаксата на оваа наредба е

[Private | Public] Type *новтип*
именаелемент **As** *тип*
именаелемент **As** *тип*
...
End Type

каде *новтип* е името на новиот кориснички дефиниран тип на податоци, *именаелемент* е името на елементот кое се додава на името на променливата од новиот кориснички дефиниран тип на податоци, *тип* е некој веќе постоечки тип на податоци. Додавањето на зборчињата Private или Public пред зборот Type не е задолжително, и зависи од тоа дали новиот тип податоци се дефинира во кодот на формата, во стандарден модул и дали тој ќе се користи во секоја од формите како составен дел од ВБ-проектот.

Бидејќи ова е почетна лекција за работа со датотеки за читање и запишување на податоци, ќе се задржиме на што е можно помалку нови поими и начини на работа. Ќе ви ги објасниме само најнеопходните поими, наредби, методи со кои може да се забележи корисноста на датотеките. Затоа, дефинирањето на новиот тип на податоци ќе го правиме во делот General Declarations во кодот на формата Form1 со додавање на зборчето Private пред зборот Type (инаку Visual Basic нема да го прифати новиот кориснички дефиниран тип на податоци). На пример, ако сакаме да креираме датотека во која ќе ги запишуваме податоци за ученици (име и презиме, оценка по математика), тогаш ќе мора однапред да го дефинираме новиот тип на подадоци Ucenik на секој од записите како

```
Private Type Ucenik
    ime As String*30
    ocenka As Integer
End Type
```

Потоа, доколку променливата zapis е променлива од новиот тип на податоци Ucenik, таа и ќе се декларира како таква, односно

Dim zapis **As** Ucenik

и тогаш променливите zapis.ime и zapis.ocenka ќе ги содржат податоците за името и оценката по математика на ученикот соодветно.

Според видот на податоците кои ги содржи и начинот на кој се пристапува кон нив, во Visual Basic разликуваме три вида на датотеки за читање и запишување на податоци: **секвенционални датотеки** (најчесто се користат за текстови, затоа што кај нив податоците се запишани во континуирани блокови), **датотеки со случаен пристап** (каде записите на податоците се со фиксна должина и има директен пристап до секој од записите) и **бинарни датотеки**. Најчесто користени наставки на датотеките за читање и запишување на податоци се наставките .txt и .dat.

Но, за да воопшто може да се работи со било кој вид од датотеките за читање и запишување на податоци треба датотеката да се "отвори", односно да се пристапи до неа. Тоа се прави со **наредбата Open**, чија синтакса е

Open *патека* **For** *мод* **As #***бројнадатотека* [**Len =** *должинаназапис*]

каде *патека* е текстуален израз кој го покажува патот каде се наоѓа датотеката кон која пристапуваме како и нејзиното име (на пример "c:\spisok.dat" значи пристапуваме до датотеката spisok.dat која се наоѓа на c:), *мод* е начинот на пристап кон датотеката и тој може да биде еден од пристапите Append, Binary, Input, Output или Random (доколку датотеката не постои таа ќе се креира само ако е отворена за еден од пристапите Append, Binary, Output или Random), *бројнадатотека* е некој број од 1 до 511 и под кој број понатаму во кодот на програмата се повикува датотеката за да се запише или да се прочита некој запис од неа и *должинаназапис* е некој број помал или еднаков на 32767 (бајти), и овој дел од синтаксата е задолжителен кај датотеките со случаен пристап каде за *должинаназапис* се зема, како што вели и самото име, должината на променливата која го складира во себе записот.

Откако ќе се заврши со работата со отворената датотека, таа треба да се "затвори" со **наредбата Close** која има синтакса

Close [#*бројнадатотека*]

каде *бројнадатотека* е бројот под кој е отворена датотеката и доколку тој се изостави се затвараат сите отворени датотеки.

Заради прегледноста и начинот на кој се пристапува до секој од записите, ќе се задржиме на **датотеките со случаен пристап**. Кон овие датотеки се пристапува со мод Random икога се читаат записи од нив и кога се запишуваат записи во нив. Да ги разгледаме уште **наредбите Put и Get** кои служат за запишување, односно читање на цели записи од отворената датотека. Нивните синтакси се

Put #*бројнадатотека*, *бројназапис*, *именапроменлива*

и

Get #*бројнадатотека*, *бројназапис*, *именапроменлива*

каде *бројнадатотека* е бројот под кој е отворена датотеката, *бројназапис* е редниот број под кој е треба да се запише записот (кај наредбата Put) или под кој е запишан (кај наредбата Get) и *именапроменлива* е името на променливата (која треба да биде декларирана од типот на податоци кој одговара на секој од записите) чија содржина се запишува во соодветниот ред за еден запис (при наредбата Put) или на која и се доделува вредноста од соодветниот запис (при наредбата Get). На пример, со

Put #1, br + 1, zapis

во датотеката отворена под број 1 во редот br+1 се запишува вредноста од променливата zapis, која е од некој кориснички дефиниран тип на податоци и содржи онолку елементи колку што има полиња во секој запис, односно одговара на записите од отворената датотека.

Додека, со

Get #1, i, zapis

од датотеката отворена под број 1 од редот i се чита записот и неговата вредност се доделува на променливата zapis, која исто така треба да биде од корисничкиот тип на податоци кој одговара на записите од отворената датотека.

Особеностите и начините на кои се работи со датотеките со случаен пристап ќе ги согледаме преку неколку конкретни задачи.

Задача 19. Креирај датотека во која ќе ги внесеш имињата на 9 ученици од едно исто одделение и нивните оценки по математика, според податоците дадени во Табела 1.

	име и презиме	оценка
1	Ана Павловска	5
2	Филип Трајковски	3
3	Анета Божиновска	4
4	Марко Јаневски	3
5	Маријан Андреевски	5
6	Билјана Блажевска	3
7	Симона Мицевска	5
8	Емил Стојановски	2
9	Татјана Ристевска	4

Табела 1

Почнете со нов ВБ-проект во кој откако ќе го внесете кодот за оваа задача, неговите составни фајлови ќе ги зачувате (снимате) под имињата **vnesuvanje.frm** и **vnesuvanje.vbp**, бидејќи овој ВБ-проект ќе служи само за додавање на нови записи во датотеката **spisok.dat** која при првото нејзино отварање ќе се креира. Отварањето на датотеката ќе го направиме со програмскиот ред

Open "c:\spisok.dat" For Random As #1 Len = Len(zapis)

односно датотеката spisok.dat ќе ја отвориме за случаен (Random) пристап под број 1 (#1) при што должината на секој запис ќе биде енаква на должината на променливата zapis која претходно ќе ја дефинираме да биде од корисничкиот дефиниран тип на податок Ucenik кој содржи два елементи ime и ocenka (спомнато погоре).

Кај датотеките со случаен пристап при запишување на нов запис треба да се наведе и бројот на редот каде се запишува новиот запис. За таа цел најнапред, веднаш по отварањето на датотеката се пребројува колку записи има во неа со

br = LOF(1) / Len(zapis)

каде LOF(1) ја означува должината на датотеката отворена под број 1 (Length Of File = должина на датотека), додека Len(zapis) е должината на еден запис, и тогаш количникот на овие два броја го одредува бројот br на записи во датотеката. Тогаш, следниот запис го запишуваме на првиот слободен ред, тоа е редот со реден број br+1.

После запишувањето на еден запис, датотеката се затвара и го прашуваме корисникот дали ќе внесува податоци за нов ученик

nov = Val(InputBox("Нов ученик? (1-ДА, 0-НЕ)"))

и доколку тој се одлучи за внесување на уште податоци, треба да внесе 1, и датотеката повторно ја отвараме, а доколку не, треба да внесе 0, и тогаш се затвара целиот ВБ-проект со **наредбата End**. Тоа е програмскиот ред

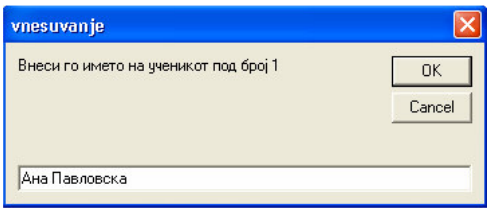
If nov = 1 Then GoTo 100 Else End

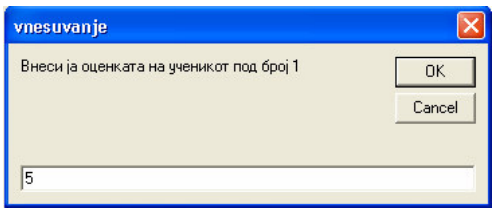
каде Go To 100 не носи на повторно отварање на датотеката. Така ја добиваме програмата:

```
Private Type Ucenik
    ime As String * 30
    ocenka As Integer
End Type
Dim zapis As Ucenik
Dim br, nov As Integer
Private Sub Form_Activate()
100:
Open "c:\spisok.dat" For Random As #1 Len = Len(zapis)
    br = LOF(1) / Len(zapis)
    zapis.ime = InputBox("Внеси го името на ученикот под број " & br+1)
    zapis.ocenka = Val(InputBox("Внеси ја оценката на ученикот под број " & br+1))
    Put #1, br + 1, zapis
Close #1
nov = Val(InputBox("Нов ученик? (1-ДА, 0-НЕ)"))
If nov = 1 Then GoTo 100 Else End
End Sub
```

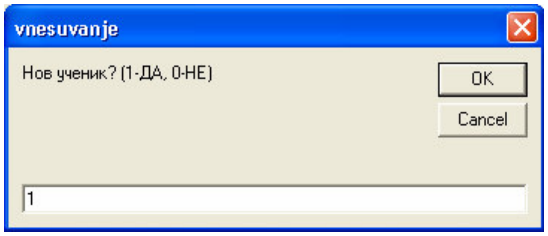
Задача 19.

По стартувањето на оваа, започнете го внесувањето на имињата и оценките на учениците според списокот од Табела 1, при што редните броеви самата програма ќе ги пресметува. По внесувањето на податоците за првиот ученик:





ќе бидете запрашани дали ќе внесувате податоци за нов ученик, одговорете со ДА, односно со 1:



Откако ќе ги внесете податоците за сите 9 ученици (одговарајќи постојано со 1 на прашањето Нов ученик? пред внесувањето на податоците за нов ученик), на крајот одговорете на ова прашање со 0 и програмата ќе се затвори, а податоците ќе останат запишани во датотеката spisok.dat.

Овој ВБ-проект за внесување може да го стартувате и секогаш кога ќе сакате да го дополните списокот со нови ученици и нивните оценки по математика.

Задача 20. Излистај ги податоците од датотеката креирана во задача 19.

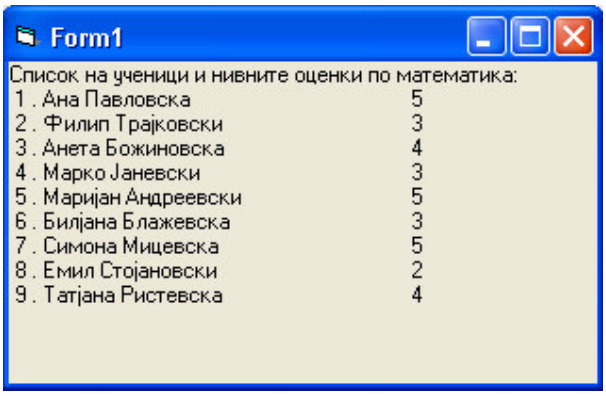
И овој пат почнете со нов ВБ-проект чии составни фајлови ќе ги снимите под имињата **listanje.frm** и **listanje.vbp**. Бидејќи, датотеката spisok.dat е независна од ВБ-проектот за внесување, за да се дојде до нејзините податоци треба само да се запази точната патека до неа (затоа при решавањето на овие задачи ја одбравме патеката "c:\spisok.dat", односно директно на c:).

Листањето на податоците од датотеката ќе го направиме најнапред со читање на запис по запис од датотеката, повторно отворена под број 1 и негово испишување на самата форма, во вид на список со реден број пред името на ученикот (редниот број е бројот на записот), потоа името на ученикот (zapis.ime) и на крајот неговата оценка (zapis.ocenka). Програмските редови за овој ВБ-проект се следните:

```
Private Type Ucenik
    ime As String * 30
    ocenka As Integer
End Type
Dim zapis As Ucenik
Dim br, i As Integer
Private Sub Form_Activate()
Print "Список на ученици и нивните оценки по математика:"
Open "c:\spisok.dat" For Random As #1 Len = Len(zapis)
    br = LOF(1) / Len(zapis)
    For i = 1 To br
        Get #1, i, zapis
        Print i; ". "; zapis.ime, zapis.ocenka
    Next i
Close #1
End Sub
```

Задача 20.

По стартувањето на програмата, доколку правилно сте ги внеле податоците од Табела 1 во датотеката spisok.dat, треба да го добиете следниот излез:



Ако не сте го добиле овој излез, тогаш еден од начините да ја исправите грешката е да почнете се од ново. Прво избришете го фајлот spisok.dat, пристапувајќи директно до него преку My Computer или Windows Explorer, и потоа повторно стартувајте го ВБ-проектот за внесување, па ВБ-проектот за листање, по тој редослед. Овој начин за исправање на направената грешка е сигурен, но многу подолг. Секако, *има пократки начини за исправање на грешките направени при внесување на податоците. Начините зависат од видот на грешката, односно дали треба да се отстранува цел запис, или треба да се промени само некое од полињата во конкретен запис.*

Кодот од овој ВБ-проект за листање се користи и во состав на задачи каде треба да се направат исправки на грешки во списокот, затоа што пред да се даде редниот број на записот каде треба да се направат исправките, треба прво да се излиста целата датотека, да се има увид на редоследот на записите во неа.

Што може да се работи со податоците од една датотека, односно во нашиот случај со списокот на ученици и нивните оценки по математика?

Задача 21. а) Пресметај го просечниот успех по математика кај учениците од датотеката креирана во задача 19.

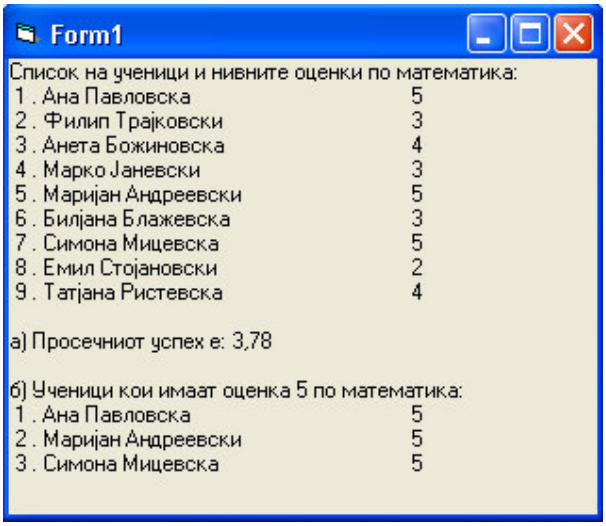
б) Излистај ги учениците од датотеката креирана во задача 19 кои имаат оценка 5 по математика.

Овие две барања ќе ги додадеме на кодот од ВБ-проектот за листање на сите податоци од датотеката, бидејќи така ќе се добие попрегледен излез. Просечниот успех (променливата uspeh) го пресметуваме така што откако прво ќе ги собереме сите оценки од учениците, добиениот збир (променливата suma) ќе го поделиме со бројот на учениците (променливата br). При излезот на вредноста на променливата uspeh за да се прикаже просечниот успех на две децимали ја користиме **функцијата Format** и со помош на форматот "##0.00" излезот ќе биде прикажан на две децимали. Додека пак, при листањето на учениците кои имаат оценка 5 по математика, променливата redbr служи за броење колку ученици има со оценка 5 по математика. Така го добиваме следниот дополнет и изменет код:

```
Private Type Ucenik
    ime As String * 30
    ocenka As Integer
End Type
Dim zapis As Ucenik
Dim br, i, suma, redbr As Integer
Dim uspeh As Double
Private Sub Form_Activate()
Print "Список на ученици и нивните оценки по математика:"
suma = 0
Open "c:\spisok.dat" For Random As #1 Len = Len(zapis)
    br = LOF(1) / Len(zapis)
    For i = 1 To br
        Get #1, i, zapis
        Print i; ". "; zapis.ime, zapis.ocenka
        suma = suma + zapis.ocenka
    Next i
Close #1
uspeh = suma / br
Print
Print "а) Просечниот успех е: "; Format(uspeh, "##0.00")
Print
Print "б) Ученици кои имаат оценка 5 по математика:"
redbr = 0
Open "c:\spisok.dat" For Random As #1 Len = Len(zapis)
    br = LOF(1) / Len(zapis)
    For i = 1 To br
        Get #1, i, zapis
        If zapis.ocenka = 5 Then
            redbr = redbr + 1
            Print redbr; ". "; zapis.ime, zapis.ocenka
        End If
    Next i
Close #1
End Sub
```

Задача 21.

По стартувањето на изменетиот и дополнет код на ВБ-проектот за листање треба да го добиете следниот излез:



Па, просечниот успех по математика на овие ученици и не е така лош, нели? Што мислите вие?

Во задача 16 од претходната лекција 7 го објаснивме алоритмот за подредување по големина на елементи од некоја низа реални броеви. Ќе го искористиме тој алгоритам за следната задача.

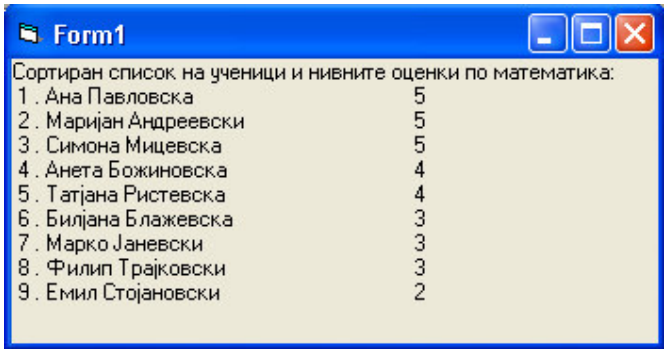
Задача 22. Сортирај го списокот на учениците од датотеката креирана во задача 19 во опаѓачки редослед според оценките (почнувајќи од учениците со највисока кон учениците со најниска оценка).

Почнете со нов ВБ-проект чии составни фајлови ќе ги снимите под имињата **sortiranje.frm** и **sortiranje.vbp**. Во еден од чекорите кај алгоритмот за подредување по големина (сортирање) има промена на места на два елемента во низа со помош на помошна променлива. Слично на тоа, овде во улога на помошни променливи ќе бидат zapis1 и zapis2 кои ќе ги превземат записите од редот i и редот maxindex соодветно, а потоа ќе си ги променат местата, односно вредноста на zapis2 ќе се запише во редот i, додека вредноста на zapis1 ќе се запише во редот maxindex. Важно е декларирањето на променливите од новиот тип на податоци Ucenik да биде поединечно. Тка, се добива следниот код на задачата:

```
Private Type Ucenik
    ime As String * 30
    ocenka As Integer
End Type
Dim zapis As Ucenik
Dim zapis1 As Ucenik
Dim zapis2 As Ucenik
Dim br, i, j, maxocenka, maxindex As Integer
Private Sub Form_Activate()
Print "Сортиран список на ученици и нивните оценки по математика:"
Open "c:\spisok.dat" For Random As #1 Len = Len(zapis)
    br = LOF(1) / Len(zapis)
    For i = 1 To br
        Get #1, i, zapis
        maxocenka = zapis.ocenka
        maxindex = i
        For j = i + 1 To br
            Get #1, j, zapis
            If zapis.ocenka > maxocenka Then
                maxocenka = zapis.ocenka
                maxindex = j
            End If
        Next j
        Get #1, i, zapis1
        Get #1, maxindex, zapis2
        Put #1, i, zapis2
        Put #1, maxindex, zapis1
        Get #1, i, zapis
        Print i; ". "; zapis.ime, zapis.ocenka
    Next i
Close #1
End Sub
```

Задача 22.

По стартувањето на програмата треба да го добиете следниот излез на сортиран список на учениците според нивните оценки по математика во опаѓачки редослед:



По извршеното сортирање на записите во датотеката spisok.dat, не само што излезот ќе биде сортиран, туку и внатре во самата датотека записите ќе останат сортирани, *датотеката spisok.dat ќе претрпи промени*.

Не беше лесно на волку малку страници да ви ја изложиме работата со датотеки за читање и запишување на податоци. Ви ги приложивме основните наредби и начини за работа со датотеките и искрено се надеваме дека можете да продолжите понатаму самите. Затоа ви ги предлагаме слениве задачи за самостојна работа:

Задача 23. Направи измени на податоците кај некој од учениците на списокот со ученици и нивните оценки по математика. (на пример, ученикот Филип Трајковски после напорна работа ја поправил оценката по математика и сега има оценка 5, измена која треба да се внесе во списокот)

Упатство: Со запишување на нов запис на местото (редниот број) на записот кој треба да претрпи измени.

Задача 24. Отстрани ги целосно податоците за некој од учениците на списокот со ученици и нивните оценки по математика. (на пример, ученичката Татјана Ристевска со оценка 4 по грешка била внесена на овој список, треба целиот запис кој ги содржи податоците за неа да се отстрани)

Упатство: Со истовремена работа со две отворени датотеки, од кои едната е spisok.dat, а другата е помошна датотека, на пример pomos.dat. Сите важечки записи се префрлаат од spisok.dat на pomos.dat. Се затвараат датотеките Со **наредбата Kill** се брише датотеката spisok.dat. Со **наредбата Name** се преименува pomos.dat во spisok.dat.

Ви посакуваме успех во работата.