## 3. Розробка та програмування алгоритмів розгалуженої структури

**Мета та основні завдання роботи**: дослідити особливості створення елементарних програмних засобів розгалуженої структури у мові програмування Visual Basic.

### 3.1 Основні теоретичні відомості

Більшість задач, що зустрічаються в практичній діяльності людини, не вкладаються в лінійну схему, а мають варіанти розрахунків, які залежать від виконання (або не виконання) тих чи інших умов. Наприклад, при розв'язанні квадратного рівняння спосіб обчислення коренів залежить від знаку дискримінанта. Або в наступному прикладі:

$$y = \begin{cases} x\text{^}2 - 4 * x\text{^}0.5, при \quad x > 0 \\ x\text{^}3 + 1.8 * x, при \quad x <= 0 \end{cases}$$

в залежності від знаку змінної *x*, змінна *y* обчислюється за різними формулами.

***Блочний оператор* IF.** При необхідності виконання великої групи операторів в залежності від якої-небудь умови найбільш зручно використовувати блочний умовний оператор **If**. За допомогою блочного оператора **If** може бути перевірено не одну умову, а будь яку кількість умов, які є альтернативами та перевіряються послідовно до «першого виконання». Для цього в операторі **If** використовуються оператори **ElseIf** (для перевірки другого і подальших умов). Виконання будь якої з альтернативних гілок приводить к виходу з блоку. Тому в загальному випадку оператор **If** має такий формат:

```
If < умова 1> Then
        1-а група операторів
[ ElseIf < умова 2> Then
        2-а група операторів ]
...
[ Else
        n-а група операторів ]
End If
```

Слід пам'ятати, що кожен блочний оператор **If** має бути «закритий» оператором **End If**. Частини **Else** і **ElseIf** є необов'язковими[19]. У скороченому варіанті оператор має такий формат:

```
If <умова> Then
        перша група операторів
Else
        друга група операторів
End If
```

[19] Оператори або їх фрагменти, «взяті в квадратні дужки», не є обов'язковими.

або такий:

```
If <умова> Then
    група операторів
End If
```

де *перша група операторів* (*група операторів*), та *друга група операторів* – оператори, які виконуються у разі дотримання або не дотримання умови відповідно. В «***умові***» записуються вирази (або змінні), пов'язані знаками відношень: **=, <, >, <=, >=, <>.** Вони називаються логічними операціями порівняння.

Умова, в операторові **If**, також може бути складною. Складні умови будуються з простих відношень за допомогою знаків логічних операцій:

- **And** – логічне множення (*кон'юнкція*) "**І**". В цій операції результат є істиною, коли кожна з простих умов істинна;
- **Or** – логічне складання (*диз'юнкція*) "**АБО**". В цій операції результат є істиною, коли хоч одна з складових умов істинна;
- **Not** – логічне заперечення (*інверсія*) "**НІ**". В цій операції хибне значення перетворюється на істинне і навпаки.

Приклад 1: `If x > 5 And x <= 50.5 Then …`

Приклад 2: `If x <= 5 Or x > 50.5 Then …`

Порядок виконання операцій при перевірці умов (при обчисленні логічних виразів) наступний:

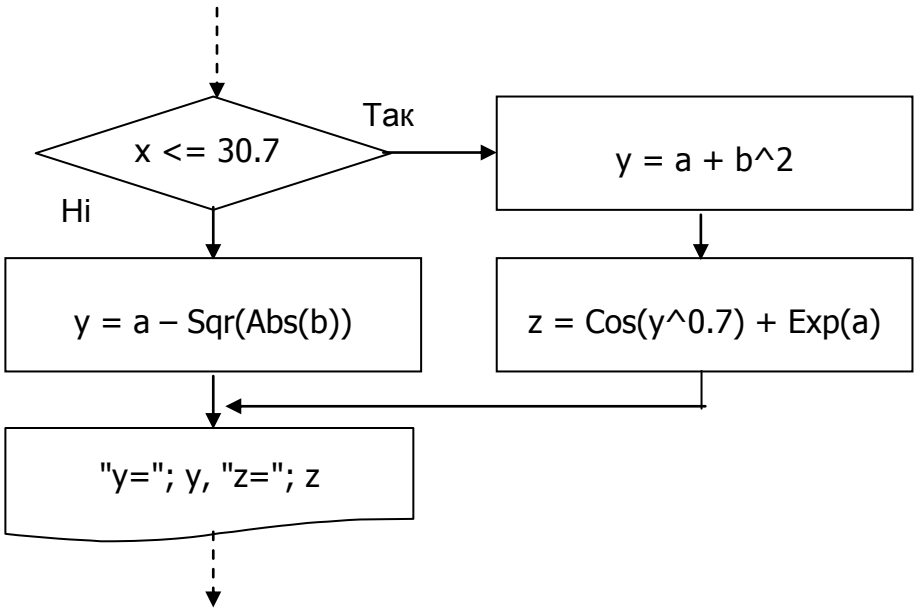
1. Обчислюються усі арифметичні вирази;
2. Виконуються усі операції відношень;
3. Виконуються усі операції **Not**;
4. Виконуються усі операції **And**;
5. Виконуються усі операції **Or**

*Робота оператора*: Якщо умова істинна, то виконується "*перша група операторів*", розташована в рядках після оператора **If**, а якщо хибна, – то виконується "*друга група операторів*", розташована в рядках після оператора **Else.** Після виконання однієї з гілок – «**Then**» або «**Else**» дія програми триває з оператора що йде за оператором **End If**, тобто завжди виконується тільки одна з гілок алгоритму розгалуження. Оператор **If** можна використовувати в скороченій формі (якщо частина «**Else**» не потрібна – приклад 4).

Приклад 3:

```
If x <= 30.7 Then
    y = a + b^2
    z = Cos(y^0.7) + Exp(a)
    MsgBox "y=" & y & "z=" & z
Else
    y = a – Sqr(Abs(b))
    MsgBox "y=" & y
End If
```

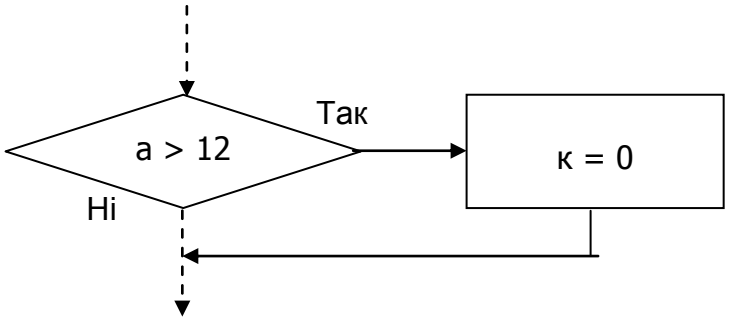
Цей оператор працює за таким алгоритмом:




Приклад 4:

```
If a > 12 Then
к = 0          ‘Якщо a>12, коефіцієнт к обнуляється
End If
```

працює за алгоритмом:




Приклад 5.

Визначити значення функції ***Y*** при різних значеннях аргументу ***x***, за наступних умов:

$$Y=\begin{cases} EXP(x), & \text{при } x = 1 \\ COS(x^2)^2, & \text{при } 2 <= x =< 5 \\ TG(SQR(x)), & \text{при } 1 < x < 2 \\ a+b^2, & \text{при } x < 1,\ x > 5 \end{cases}$$

Варіант алгоритму для розв’язання цього прикладу наведений на рис. 3.1

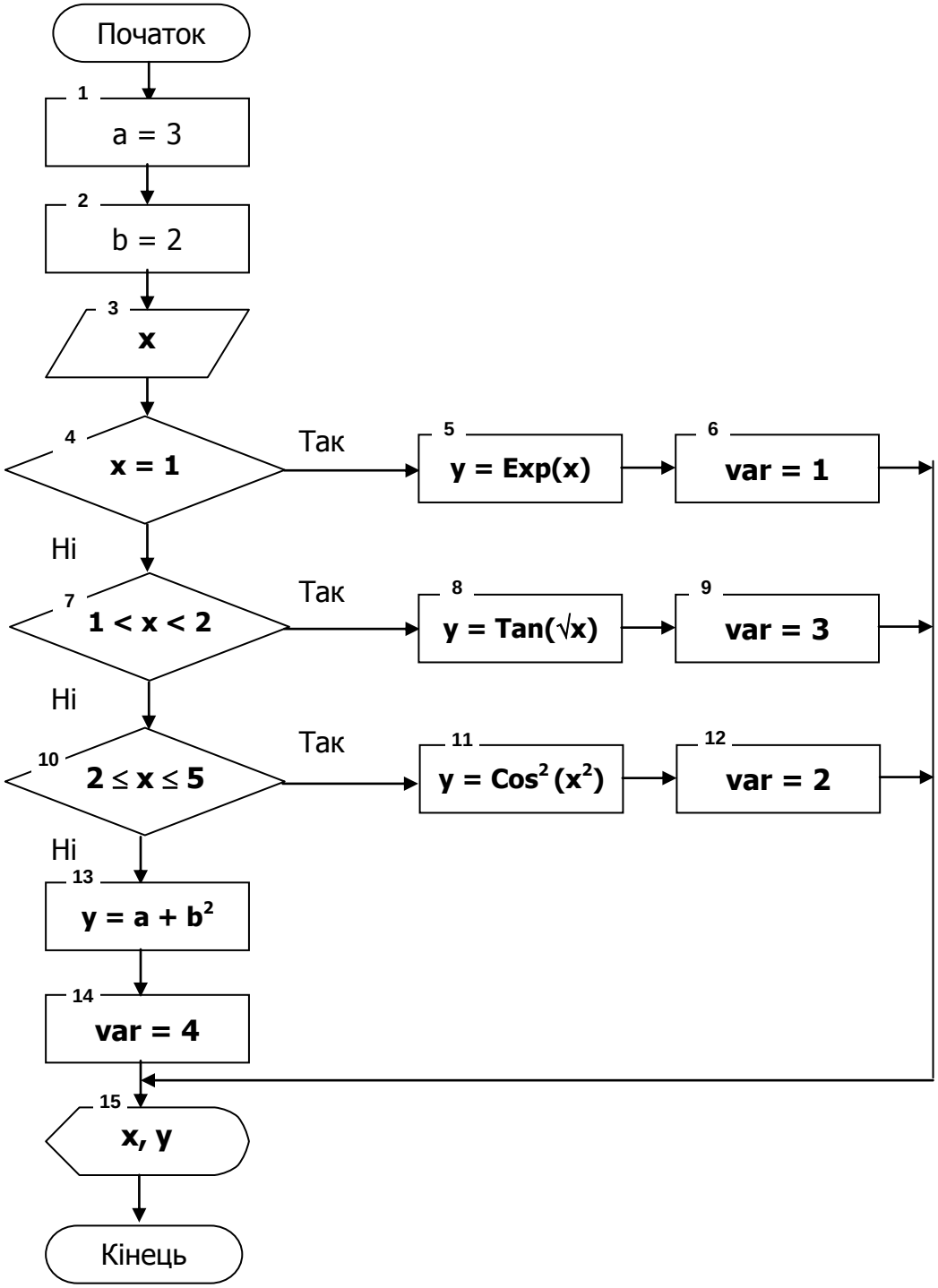



Рис. 3.1. Алгоритм розв’язання прикладу 5

Приклад розв'язання:

```
Option Explicit

Public Sub Розгалуження1()
    Dim x As Single, y As Single, a As Single, b As Single, var As Integer
    a = 3: b = 2
    x = InputBox("Введіть x")
    If x = 1 Then
        y = Exp(x)
        var = 1
    ElseIf x > 1 And x < 2 Then
        y = Tan(Sqr(x))
        var = 3
    ElseIf x >= 2 And x <= 5 Then
        y = Cos(x ^ 2) ^ 2
        var = 2
    Else
        y = a + b ^ 2
        var = 4
    End If
    MsgBox "x=" & x & "  y=" & y & vbNewLine & "варіант" & var
End Sub…
```

Одна умова може бути вкладена усередині іншої, тобто перевірятися тільки, якщо виконалася перша (зовнішня) умова. Наприклад:

Приклад 6.

```
If x > 0 Then
        If y > 0 Then                   ` перевіряється тільки, якщо x > 0
            Debug.Print " x > 0  та  y > 0 "
        Else
            Debug.Print " тільки x > 0 "
        End If
ElseIf x = 0 Then
        If y = 0 Then                   ` перевіряється тільки, якщо y = 0
            Debug.Print " x = y = 0 "
        Else
            Debug.Print " тільки  x = 0 "
        End If
Else
        Debug.Print " x < 0 "
End If
```

Для наочності програми (тобто візуального визначення, до якого **If** відносяться частини **Else** та **End If**, а також чітко бачити всі гілки розгалуження) необхідно використовувати елементи структурного програмування, виділяючи кожний блок (в даному випадку – **If-Else-End If**), який є в тексті програми. Для цього, використовуючи певну кількість проміжків, слід зміщувати залежні рядки оператора **If** (оператори кожної з гілок) праворуч. Див. приклад 6.

***Рядковий оператор* IF** є історично більш старим інструментом і має менше можливостей у порівнянні з блочним оператором **If**. Його доцільно використовувати в разі перевірки нескладних (не громіздких) окремих умов. Він записується в один рядок і має такий формат:

If <умова> Then <оператор 1> [Else <оператор 2>]

де *оператор* 1, *оператор* 2 – оператори мови Visual Basic, які виконуються у випадку виконання або не виконання умови. "*Умова*" записується за тими самими правилами, що і для блочного оператора **If.**

*Робота оператора*: Якщо умова істинна, то виконується "оператор 1", записаний після слова **Then**, а якщо хибна, – то виконується "оператор 2", записаний після слова **Else**. Після виконання однієї з гілок (**Then** або **Else**) виконання програми продовжується з оператора наступного за оператором **If**. Оператор **If** можна використовувати у скороченій формі (частина "**Else**" відсутня – приклади 2 і 3). Приклади:

1) **If** a > 0 **Then** y = Sin(a-2) **Else** y = 0.5 * a - 1
2) **If** b > 0 **Then** y = Cos(b)
3) **If** z > 0 **Then** k%=1

В двох останніх випадках якщо умова хибна, то фактично нічого не відбувається – виконання програми продовжується з оператора програми, який слідує за оператором **If** оскільки відсутня частина **Else**.

Нижче наведений варіант програми з використанням рядкового оператору **If** для розв'язання прикладу №5.

```
' ПРОГРАМА РОЗРАХУНКУ y
a = InputBox("Введіть значення змінних a")
b = InputBox("Введіть значення змінних b")
x = InputBox("Введіть значення змінних x")
If x = 1 Then y = Exp(x)
If x > 1 And x < 2 Then y = Tan(Sqr(x))
If x >= 2 And x <= 5 Then y = Cos(x ^ 2) ^ 2
If x < 1 Or x > 5 Then y = a + b ^ 2
Debug.Print " При x="; x, "y="; y
```

# 4. Програмування розгалужень за допомогою оператора багатоваріантного вибору

**Мета та основні завдання роботи**: дослідити особливості створення елементарних програмних засобів розгалуженої структури у мові програмування Visual Basic.

## 4.1 Основні теоретичні відомості

***Оператор багатоваріантного вибору*** **SELECT CASE** подібно до блочного оператора **IF** може реалізовувати алгоритми розгалуженої структури. Але його відмінність полягає в тому, що всі варіанти розгалуження в ньому залежать від значення тільки одного параметру, який визначається в головному операторі **Select Case**.

```
Select Case <параметр>
    Case K1
        <1-а група операторів>
    Case K2
        <2-а група операторів>
        ...
    Case Else
        <n-а група операторів>
End Select
```

Варіанти форматів оператору **Case**:

1. Аргументи в списку оператора **Case** задаються переліком значень (розділяються комами)

   Приклади (перевіряється чи дорівнює значення параметру заданим числам):

   ```
   Case 1,3,5
   Case 2,4,
   Case 0
   ```

2. Аргументи в списку оператора **Case** задаються діапазоном значень:

   **Case <вираз 1> TO < вираз 2>**

(значення < *вираз 2*> повинно бути більше значення < *вираз 1*>).

Приклади (перевіряється чи знаходиться значення параметру у вказаному діапазоні):

```
Case 1 To 9
Case 2.5 To 273.2
Case -7 To -3
```

3. Аргументи в списку оператора **Case** задаються за допомогою умови:

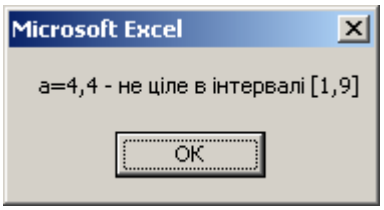
   **Case Is < знак відношення> <вираз>**

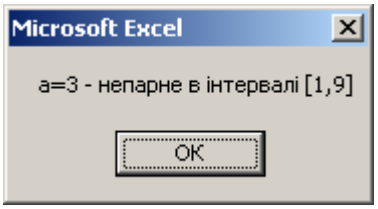
Приклади (перевіряється чи влаштовує значення параметру заданій умові):

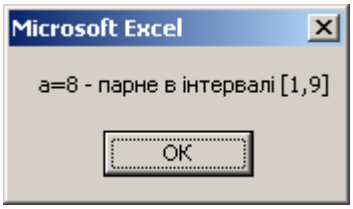
**Case is < 1**
**Case is > 9**
**Case Is < 2.1 + b ^ 2**

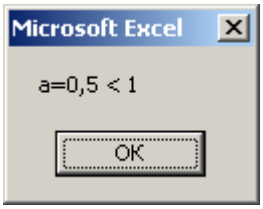
Нижче наведений приклад програми із використанням оператору **SELECT CASE:**

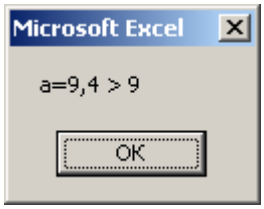
```
Public Sub Vybir1()
' Оператор багатоваріантного вибору. Приклад 1.
   Dim a As Single
   a = Application.InputBox("Задайте число", , , , , , , 1)
   Select Case a
      Case 1, 3, 5, 7, 9
         MsgBox "a=" & a & " - непарне в інтервалі [1,9]"
      Case 2, 4, 6, 8
         MsgBox "a=" & a & " - парне в інтервалі [1,9]"
      Case Is < 1
         MsgBox "a=" & a & " < 1"
      Case Is > 9
         MsgBox "a=" & a & " > 9"
      Case Else
         MsgBox "a=" & a & " - не ціле в інтервалі [1,9]"
   End Select
End Sub
```



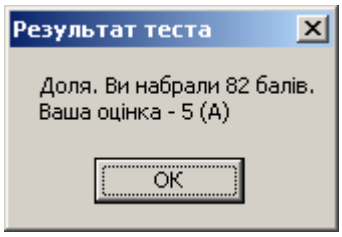



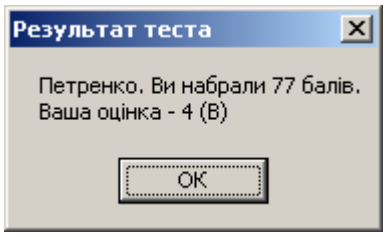

Далі наведений варіант програми із використанням оператора **SELECT CASE** для розв'язання прикладу №5.

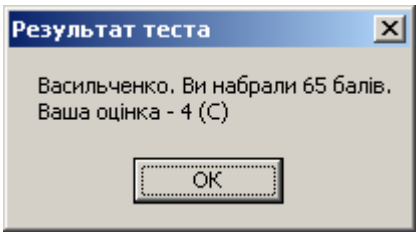
```
Sub TestRezalt5()
' Оператор багатоваріантного вибору. Приклад 2.
   Dim sName As String, sMessage As String, iMark As Integer
   sName = InputBox("Ваше прізвище?")
   iMark = Application.InputBox("Скільки балів Ви набрали?", , , , , , , 1)
            sMessage = sName & ". Ви набрали " & iMark & " балів." & vbNewLine &
                                                          "Ваша оцінка - "
   Select Case iMark  'Блок вибору оцінки залежно від кількості набраних балів
      Case Is >= 82
         sMessage = sMessage & "5 (A)"
      Case 73 To 81
```

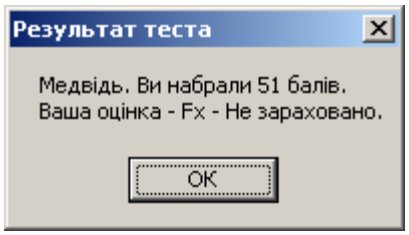
```
        sMessage = sMessage & "4 (B)"
    Case 65 To 72
        sMessage = sMessage & "4 (C)"
    Case 56 To 64
        sMessage = sMessage & "3 (D)"
    Case 52 To 55
        sMessage = sMessage & "3 (E)"
    Case Is < 52
        sMessage = sMessage & "Fx - Не зараховано."
  End Select
  MsgBox sMessage, , "Результат теста"
End Sub
```

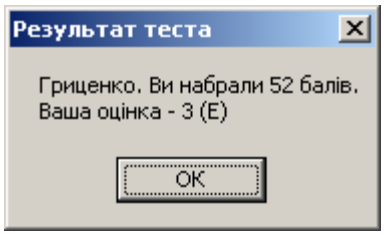
Приклади результатів, отриманих за програмою **Sub TestRezalt5()**

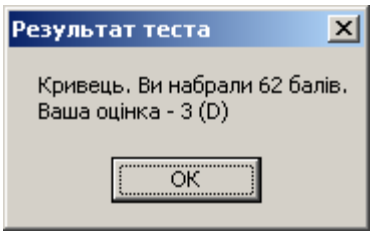



**Sub Vybir2()**

```
' Оператор багатоваріантного вибору. Приклад 3.
   Dim x As Single, y As Single, a As Single, b As Single, var As Integer
   a = 3: b = 2
   x = InputBox("Введіть x")
   Select Case x
     Case 1
        y = Exp(x)
        var = 1
     Case Is < 1
        y = a + b ^ 2
        var = 4
     Case Is < 2
        y = Tan(Sqr(x))
        var = 3
     Case Is <= 5
        y = Cos(x ^ 2) ^ 2
        var = 2
```

```
      Case Else
         y = a + b ^ 2
         var = 4
   End Select
   MsgBox "x=" & x & "  y=" & y & vbNewLine & "варіант" & var
End Sub
```

## 5. Організація арифметичних циклів

**Мета та основні завдання роботи**: дослідити особливості створення елементарних програмних засобів циклічної структури у мові програмування Visual Basic.

### 5.1 Основні теоретичні відомості

Циклічним називають алгоритм, в якому певна група операцій повторюється декілька разів. ***Арифметичні цикли*** *(цикли з лічильником)* – це цикли, в яких кількість повторень циклу відома (може бути розрахована) до його виконання. Для програмування таких циклів використовується оператор **FOR…NEXT.** Синтаксис цього оператора має вигляд:

```
FOR <змінна>=<початок> TO <кінець> [ STEP <крок> ]
    ...
    ...      [ EXIT FOR ]      } Тіло циклу
    ...
NEXT <змінна>
```

де *<змінна>* – це змінна – *параметр циклу*, *<початок>* та *<кінець>* – відповідно початкове та кінцеве значення цієї змінної, а *<крок>* – крок її змінення. Якщо крок не задано, він за умовчанням приймає значення $+1$.

Виконання циклу можна перервати достроково за допомогою оператора **Exit For**. В цьому випадку програма продовжує свою роботу з оператора, що слідує за оператором **Next.**

Приклади заголовків циклів:

```
For i = 1 To 28                    For x = xn To xk Step dx
For i = m To n                     For y = 0.15 To 4.5 Step 0.25
For k = 23 To -44 Step -3          For z = 1.9 To 0.3 Step -0.05
```

Алгоритм, за яким працює оператор **For…Next,** наведено на рис. 4.1[29]. Тобто, слід пам'ятати, що в операторі циклу **For…Next** умова завершення циклу перевіряється на його початку (цикл з передумовою). Тому, якщо ***<початок> > <кінець>*** (***<крок> > 0***) або, якщо ***<початок> < <кінець>*** (***<крок> < 0***), то цикл не буде виконаний жодного разу – програма продовжить своє виконання з оператору, який знаходиться після оператора **Next**. Тому значення параметру циклу в операторі **For** слід ретельно перевіряти.

Одним з типових прикладів циклічних алгоритмів є задача табулювання функції. Табулювання функції – це обчислення значень функції при зміні аргументу від деякого початкового значення до деякого кінцевого значення з певним кроком.

[29] Наведено приклад циклічного алгоритму, коли кінцеве значення більше за початкове.

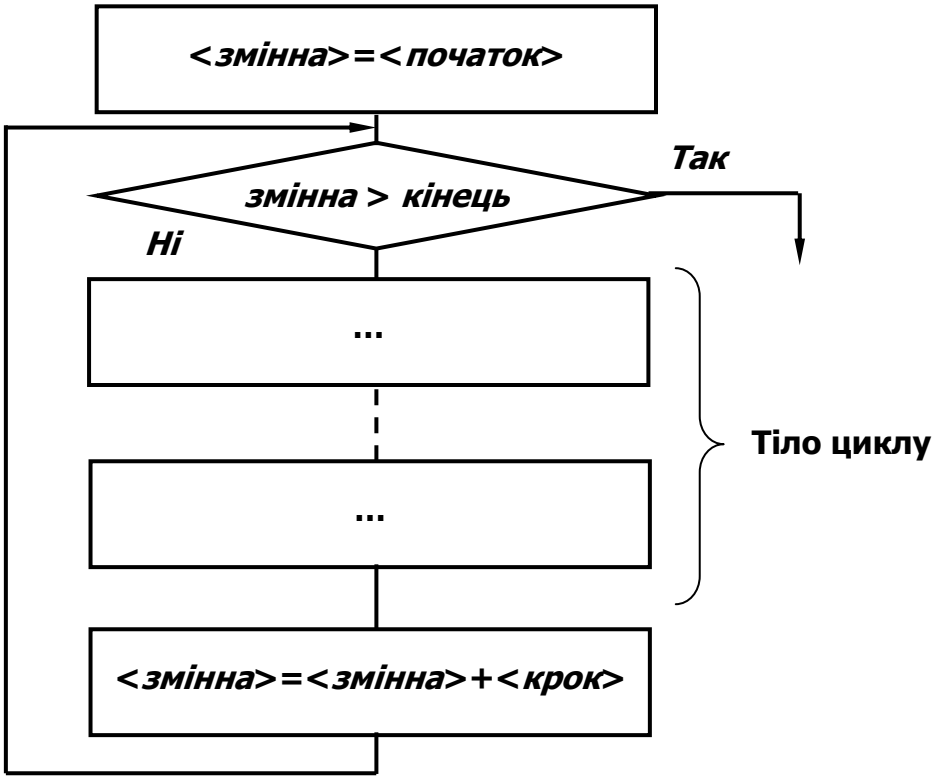
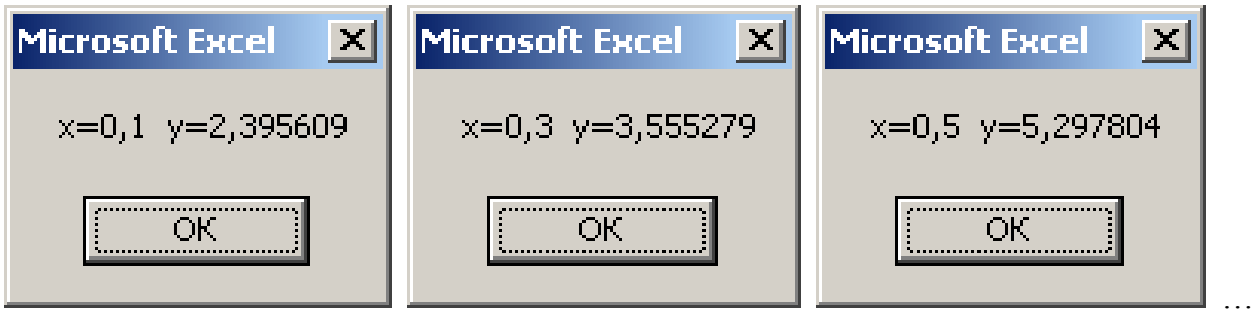



Рис. 4.1 – Алгоритм роботи оператора циклу (коли ***<крок>*** додатний)

Нижче наведено приклад програми табулювання функції ***y = f(x)***, де $y = \alpha \sin x^2 + e^{\beta x}$:

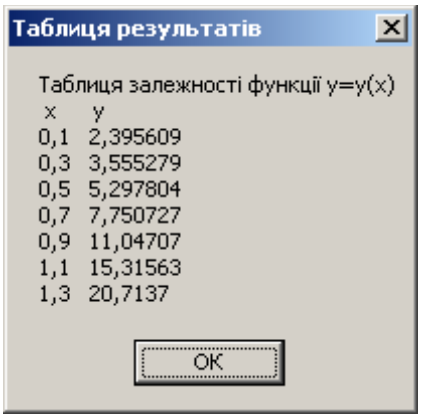
```
Public Sub Tabul1()
   '
   ' Приклад циклічної програми табулювання функції  y = f(x)
   '
   Dim alfa As Single, beta As Single, xn As Single, xk As Single, dx As Single
   Dim x As Single, y As Single
   alfa = InputBox("Введіть коефіцієнт alfa", "Введення коефіцієнтів")
   beta = InputBox("Введіть коефіцієнт beta", "Введення коефіцієнтів")
   xn = InputBox("Введіть початкове значення x – xn", "Введення даних")
   xk = InputBox("Введіть кінцеве значення x – xk", "Введення даних")
   dx = InputBox("Введіть крок змінення x – dx", "Введення даних")
   For x = xn To xk Step dx
       y = alfa * Sin(x ^ 2) + 2 * Exp(beta * x)
       MsgBox "x=" & x & "  y=" & y
   Next x
End Sub
```

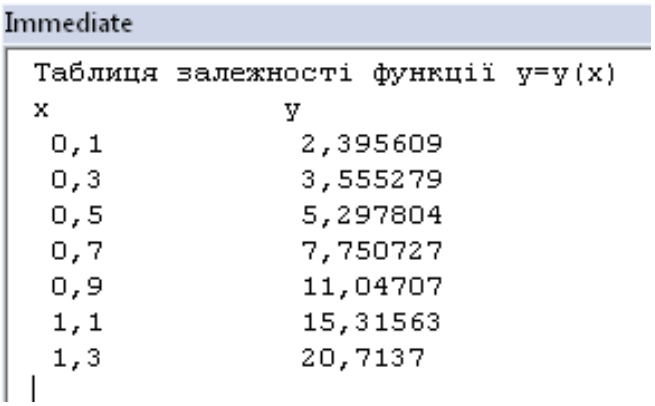


…

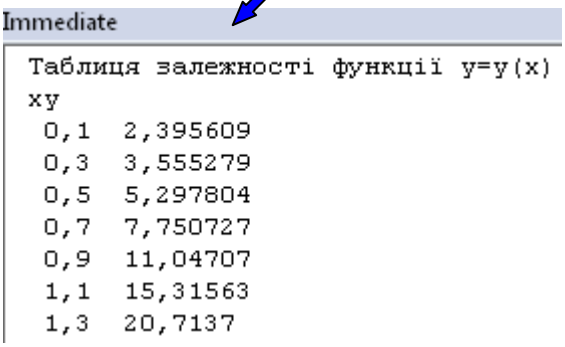
```
Public Sub Tabul2()
    '
    ' Приклад циклічної програми табулювання функції  y = f(x)
    ' Формування таблиці для виведення у вікні функції MsgBox
    '
    Dim alfa As Single, beta As Single, xn As Single, xk As Single, dx As Single
    Dim x As Single, y As Single, sTab As String
    alfa = InputBox("Введіть коефіцієнт alfa", "Введення коефіцієнтів")
    beta = InputBox("Введіть коефіцієнт beta", "Введення коефіцієнтів")
    xn = InputBox("Введіть початкове значення x – xn", "Введення даних")
    xk = InputBox("Введіть кінцеве значення x – xk", "Введення даних")
    dx = InputBox("Введіть крок змінення x – dx", "Введення даних")
    sTab = "Таблиця залежності функції  y=y(x)"
    sTab = sTab & vbNewLine & " x " & "     y  "
    For x = xn To xk Step dx
        y = alfa * Sin(x ^ 2) + 2 * Exp(beta * x)
        sTab = sTab & vbNewLine & x & "   " & y
    Next x
    MsgBox sTab, , "Таблиця результатів"
End Sub

Public Sub Tabul3()
    ' Приклад циклічної програми - табулювання функції  y = f(x)
    ' Виведення таблиці у вікні Immediate
    Dim alfa As Single, beta As Single, xn As Single, xk As Single, dx As Single
    Dim x As Single, y As Single, sTab As String
    alfa = InputBox("Введіть коефіцієнт alfa", "Введення коефіцієнтів")
    beta = InputBox("Введіть коефіцієнт beta", "Введення коефіцієнтів")
    xn = InputBox("Введіть початкове значення x - xn", "Введення даних")
    xk = InputBox("Введіть кінцеве значення x - xk", "Введення даних")
    dx = InputBox("Введіть крок змінення x - dx", "Введення даних")
    Debug.Print "Таблиця залежності функції y=y(x)"
    Debug.Print "x", "y"
    For x = xn To xk Step dx
        y = alfa * Sin(x ^ 2) + 2 * Exp(beta * x)
        Debug.Print x, y
    Next x
End Sub
```

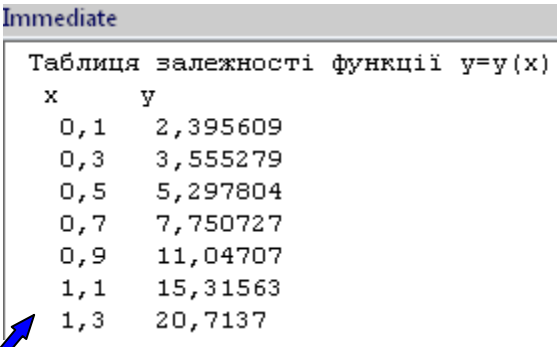
Результати виконання програм **Tabul2** та **Tabul3**:




```
Debug.Print "Таблиця залежності функції y=y(x)"
Debug.Print "x"; "y"
For x = xn To xk Step dx
    y = alfa * Sin(x ^ 2) + 2 * Exp(beta * x)
    Debug.Print x; y
Next x
```



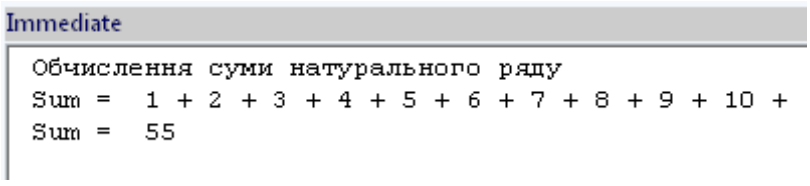

```
Debug.Print "Таблиця залежності функції y=y(x)"
Debug.Print Tab(3); "x"; Tab(9); "y"
For x = xn To xk Step dx
    y = alfa * Sin(x ^ 2) + 2 * Exp(beta * x)
    Debug.Print Tab(2); x; Tab(8); y
Next x
```

### Обчислення сум та добутків

Обчислення сум та добутків відбувається шляхом накопичення. Як видно з подальшого прикладу, сума накопичується у змінній **Sum**, який попередньо надано нульове значення. В циклі на кожному кроці до змінної **Sum** додається один доданок (для даного прикладу – одне з чисел натурального ряду – **i**), а цикл повторюється стільки разів, скільки доданків в сумі.

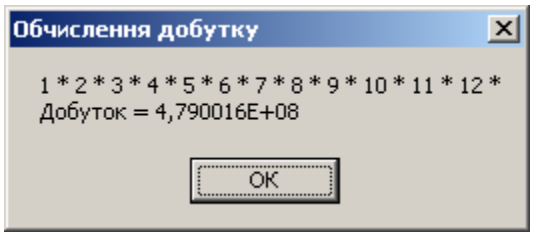
```
Public Sub Suma()
    ' Приклад накопичення суми чисел натурального ряду
    Dim i As Integer, n As Integer, sRez As String, Sum As Single
    n = InputBox("Введіть значення n")
    Sum = 0
    For i = 1 To n
        Sum = Sum + i
        sRez = sRez & i & " + "
    Next i
    sRez = sRez & vbNewLine & "Suma=" & Sum
    MsgBox sRez, , "Обчислення суми"
End Sub
```


```
Public Sub Suma_D()
    ' Приклад накопичення суми чисел натурального ряду
    Dim i As Integer, n As Integer, sRez As String, Sum As Single
    n = InputBox("Введіть значення n")
    Sum = 0
    Debug.Print "Обчислення суми натурального ряду"
    Debug.Print "Sum = ";
    For i = 1 To n
       Sum = Sum + i
       Debug.Print i; "+";
    Next i
    Debug.Print
    Debug.Print "Sum = "; Sum
End Sub
```




```
Public Sub Dobutok()
    ' Приклад накопичення добутку чисел натурального ряду
    Dim i As Integer, n As Integer, sRez As String, Dob As Integer
    n = InputBox("Введіть значення n")
    Dob = 1
    For i = 1 To n
        Dob = Dob * i
```

```
        sRez = sRez & i & " * "
    Next i
    sRez = sRez & vbNewLine & "Добуток = " & Dob
    MsgBox sRez, , "Обчислення добутку"
End Sub
```




Примітка. Якщо **Dob As Integer**, при **n>7** виникає переповнення (**Overflow**)

Якщо **Dob As Single**, при **n>34** виникає переповнення (**Overflow**)

```
Public Sub Dobutok_D()
    ' Приклад накопичення добутку чисел натурального ряду
    Dim i As Integer, n As Integer, sRez As String, Dob As Double
    n = InputBox("Введіть значення n")
    Dob = 1
    Debug.Print "Обчислення добутку чисел натурального ряду"
    Debug.Print "Dob = ";
    For i = 1 To n
        Dob = Dob * i
        Debug.Print i; "*";
    Next i
    Debug.Print
    Debug.Print "Dob = "; Dob
End Sub
```