Základy programování ve Visual Basic

Proměnná (variable)

Proměnnou, před jejím prvním použitím, musíme definovat. Definujeme jméno, typ a rozsah platnosti. Rozsah platnosti znamená, pro kterou část programu bude tato proměnná platit.

Základní typy proměnných

Visual Basic type	Common language runtime type structure	Nominal storage allocation	Value range
Boolean	Boolean	Depends on implementing platform	`True` or `False`
Byte	Byte	1 byte	0 through 255 (unsigned)
Char (single character)	Char	2 bytes	0 through 65535 (unsigned)
Date	DateTime	8 bytes	0:00:00 (midnight) on January 1, 0001 through 11:59:59 PM on December 31, 9999
Decimal	Decimal	16 bytes	0 through +/-79,228,162,514,264,337,593,543,950,335 (+/-7.9...E+28) † with no decimal point; 0 through +/-7.9228162514264337593543950335 with 28 places to the right of the decimal; smallest nonzero number is +/-0.0000000000000000000000000001 (+/-1E-28) †
Double (double-precision floating-point)	Double	8 bytes	-1.79769313486231570E+308 through -4.94065645841246544E-324 † for negative values; 4.94065645841246544E-324 through 1.79769313486231570E+308 † for positive values
Integer	Int32	4 bytes	-2,147,483,648 through 2,147,483,647 (signed)
Long (long integer)	Int64	8 bytes	-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807 (9.2...E+18 †) (signed)
Object	Object (class)	4 bytes on 32-bit platform 8 bytes on 64-bit platform	Any type can be stored in a variable of type `Object`

SByte	SByte	1 byte	-128 through 127 (signed)
Short (short integer)	Int16	2 bytes	-32,768 through 32,767 (signed)
Single (single-precision floating-point)	Single	4 bytes	-3.4028235E+38 through -1.401298E-45 † for negative values; 1.401298E-45 through 3.4028235E+38 † for positive values
String (variable-length)	String (class)	Depends on implementing platform	0 to approximately 2 billion Unicode characters
UInteger	UInt32	4 bytes	0 through 4,294,967,295 (unsigned)
ULong	UInt64	8 bytes	0 through 18,446,744,073,709,551,615 (1.8...E+19 †) (unsigned)
User-Defined (structure)	(inherits from ValueType)	Depends on implementing platform	Each member of the structure has a range determined by its data type and independent of the ranges of the other members
UShort	UInt16	2 bytes	0 through 65,535 (unsigned)

Způsob definování proměnných

- definice jména a typu proměnné
- rozsah platnosti
  - a) lokální např. Dim A, B As Integer) - platí pro proceduru, ve které je definována nebo pro procedury formuláře, pro který je definován
  - b) globální např. Public C As Long - platí pro celou aplikaci – definuje se zpravidla v Module

Definice pole:

- všechny prvky pole jsou stejného typu
- první prvek pole má index 0 (nula), definuje se maximální index
- např.: Dim A(10) As Integer

Změna velikosti pole – počtu prvků v poli:

- `Dim A(10) As Integer` - definice pole A, 10 je maximální index, první je 0 (nula)
- `ReDim A(20)` - Redefinice pole A – zvětšení počtu prvků, všechna existující data budou ztracena
- `ReDim Preserve A(20)` - Redefinice pole A – zvětšení počtu prvků, všechna existující data budou zachována – zůstanou v paměti

Při změně počtu prvků pole (ReDim) zůstane zachován typ proměnné!

Konverze typů proměnných

```
CBool(expression)       CInt(expression)        CStr(expression)
CByte(expression)       CLng(expression)        CUInt(expression)
CChar(expression)       CObj(expression)        CULng(expression)
CDate(expression)       CSByte(expression)      CUShort(expression)
CDbl(expression)        CShort(expression)
CDec(expression)        CSng(expression)
```

Použití:

```
Dim A As Integer
Dim B As String
B = "123"
A = Cint(B)              Konverze hodnoty proměnné B
A = Cint("123")          Konverze řetězce (string)
```

Pokud by konverze nebyla možná, dojde k fatální chybě. To vyřešíme pomocí příkazu Try – Catch:

```
Dim A As Integer
Dim B As String
B = "123"
Try
     A = CInt(B)
     A = CInt("123")
Catch ex As Exception
     MsgBox("Konverze nebyla možná!")
End Try
```

Matematické operandy a základní funkce

+, -	sčítání, odečítání
/	dělení
\	celočíselné dělení
MOD	zbytek po dělení
=	přiřazení
>, <, >=, <=, =, <> (není rovno)	porovnání

Matematické funkce

Pro aktivaci matematických funkcí musí být importován modul **System.Math**, import se definuje v záhlaví kódu formuláře, na kterém tyto funkce použijeme.

```
Imports System.Math
```

Abs	Vrátí absolutní hodnotu čísla.
Acos	Vrátí úhel, jehož kosinus je zadané číslo.
Asin	Vrátí úhel, jehož sinus je zadané číslo.
Atan	Vrátí úhel, jehož tangens je zadané číslo.
Atan2	Vrátí úhel, jehož tangens je podíl dvěma zadanými čísly.
BigMul	Vrátí úplný produkt dvě čísla 32-bit.

Ceiling	Vrátí nejmenší celočíselné hodnoty, která je větší než nebo rovno zadanému **Decimal** nebo **Double**.
Cos	Vrátí kosinus zadaného úhlu.
Cosh	Vrátí hyperbolický kosinus zadaného úhlu.
DivRem	Vrátí podíl dvou podepsané celá 32bitová nebo 64bitová verze a také v výstupní parametr vrátí zbytek.
Exp	Vrátí e (základ přirozeného logaritmu) umocněné na zadané.
Floor	Vrátí největší celé číslo, které je menší nebo rovno zadanému **Decimal** nebo **Double** číslo.
IEEERemainder	Vrátí zbytek, který je výsledkem dělení zadaného čísla jiným určené číslo.
Log	Vrátí přírodního (základní e) logaritmus zadaného čísla nebo logaritmus zadaného čísla v zadaném základu.
Log10	Vrátí dekadický logaritmus zadaného čísla.
Max	Vrátí větší ze dvou čísel.
Min	Vrátí menší z obou čísel.
Pow	Vrátí zadané číslo umocněné o zadaný exponent.
Round	Vrátí **Decimal** nebo **Double** hodnota zaokrouhlena na nejbližší celočíselné hodnoty nebo zadaný počet zlomkových číslic.
Sign	Vrátí **Integer** hodnotu označující znaménko čísla.
Sin	Vrátí sinus zadaného úhlu.
Sinh	Vrátí hyperbolický sinus zadaného úhlu.
Sqrt	Vrátí druhou odmocninu zadaného čísla.
Tan	Vrátí tangens zadaného úhlu.
Tanh	Vrátí hyperbolický tangens zadaného úhlu.
Truncate	Vrátí celou část čísla **Decimal** nebo **Double** číslo. - Truncate(-23,123) bude -23

Příklad: X = Math.Sqtr(Y)

Základní příkazy Visual Basic

MsgBox

- zpráva pro uživatele, příkaz vypíše na monitor řetězec, obsah proměnné nebo jejich různé kombinace

Syntaxe: Public Function MsgBox(ByVal Prompt As Object, Optional ByVal Buttons As_ MsgBoxStyle = MsgBoxStyle.OKOnly, Optional ByVal Title As Object = Nothing) As _ MsgBoxResult

Např.:
```
MsgBox("AHOJ!")
MsgBox("AHOJ " & Jméno & "!")
If MsgBox("Máš hlad " & Jméno & "?", vbYesNo) = vbYes Then MsgBox("Tak se najez!")
```

Rozhodování – větvení algoritmu

If

- příkaz rozvětvení algoritmu, větví algoritmus na základě pravdivosti podmínky

Syntaxe:
```
If condition [ Then ]
    [ statements ]
[ ElseIf elseifcondition [ Then ]
    [ elseifstatements ] ]
[ Else
    [ elsestatements ] ]
End If
```

Condition je podmínka, která má pravdivostní hodnotu True (je pravda) nebo False (není pravda).
Then – za příkazem Then budou provedeny všechny příkazy (blok Statemens) pouze pokud je pravdivostní hodnota podmínky rovna True.
ElseIf – vykoná se pouze když je pravdivostní hodnota podmínky Condition False, za příkazem ElseIf následuje nová podmínka (elseifcondition), platí pro ni stejná pravidla jako pro podmínku Condition

Else – v případě, že je False podmínka If a také podmínka ElseIf, vykonají se všechny příkazy v bloku elsestatemens.
End if – zakončení příkazu If – je nutné

Např.:

```
If A < 1 Then
    B = 11
ElseIf A < 12 Then
    B = 22
Else
    B = 33
End If
```

Hodnota proměnné A	Hodnota proměnné B
A< 1	11
1<=A<12	22
A>=12	33

Select Case

- příkaz pro vícenásobné větvení algoritmu

syntaxe:
```
Select [Case] testexpression
    [Case expressionlist
        [statements] ]
    [Case Else
        [elsestatements] ]
End Select
```

textexpresion – proměnná, jejíž obsah rozhodne o další cestě
case expresion list – vyjmenované konkrétní hodnoty proměnné, v tokmto případě budou vykonány příkazy **statements**
case else – při osttaních hodnotách prměnné budou vykonány příkazy **elsestatements**

např.:
```
Select Case A
    Case 1, 2, 3
          B = 0
    Case 4, 5
          B = 1
    Case Else
          B = 3
End Select
```

Cykly

For

- příklad cyklu

syntaxe:

```
For counter [ As datatype ] = start To end [ Step step ]
    [ statements ]
    [ Exit For ]
    [ statements ]
Next [ counter ]
```

Counter – řídící proměnná cyklu, musí být celočíselného typu
Start – počáteční hodnota řídící proměnné
End – konečná hodnota řídící proměnné
Step - krok cyklu, celočíselná hodnota, může být i záporná (setupný cyklus)
Exit For – příkaz výstupu z cyklu
Next - konec cyklu

např.:
```
D = 3
For A = 2 To 9 Step 2
        D = D + 1
        If D = 10 Then Exit For
Next A
```
Pozn. :V tomto cyklu nikdy nedojde k vykonání příkazu Exit For, neboť proměnná D nikdy nebude mít hodnotu 10!

```
For A = 1 To 9
        D(0) = D(0) + D(A)
Next A
```
Pozn. :Do buňky s indexem nula cyklus postupně vloží hodnotu, která se rovná součtu hodnot buněk s indexem 1 až 9.

Do Loop

- příkaz cyklu

syntaxe:
```
Do {While|Until} condition
    [statements]
    [Exit Do]
    [statements]
Loop
```
nebo
```
Do
    [statements]
    [Exit Do]
    [statements]
Loop {While|Until} condition
```

While|Until - je podmínka, která řídí cyklus. **While** se opakuje tak dlouho, dokud je hodnota podmínky rovna **True, Until** dokud je hodnota podmínky rovna **False.**

Např.:
```
Dim check As Boolean = True
Dim counter As Integer = 0
Do
    Do While counter < 20
        counter = conter + 1
        If counter = 10 Then
            check = False
            Exit Do
        End If
    Loop
Loop Until check = False
```

Tento příklad má dva cykly, jeden vnořený (vnitřní) do druhého. Vnější cyklus se opakuje dokud je hodnota proměnné check True (tedy dokud je podmínka Check=false nepravda (false)). Vnitřní cyklus se opakuje dokud hodnota podmínka rovna true, tedy dokud je hodnota proměnné counter menší než 20. Pozor, cyklus ale končí když je hodnota proměnné counter rovna 10 neboť se aktivuje příkaz Exit Do!

Příklady cyklů

1)
```
D = 10
For A = -5 To 5
    D = D + A
Next
```

Na konci cyklu bude hodnota A rovna 6 a hodnota D rovna 10.

2)
```
D = 10
For A = 5 To -5 Step -3
    D = D + A
Next
```

Na konci cyklu bude hodnota A rovna -7 a hodnota D rovna 12.

3)
```
D = 10
For A = -5 To 5 Step -3
    D = D + A
Next
```

Na konci cyklu bude hodnota A rovna -5 a hodnota D rovna 10.

4)
```
D = 10
A = 1
Do While (A < 10)
    A = A + 1
    D = D + A
Loop
```

Na konci cyklu bude hodnota A rovna 10 a hodnota D rovna 64.

5)
```
D = 10
A = 1
Do While (A < 10)
    D = D + A
    A = A + 1
Loop
```

Na konci cyklu bude hodnota A rovna 10 a hodnota D rovna 55.

6)
```
D = 10
A = 1
Do
    D = D + A
    A = A + 1
Loop While (A < 10)
```

Na konci cyklu bude hodnota A rovna 10 a hodnota D rovna 55.

7)
```
D = 10
A = 1
Do Until (A < 10)
    D = D + A
    A = A + 1
Loop
```

Na konci cyklu bude hodnota A rovna 1 a hodnota D rovna 10.

8)
```
D = 10
A = 1
Do
    D = D + A
    A = A + 1
Loop Until (A < 10)
```

Na konci cyklu bude hodnota A rovna 2 a hodnota D rovna 11.

Operace s textovými řetězci (stringy)

Trim, Ltrim, Rtrim

– příkazy, které ořezávají (odstraňují) prázdné znaky (space – mezerník) ze začátku a konce proměnné typu string. Trim z obou konců, Ltrim zleva (ze začátku) a Rtrim zprava (konce) řetězce.

Slovo=“ Ahoj ! “
Slovo=LTrim(Slovo)
Slovo bude “Ahoj ! “

Slovo=“ Ahoj ! “
Slovo=RTrim(Slovo)
Slovo bude “ Ahoj !“

Slovo=“ Ahoj ! “
Slovo=Trim(Slovo)
Slovo bude “Ahoj !“

Len

- vrátí počet znaků (symbolů) řetězce

```
Dim A As String
Dim B As Integer
A = "as SAD ASHDKS"
B = Len(A)
```

Hodnota proměnné B bude 13.

Mid

- vrátí část řetězce

syntaxe: `Mid (string, start, length)`

String – proměnná typu string
Start – pozice v proměnné string od které operace bude začínat
Length – počet znaků, které budou vráceny

Např.: Slovo=“Ahoj pracovitý studente!“
Slovo=Mid (slovo,6,3)
Slovo bude = “pra“

InStr

– vrátí číslo, které definuje pozici hledaného výrazu (řetězce – stringu) v prohledávaném výrazu (stringu), pokud výraz není nalezen, vrátí hodnotu 0 (nula)

syntaxe: `Instr(Pozice, String1, String2, CompareMethod )`

Pozice – pozice, od které se bude ve stringu String1 hledat (celé číslo)
String1 – řetězec (string), ve kterém se bude hledat
String2 - řetězec (string), který se bude hledat ve stringu String1
CompareMethod – Text – platí, že "A" = "a", Binary – platí, že "A" <> “a“

Např. Slovo1 = "Dobrý den, studenti! Jak se máte?"
Slovo2 = "Den"
Nalezeno1 = Instr(1, Slovo1, Slovo2, vbTextCompare)
Nalezeno1 bude rovno 7.
Nalezeno2 = Instr(10, Slovo1, Slovo2, vbTextCompare)
Nalezeno2 bude rovno 0 (protože od pozice 10 Slovo2 v Slovo1 není!
Nalezeno3 = Instr(3, Slovo1, Slovo2, vbTextCompare)
Nalezeno3 bude rovno 7 (start hledání nemění pozici nalezení).
Nalezeno4 = Instr(1, Slovo1, Slovo2, vbBinaryCompare)
Nalezeno4 bude rovno 0, protože „den“ <> “Den“!

Zachycení chyb

Try

- příkaz, který umožní odchycení chyb, program tak neskončí fatální chybou.

Syntaxe:

```
Try
      [statements]
Catch ex As Exception
      [statements]
End Try
```

Příklad použití:

```
 Try
      A = CInt(B)
      A = CInt("123")
Catch ex As Exception
      MsgBox("Konverze nebyla možná!")
End Try
```

Práce s datem a časem:

Now

Vrací aktuální datum a čas

```
Dim ThisMoment As Date
 ThisMoment = Now
```

Today

Vrací aktuální datum (čas je 0:00:00)

```
Dim thisDate As Date
thisDate = Today
```

TimeOfDay

Vrací aktuální čas dne

```
Dim currentTime As Date
currentTime = TimeOfDay
```

DateAdd

DateAdd(DateInterval, Double, DateTime)
Přidá k časovému údaji DateTime počet Double časových intervalů DateInterval

Možnosti parametrů DateInterval

DateInterval.Day	DateInterval.Quarter
DateInterval.DayOfYear	DateInterval.Second
DateInterval.Hour	DateInterval.Weekday
DateInterval.Minute	DateInterval.WeekOfYear
DateInterval.Month	DateInterval.Year

Příklady:

```
Dim NextTime As Date = Now
NextTime = NextTime.AddDays(3.4) ' Increment by 3 2/5 days.
```

Jestliže Now bude 02.02.2018 14:46:01 potom Next time bude 06.02.2018 0:22:01.

```
Dim NextMonth As Date = DateAdd(DateInterval.Month, 1, #1/31/1995#)
```

Tento příkaz vrátí hodnotu #2/28/1995# ne hodnotu #2/31/1995#.
Pro datum #1/31/1996# vrátí #2/29/1996#, protože rok 1996 byl přestupný.

DateDiff

- vypočítá rozdíl mezi dvěma daty v definovaných jednotkách
- syntaxe:

```
DateDiff(DateInterval, datTim1, datTim2)
```

Příklady:

```
Dim datTim1 As Date = #1/4/2001#
Dim datTim2 As Date = #1/9/2001#
Dim D As Long = DateDiff(DateInterval.Day, datTim1, datTim2)
```

Hodnota proměnné D bude 5.

```
Dim datTim1 As Date = #1/2/2001#
Dim datTim2 As Date = #1/31/2001#
Dim D As Long = DateDiff(DateInterval.Weekday, datTim1, datTim2)
```

Hodnota proměnné D bude 4 (protože stejný den v týdnu bude v intervalu celkem 4 krát).

```
Dim datTim1 As Date = #1/2/2001#
Dim datTim2 As Date = #1/19/2001#
Dim D As Long = DateDiff(DateInterval.WeekOfYear, datTim1, datTim2)
```

Hodnota proměnné D bude 2 (protože mezi daty jsou dva týdny).

RunLength

- měření času procesu

Příklad:
Změření času v milisekundách mezi časem startTime a TimeSpan

```
Dim startTime As Date = Now
'proces – příkazy algoritmu
Dim runLength As Global.System.TimeSpan = Now.Subtract(startTime)
Dim millisecs As Double = runLength.TotalMilliseconds
```

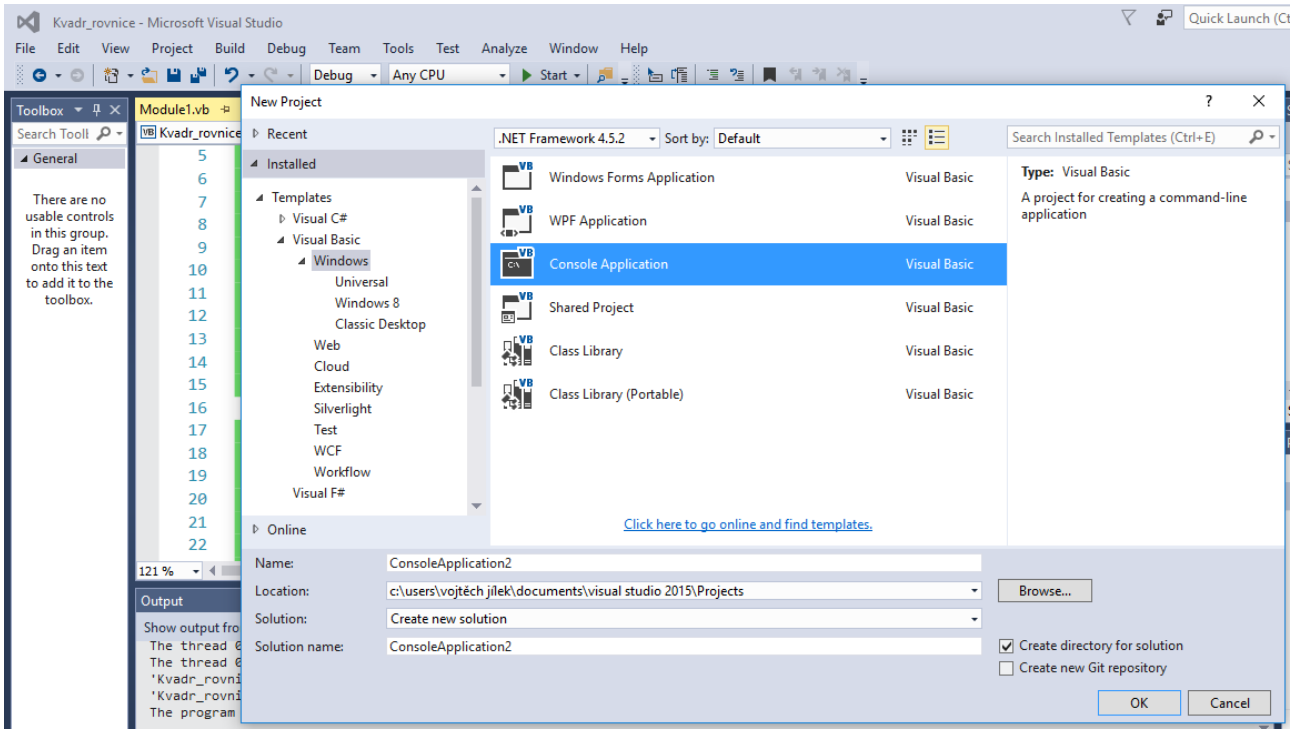
Threading.Thread.Sleep

```
Threading.Thread.Sleep(500)
```

- příkaz, který přesuší běh programu na daný počet milisekund
- po uplynutí definovaného času (počtu milisekund) pokračuje algoritmus následujícím řádkem – příkazem
- hodnoty proměnných nejsou nijak ovlivněny - změněny

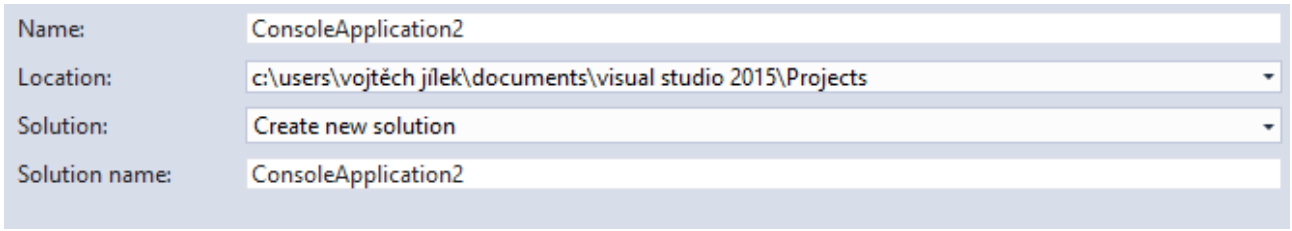
Vytvoření projektu Console (konzole)

Při vytvoření nového projektu zvolíme Console Aplication:



Pozor, projekt se vytváří do objektu module!

… a dále vyplníme potřebné názvy projektu:



Příklad programu, který bude řešit rovnici:

$a.x^2+b.x+c=0$

Syntaxe ve Visual Basic 2017:

```
Module Module1

    Function nacti(ByVal param As String) As Double
        Dim a As Double
        Console.WriteLine("Zadejte " & param & "!")
        Try
            a = Convert.ToInt32(Console.ReadLine())
        Catch ex As Exception
            Console.WriteLine("Nespravny vstup!")
            Threading.Thread.Sleep(500)
            Environment.Exit(0)
        End Try
        Return a
    End Function

    Sub Main()
        Dim a, b, c, x1, x2, D As Double
        Console.WriteLine(Date.Now.DayOfWeek)
        Console.WriteLine("Reseni rovnice ve tvaru ax^2 + bx + c = 0")
        a = nacti("a")
        b = nacti("b")
        c = nacti("c")
        D = b * b - 4 * a * c
        If a = 0 Then
            If b = 0 Then
                If c = 0 Then
                    Console.WriteLine("Rovnice ma nekonecno reseni!")
                Else
                    Console.WriteLine("Rovnice nema reseni!")
                End If
            Else
                x1 = -c / b
                Console.WriteLine("Rovnice ma jedno reseni a to: x = " & x1)
            End If
        ElseIf (D < 0) Then
            Console.WriteLine("Rovnice nema reseni!")
        ElseIf (D = 0) Then
            x1 = -b / (2 * a)
            Console.WriteLine("Rovnice ma jedno reseni a to: x = " & x1)
        Else
            x1 = (-b + Math.Sqrt(D)) / (2 * a)
            x2 = (-b - Math.Sqrt(D)) / (2 * a)
            Console.WriteLine("Rovnice ma dve reseni a to: x = " & x1 & ", a x = " & x2)
        End If
        Console.ReadKey()
    End Sub

End Module
```

Stejný příklad napsaný v programovacím jazyce C:

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

double nacti(char param)
{
	double a;
	printf("Zadejte %c!\n", param);
	if(scanf("%lf",&a)!=1)return 1/0.0;
	else return a;
}

int main(void)
{
	double a,b,c,D,x1,x2;
	printf("Reseni rovnice ve tvaru ax^2 + bx + c = 0\n");
	a = nacti('a');
	b = nacti('b');
	c = nacti('c');
	if(a==1/0.0 || b==1/0.0 || c==1/0.0){printf("Nespravny vstup!\n"); return 1;}
	D = b * b - 4 * a * c;
	if(a==0)
	{
		if(b==0)
		{
			if(c==0)printf("Rovnice ma nekonecno reseni!\n");
			else printf("Rovnice nema reseni!\n");
		}
		else
		{
			x1 = -c / b;
			printf("Rovnice ma jedno reseni a to: x = %f\n",x1);
		}
	}
	else if(D < 0)printf("Rovnice nema reseni!\n");
	else if(D==0)
	{
		x1 = -b / (2 * a);
		printf("Rovnice ma jedno reseni a to: x = %f\n",x1);
	}
	else
	{
		x1 = (-b + sqrt(D)) / (2 * a);
		x2 = (-b - sqrt(D)) / (2 * a);
		printf("Rovnice ma dve reseni a to: x = %f, a x = %f\n",x1,x2);
	}

	return 0;
}
```

Funkce ve VB

Funkce je speciální typ podprogramu, má vždy návratovou hodnotu, kterou přiřadí definované proměnné v hlavním programu. Může mít vstupní a výstupní parametry. Příklad použití:

```
Public Function Soucet(a As Integer, b As Integer) As Integer
    Dim a1 As Integer
    a1 = a + 2
    Return (a1 + b)
End Function
```

Vstupními parametry této funkce jsou hodnoty proměnných „a“ a „b“, které jsou zadány (v našem příkladu) v hlavním programu formuláře Form1 a mají hodnotu 1 a 2. Návratová hodnota je typu Integer (As Integer v definici funkce) je pro ni uloženo místo ale není to proměnná. Tuto hodnotu definujeme za klíčovým slovem (příkazem) Return. Tímto způsobem vracíme pouze jednu hodnotu, lze vrátit i pole.

```
Public Function Soucet1(a As Integer, b As Integer, ByRef x As Integer) As Integer
    Dim a1 As Integer
    a1 = a + 2
    x = 3
    Return (a1 + b)
End Function
```

Tato druhá funkce má navíc výstupní parametr x, který je vrácen, spolu s návratovou hodnotou (za Return) do místa volání procedury (hlavního programu formuláře Form1). Tímto způsobem vracíme více hodnot

```
'toto je hlavní program formuláře Form1
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    MsgBox(Soucet(1, 2).ToString())
    Dim x As Integer = 1
    Dim z As Integer = Soucet1(1, 2, x)
    MsgBox(z & ", " & x)
End Sub
```

Rekurze - Rekurze je zvláštní aktivace (spuštění, volání) funkce nebo procedury, kdy funkce nebo procedura aktivuje (spouští, volá) sebe sama.

```
Function faktorial(cislo As Integer) As Integer
    If cislo = 0 Then Return 1
    Return cislo * faktorial(cislo - 1)
End Function

Sub Main()
    Dim cislo As Integer
    Console.WriteLine("Zadej cislo!")
    Try
        cislo = Convert.ToInt32(Console.ReadLine())
    Catch
        Console.WriteLine("Nespravny vstup!")
    End Try
    Console.Write(faktorial(cislo))
    Console.WriteLine()
    Console.ReadKey()
End Sub
```