МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ ТА ПРОГРАМУВАННЯ

2. ПРОГРАМУВАННЯ ТИПОВИХ ЗАДАЧ

МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ
ЛАБОРАТОРНИХ РОБІТ ТА САМОСТІЙНОЇ РОБОТИ
ДЛЯ СТУДЕНТІВ НАПРЯМУ ПІДГОТОВКИ
6.050202 «АВТОМАТИЗАЦІЯ ТА КОМП'ЮТЕРНО-ІНТЕГРОВАНІ ТЕХНОЛОГІЇ»

НАВЧАЛЬНЕ ЕЛЕКТРОННЕ ВИДАННЯ

Київ 2014

Комп'ютерні технології та програмування 2. Програмування типових задач: метод. вказівки до викон. лаб. робіт та самост. роботи для студ. напряму підготовки 050202 «Автоматизація та комп’ютерно-інтегровані технології» (Навч. електронне видання)/ Автори: О.О. Квітка, А.М. Шахновський, С.Л. Мердух – К.: 2014. – 59 с.

Гриф надано Вченою радою ХТФ,
протокол № 5
від «26» травня 2014 р.

Навчальне електронне видання

КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ ТА ПРОГРАМУВАННЯ
2. ПРОГРАМУВАННЯ ТИПОВИХ ЗАДАЧ

методичні вказівки до виконання

лабораторних робіт та самостійної роботи

для студентів напряму підготовки

6.050202 «Автоматизація та комп’ютерно-інтегровані технології»

Автори: Квітка Олександр Олександрович
Шахновський Аркадій Маркусович
Мердух Світлана Леонідівна

Відповідальний редактор: С.Г. Бондаренко, к.т.н., доц.

© О.О. Квітка, А.М. Шахновський, 2014 р.

ЗМІСТ

ПЕРЕДМОВА

Розвиток науки та промисловості вимагає від сучасних фахівців із комп'ютерно-інтегрованих технологій, хімічної технології та інженерії високого рівня професійної підготовки в області інформаційних технологій.

Кваліфікований фахівець з напряму підготовки "Автоматизація та комп'ютерно-інтегровані технології" має успішно застосовувати у своїй професійній діяльності знання та навички «прикладного програмування», зокрема алгоритмізації практичних задач, поставлених замовником, розробки раціональної архітектури та створення засобами сучасних середовищ програмування програмних продуктів з метою виконання технологічних та техніко-економічних обчислень, програмування засобів автоматизації, тощо.

Створення ефективних програмних продуктів у сучасних середовищах програмування, крім володіння синтаксисом відповідної мови програмування, специфічними засобами середовища програмування, та знання візуальних компонентів, вимагає від програміста насамперед вміння алгоритмізувати вихідну задачу.

Оскільки мова програмування Visual Basic використовується, зокрема, для автоматизації додатків MS Office, при програмуванні багатьох засобів в складі автоматизованих систем керування технологічними процесами, тощо, то можна сміливо твердити, що використання MS Visual Basic for Applications у якості навчального середовища програмування не тільки дає можливість опанувати базові навички алгоритмізації, а й надає можливість здобути навички автоматизації задач, що виникають при роботі у поширених прикладних програмах (в першу чергу – в MS Excel), а також дає також змогу студентам засвоїти синтаксис мови програмування сімейства Visual Basic, яка є однією з найпоширеніших в складі засобів створення прикладних програмних продуктів.

Методичні вказівки розроблено відповідно до програми підготовки бакалаврів за напрямом підготовки «Автоматизація та комп'ютерно-інтегровані технології» і є складовою частиною дисципліни «Комп'ютерні технології та програмування». Ця дисципліна є базовою у підготовці бакалаврів вказаного напряму.

У методичних вказівках наведені мета та завдання для лабораторних робіт, стисло викладений необхідний теоретичний матеріал та зазначено літературу для поглибленого вивчення, надано порядок виконання робіт та обробки результатів, перелічено вимоги до оформлення звітів та порядку їх подання, сформульовано контрольні питання для самопідготовки студентів та визначені засоби безпеки, яких слід дотримуватися при виконанні робіт.

1. Програми складної структури

Мета та основні завдання роботи: дослідити особливості організації програм, що мають складну структуру – складаються з кількох процедур, та особливості обміну даними між цими процедурами.

1.1 Основні теоретичні відомості

Сучасні програми зазвичай мають складну структуру – вони побудовані з деякої кількості підпрограм, кожна з яких реалізує певний алгоритм і є автономною частиною програми. При використанні такого *модульного принципа побудови*, програми складаються з основної (головної) процедури і набору інших процедур і функцій користувача. До процедури (*підпрограми*) можна звернутися з будь-якої точки програми, призупинивши, таким чином, виконання основної програми. Застосування цього інструменту надає програмістові додаткові, гнучкіші можливості при розробці нових програм.

Кожна процедура є автономною (незалежною від інших) частиною програми. При виконанні програми кожна з них займає окрему область оперативної пам'яті не "перемішуючись" з іншими процедурами або функціями. Тому, змінні, використовувані в різних процедурах (що входять до складу однієї програми), є незалежними одна від одної, навіть, якщо вони мають однакові імена, і кожна з них має "своє" значення. Завдяки цьому, зміна значення змінної в одному модулі не зможе викликати несподіваної зміни в іншому модулі, що в досить великих програмах є частим джерелом помилок. Наприклад, в основній програмі використовується деяка змінна *alfa* і в якомусь іншому модулі також є змінна *alfa*. Хоча ці дві змінні мають однакові імена, вони зберігаються в різних "комірках" пам'яті, тобто фактично вони є абсолютно різними змінними. Змінні, відомі в межах тільки однієї процедури називаються *локальними*. Обмін даними між декількома процедурами при необхідності має бути спеціально організований. Тільки в цьому випадку значення змінних, відомі в одному модулі, стають відомими в іншому.

Технологія проектування програми для розв’язання деякої задачі полягає в тому, що задача розподіляється на певну кількість підзадач, кожна з яких є автономною. Кожна така підзадача програмується як окрема процедура (підпрограма або функція). Для правильної роботи всієї програми необхідно налагодити взаємодію між всіма процедурами, з яких складається програма. Для обміну інформацією між процедурою, що викликається, і процедурою, яка викликає, найчастіше використовується механізм *формальних* та *фактичних параметрів*.

Формальні параметри характеризуються тим, що до початку роботи підпрограми не мають ніяких конкретних значень. Вони використовуються для того, щоб формально (абстрактно по відношенню до даних) описати порядок

виконання операцій (розрахунку) в процедурі. *Підпрограма* (*процедура*) має таку структуру:

[<*Статус*>] Sub <*Ім'я*> [(<*Список формальних параметрів*>)]

```
. . .
. . .
 [ Exit Sub ]
. . .
```

} операторы, що входять до складу підпрограми ("*тіло підпрограми*")

End Sub

Ім'я підпрограми повинне відповідати правилу формування імен в Visual Basic. **[<Статус>]** процедури (найчастіше це – **Public** або **Private**) визначає умови її роботи. До процедури, яка позначена **Public** (за умовчанням), можна звернутись з програм, що зберігаються в будь-якому іншому модулі. А, якщо процедура позначена як **Private**, до неї можна звернутись тільки з програм, які зберігаються в тому модулі, де описана ця процедура (або функція).

Список формальних параметрів – формується з тих змінних (з числа використовуваних в підпрограмі), через які відбувається передача необхідних даних в підпрограму і повернення результатів з підпрограми. До списку формальних параметрів можуть входити імена як простих змінних, так і масивів. Якщо формальний параметр є масивом, після його імені записуються порожні дужки та вказується його тип (наприклад, ***x*() As Integer**). Такі масиви не слід описувати в підпрограмі ще раз за допомогою оператора DIM – цей опис буде повторним і викличе помилку. Формальні параметри в списку розділяються комами. В разі необхідності дострокового завершення роботи підпрограми і "виходу" з неї в "тілі підпрограми" може бути використаний оператор **Exit Sub**.

Для звернення до підпрограми використовується спеціальний оператор. Він може мати вигляд:

Call <*Ім'я_підпрограми*> [(<*Список фактичних параметрів*>)]

або:

***Ім'я_підпрограми* [<*Список фактичних параметрів*>]**

Фактичні параметри – це змінні, значення яких передаються в підпрограму, що викликається, для виконання розрахунків. Значення фактичних параметрів передаються відповідним формальним параметрам зі списку оператора **Sub** і з цими значеннями виконується підпрограма.

До переліку *формальних* та *фактичних параметрів* включають змінні, які зберігають вхідну та вихідну інформацію процедури, що викликається. При виклику процедури значення кожного фактичного параметру передається відповідному формальному параметру (1-й → 1-му, 2-й → 2-му, …). Після закінчення роботи процедури відбувається зворотна передача інформації. Тому при зверненні до процедури ці списки повинні відповідати один одному:

– за порядком/кількістю;
– за типом (способом зберігання значення);
– за видом (проста змінна або масив).

Фактичні параметри можуть бути константами, простими змінними і окремими елементами массиву (якщо відповідний формальний параметр є простою зміною) або массивом (якщо відповідний формальний параметр також є масивом). Якщо параметр є масивом, то після його імені записуються порожні дужки. Фактичні параметри в списку розділяються комами.

Розглянемо використання процедур на прикладі задачі пошуку найменших елементів в двох довільних масивах. В головній процедурі задаються розміри масивів, виконується їх опис. Для надання значень масивів викликається процедура **ArrayInput.** Для підготовки виведення інформації у вікно функції **MsgBox** кожного разу викликається процедура **ArrayPrint.** Після звернення до процедури пошуку найменшого елементу **ArrayMin** кожного з масивів за допомогою функції **MsgBox** виводиться отриманий результат:

```
Public Sub Primer1_1()
  Dim iNumbA As Integer, iNumbB As Integer, ii As Integer
  Dim iMinA As Integer, iMinB As Integer, iNA As Integer
  Dim iNB As Integer, sPrint As String
  Dim a() As Integer, b() As Integer,
  iNumbA = InputBox("    Dim розмір масиву A", "Введення даних")
  iNumbB = InputBox("Задайте розмір масиву B", "Введення даних")
  ReDim a(iNumbA) As Integer, b(iNumbB) As Integer
  Range("A1").Select
  Call ArrayInput(iNumbA, a())
  Call ArrayPrint(iNumbA, a(), "A", sPrint)
  Call ArrayMin(iNumbA, a(), iMinA, iNA)
  sPrint = sPrint & vbNewLine & "Min = " & iMinA & vbNewLine & "Це елемент №" & iNA
  MsgBox sPrint, , "Результат"
  Call ArrayInput(iNumbB, b())
  Call ArrayPrint(iNumbB, b(), "B", sPrint)
  Call ArrayMin(iNumbB, b(), iMinB, iNB)
  sPrint = sPrint & vbNewLine & "Min = " & iMinB & vbNewLine & "Це елемент №" & iNB
  MsgBox sPrint, , "Результат"
End Sub

Public Sub ArrayMin(iNn, x() As Integer, Xmin, iNumb)
  'Пошук найменшого елементу в масиві x
  Dim ii As Integer
  Xmin = x(1)
  iNumb = 1
  For ii = 2 To iNn
     If x(ii) < Xmin Then
        Xmin = x(ii)
```

```
        iNumb = ii
      End If
  Next ii
End Sub

Public Sub ArrayInput(iNn, x() As Integer)
  'Введення випадкоаих значень елементів масиву x
  Dim ii As Integer
  Randomize Timer
  For ii = 1 To iNn
    x(ii) = Int(Rnd * 100)
  Next ii
End Sub

Public Sub ArrayPrint(iNn, x() As Integer, ArName, sPrint)
  'Виведення масиву x на лист MS Excel
  Dim ii As Integer
  sPrint = "Масив " & ArName & vbNewLine
  For ii = 1 To iNn
    If ii = iNn Then
       sPrint = sPrint & x(ii)
    Else
       sPrint = sPrint & x(ii) & ", "
    End If
  Next ii
End Sub
```

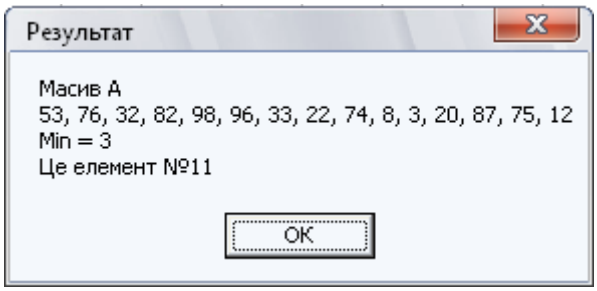
Отже головна програма двічи «звертається» до кожної з використаних процедур, кожного разу з різними фактичними парметрами. Нижче для порівняння наведені список формальних парметрів процедури **ArrayMin** та відповідні списки фактичних парметрів при звертанні до неї (табл. 1.1.).

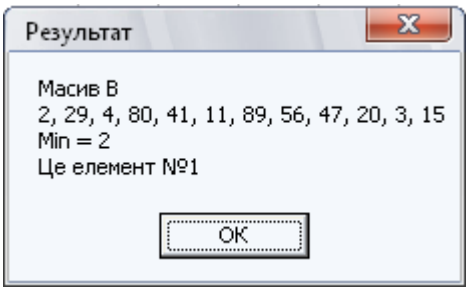
Таблиця 1 Формальні та фактичні параметри для підпрограми **ArrayMin**

Формальні	**iNn**	**x()**	**Xmin**	**iNumb**
Фактичні – 1	**iNumbA**	**a()**	**iMinA**	**iNA**
Фактичні – 2	**iNumbB**	**b()**	**iMinB**	**iNB**

Як видно з цього порівняння всі списки співпадають за усіма необхідними критеріями. При цьому, співпадання або неспівпадання імен формальних і фактичних параметрів значення не має. Таким чином, використовуючи різні фактичні параметри при зверненні до процедури можна "змусити" її працювати з різними початковими даними не міняючи текст операторів процедури.

Результат роботи програми:





Іншим способом обміну даними між процедурами є використання опису необхідних змінних «на рівні модуля». Змінні, які описані «на рівні модуля», можуть бути використані в будь-якій процедурі або функції, що зберігаються у цьому модулі. Розглянемо використання цього способу обміну інформацією між процедурами в наступному прикладі. Масив **Matr()** та змінні, що вистачають його розміри – **iRows** та **iCols** – описані «на рівні модуля». Тому вони доступні для всіх процедур програми і їх не треба передавати через списки формальних та фактичних параметрів.

Приклад №2. Знайти суми додатних елементів в парних стовпцях довільної матриці.

```
Option Explicit
Dim Matr() As Single
Dim iRows As Integer, iCols As Integer

Public Sub Primer2()
   Dim iCol As Integer, SumDod As Single
   Call MatrEnter
   Call MatrPrint
   ActiveCell.Value = "Суми:"
   Call FmtFt(12, 1, 15)
   ActiveCell.Offset(0, 1).Select
   For iCol = 2 To iCols Step 2
      Call ColSum(iCol, SumDod)
      ActiveCell.Value = SumDod
      Call FmtFt(12, 1, 15)
      Call FmtBorders1
      ActiveCell.Offset(0, 2).Select
   Next iCol
   Call MatrFormat
End Sub

Public Sub MatrEnter()
  Dim iRow As Integer, iCol As Integer
  iRows = InputBox("Для створення матриці вкажіть кількість рядків", "Розмір матриці")
```

```
  iCols = InputBox("та кількість стовпців", "Розмір матриці")
  ReDim Matr(iRows, iCols)
  Randomize Timer
  For iRow = 1 To iRows
    For iCol = 1 To iCols
      Matr(iRow, iCol) = Int(Rnd * 200 - 100)
    Next iCol
  Next iRow
End Sub

Public Sub MatrPrint()
  Dim iRow As Integer, iCol As Integer
  Range("A1").Select     'Комірка, де друкується заголовок матриці
  ActiveCell.Value = "Вихідна матриця"
  Range(Cells(1, 1), Cells(1, iCols)).Select
  Call FmtFt(12, 1, 15)
  Range("A2").Select
  For iRow = 1 To iRows
    For iCol = 1 To iCols
      ActiveCell.Value = Matr(iRow, iCol)
      ActiveCell.Offset(0, 1).Select        'Перехід до сусідньої комірки праворуч
    Next iCol
    ActiveCell.Offset(1, -iCols).Select       'Перехід до початку наступного рядка
  Next iRow
End Sub

Public Sub FmtFt(iFs As Integer, iFI As Integer, iCI As Integer)
   Selection.Font.Size = iFs
   If iFI = 1 Then
      Selection.Font.Bold = True
   End If
   Selection.Interior.ColorIndex = iCI
End Sub

Public Sub ColSum(iCn, SumDod)
  Dim iRow As Integer, elt As Single
  SumDod = 0
  For iRow = 1 To iRows
    elt = Matr(iRow, iCn)
    If elt > 0 Then
      SumDod = SumDod + elt
    End If
  Next iRow
End Sub

Public Sub MatrFormat()
   Range(Cells(2, 1), Cells(2 + iRows - 1, iCols)).Select
```

```
    Selection.NumberFormat = "0"
    Selection.HorizontalAlignment = xlCenter
    Call FmtFt(11, 0, xlNone)
    Call FmtBordersN
    Range("A1").Select
End Sub

Public Sub FmtBordersN()
    Selection.HorizontalAlignment = xlCenter
    With Selection.Borders(xlEdgeLeft)
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeTop)
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeBottom)
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeRight)
        .Weight = xlThin
    End With
    With Selection.Borders(xlInsideHorizontal)
        .Weight = xlThin
    End With
    With Selection.Borders(xlInsideVertical)
        .Weight = xlThin
    End With
End Sub

Public Sub FmtBorders1()
    With Selection.Borders(xlEdgeLeft)
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeTop)
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeBottom)
        .Weight = xlThin
    End With
    With Selection.Borders(xlEdgeRight)
        .Weight = xlThin
    End With
    Selection.HorizontalAlignment = xlCenter
End Sub
```

Результат роботи програми:

	A	B	C	D	E	F	G
1	**Вихідна матриця**						
2	93	84	-97	58	-13	-93	-31
3	-93	-41	91	-32	-97	36	-85
4	-25	91	97	-90	-87	-39	84
5	7	99	25	73	-57	33	48
6	-15	-85	37	-79	-44	82	-50
7	-6	-52	-84	64	54	19	9
8	**Суми:**	**274**		**195**		**170**	

1.2 Вказівки до самостійної роботи

Самостійна робота є необхідним елементом підготовки до практичної і лабораторної робіт в комп'ютерному класі. Під час підготовки до виконання роботи вивчити та описати в протоколі лабораторної роботи такі поняття (визначення) і операції (команди):

- поняття програми програми складної структури;
- особливості організації програм, побудованих з використанням кількох процедур (підпрограм);
- механізм виклику підпрограм та організацію обміну інформацією з процедурою, що викликається (списки формальних и фактичних параметрів);
- технологію створення та налаштування програм, що включають підпрограми.
- різницю між описом змінних на рівні процедури та на рівні модуля;
- створити варіант програми для прикладу №1, який використовує процедуру виведення результатів роботи на лист MS Excel:

Необхідно також повторити:

- поняття двовимірного масиву;
- особливості прийомів роботи з такими масивами (опис, введення та друк масивів, доступ до окремих елементів масиву, як по рядках, так и по стовпцях, а також по діагоналях);
- характерні прийоми програмування і типові алгоритми (пошук елементів масиву у відповідності до заданої ознаки: знак числа; кратність числа заданому коефіцієнту; найменший чи найбільший елемент).
- розробити програму (відповідно своєму варіанту завдання – Додаток Г[1]), яка реалізує поставлену задачу у окремій процедурі (підпрограмі). Організувати виведення інформації в формі таблиці. Організувати також запис інформації в файл;
- налаштувати програму та перевірити її роботу;
- продемонструвати роботу програми викладачу.

[1] За вказівкою викладача.

1.3 Опис лабораторних засобів та обладнання

Лабораторна робота №1 виконується на персональному комп'ютері стандарту IBM PC під керуванням операційної системи MS Windows.

1.4 Заходи безпеки під час виконання лабораторної роботи

Заходи безпеки, яких треба дотримуватись при виконанні даної лабораторної роботи, наведені у додатку А.

1.5 Послідовність виконання лабораторної роботи № 1.

1. Відповідно до свого варіанту розробити алгоритм розв'язання задачі та реалізувати його у вигляді окремої процедури, яка викликається з головної процедури[2].
2. При розробці програми слід дотримуватись таких вимог:
   - ✓ введення вихідних значень та виведення результатів повинно виконуватись в головній процедурі (або в окремих процедурах[3]);
   - ✓ введення значень матриці слід виконувати або за допомогою датчика випадкових чисел, або з файлу даних, або з листа MS Excel;
   - ✓ результати виводити[2] у вікно функції ***MsgBox***[4], у вікно ***Immediate***[3] та на лист MS Excel[3];
   - ✓ результати, виведені на лист MS Excel, необхідно відформатувати[5];
   - ✓ організувати запис вхідної та вихідної інформації в файл.
3. Продемонструвати роботу програми викладачу.
4. Оформити протокол лабораторної роботи.

1.6 Обробка та аналіз результатів. Оформлення звіту

Звіти з усіх лабораторних робіт виконуються на листах формату А4[6]. Протокол кожної лабораторної роботи починається з номера та теми роботи, прізвища студента та шифру групи. Наприклад:

---

[2] Головну процедуру слід називати за прізвищем студента + номер лабораторної роботи (наприклад – **Sub Klymenko_1**).

[3] Бажано за допомогою окремих процедур.

[4] Передбачити друкування як вихідних масивів, так і отриманих для них результатів. Передбачити запис результатів, які отримані для певного рядка або стовпця, у комірки, що знаходяться у відповідному рядку або стовпці

[5] Для форматування (границі, шрифти, виділення кольором потрібних комірок) створити окрему процедуру (або декілька), що викликається з головної процедури.

[6] З дозволу викладача допускається оформлення протоколів лабораторних робіт в зошиті. В такому випадку всі роздруковані тексти програм, розрахунків, блок-схем повинні бути вклеєні у відповідні місця протоколів.

Лабораторна робота № 1

Тема: Програми складної структури

Виконав студент гр. ХА-11
Козаченко В.І.

До протоколу лабораторної роботи слід включати:

- мету роботи;
- короткі теоретичні відомості[7];
- результати самостійної роботи;
- номер індивідуального варіанту;
- повну умову кожної поставленої індивідуальної задачі;
- алгоритм (блок-схему)[8] розв'язку кожної задачі;
- роздрукований текст програми[9];
- роздруковані результати виконання розрахунків та вихідних даних, при яких вони отримані[10;]
- висновки.

Після захисту лабораторної роботи протокол, підписаний викладачем, зберігається студентом в окремому файлі[11]. По закінченні циклу лабораторних робіт протоколи всіх робіт збираються разом в окрему папку з титульним листом (додаток В) і здаються викладачеві.

1.7 Контрольні запитання

1. Яку структуру має підпрограма (процедура)?
2. Що таке формальні і фактичні параметри?
3. Яким вимогам повинні відповідати списки формальних і фактичних параметрів?
4. Які змінні називаються глобальними і які локальними?
5. Як відбувається обмін даними між процедурами?
6. Як розміщуються в пам'яті комп'ютера різні процедури (підпрограми), що входять до складу однієї програми?
7. Як розміщуються тексти різних процедур однієї програми в середовищі VBA?

---

[7] Ця частина протоколу готується при підготовці до лабораторної роботи і перевіряється викладачем перед виконанням роботи згідно РСО.

[8] Правила запису блок-схем наведені в Додатку Б.

[9] Текст кожної програми повинен починатися з рядків коментарів, де вказано прізвище студента та назва групи, тема роботи та № індивідуального варіанту. заголовки всіх процедур (функцій користувача) необхідно виділити.

[10] Вихідні дані та результати розрахунків при роздруківці повинні супроводжуватись текстовими поясненнями (наприклад, <Температура = 315 K> або <y = -14.53>).

[11] Необхідно зберігати електронні копії всіх розроблених програм до кінця семестру на своїй дискеті (флешці).

8. У яких випадках змінні, що мають однакові імена але відносяться до різних процедур, набувають одно і те ж значення?
9. Як здійснити достроковий вихід з підпрограми?

2. Створення та використання функцій користувача

Мета та основні завдання роботи: дослідити особливості створення та застосування функцій користувача.

2.1 Основні теоретичні відомості

У тих випадках, коли результатом роботи деякого алгоритму є якесь одне значення (воно може бути отримане як із однієї формули, так і з цілої послідовності обчислень), такий алгоритм може бути реалізований у вигляді *функції користувача*. До функції користувача можна звертатися з будь-якої іншої підпрограмни, що входить до складу цієї програми.

Будь-яка функція користувача має таку структуру:

```
[<Статус>] Function <Ім’я> [(<Список формальних параметрів>)] [As type]
        . . .
        . . .
                    [ Exit Function]
        . . .
    <Ім’я_ функції> = <результат>
End Function
```

(рядки між заголовком і **End Function** — оператори, що входять до складу функції ("*тіло функції*"))

Ім'я функції повинне відповідати правилу формування імен в Visual Basic. *Список формальних параметрів* – це список тих змінних (з числа використовуваних в підпрограмі), через які відбувається передача вхідних даних при виклику функції. Для передачі обчисленого значення функції в модуль, звідки викликано функцію, необхідно обов’язково присвоїти це значення змінній, співпадаючій по імені та типу з ім'ям функції. Для дострокового завершення роботи зовнішньої функції і виходу з неї використовується оператор **Exit Function**.

Звернення до зовнішньої функції відбувається також, як і звернення до будь-якої стандартної функції. При цьому, після імені функції в дужках записується *список фактичних параметрів*, що відповідає списку формальних параметрів в операторові **Function**. Використовуються ті ж самі правила, що і для процедури типу **Sub**.

Найпростішим прикладом застосування функції користувача є відома задача табулювання функції. В цьому прикладі функція функція представлена однією формулою:

Функція користувача, що задає вигляд функції:

```
Public Function MyFun(alfa, beta, x) As Single
   MyFun = alfa * Sin(x ^ 2) + 2 * Exp(beta * x)
End Function
```

Головна програма, що виконує табулювання функції:

```
Public Sub Tabul2()
   ' 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, 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 = MyFun(alfa, beta, x)
      sTab = sTab & vbNewLine & x & "   " & y
   Next x
   MsgBox sTab, , "Таблиця результатів"
End Sub
```

Функція може бути записана, не тільки, як одна формула, але і обчислюватись за допомогою будь-якої кількості операторів. Розглянемо наступні приклади:

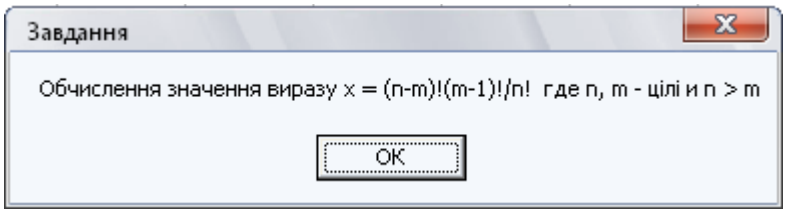
```
Public Function Suma(iNn) As Single
'Функція обчислення суми чисел натурального ряду
Dim ii As Integer
  Suma = 0
  For ii = 1 To iNn
    Suma = Suma + ii
  Next ii
End Function

Public Sub Primer2_1()
'Процедура зі звертанням до функції обчислення суми чисел натурального
ряду
  Dim iNumb As Integer, Sum As Single
  iNumb = InputBox("Для обчислення суми чисел натурального ряду, введіть
                                                  кінцеве число", " Введення")
  Sum = Suma(iNumb)
  MsgBox " Сума дорівнює " & Sum, , "Результат"
End Sub
```

Нижче наведено приклад використання функції, яка розраховує значення факторіалу. В цьому випадку обчислення функції виконується в циклі, а, крім того виконуються допоміжні перевірки. Ця функція використовується при обчисленні наступної величини:

$$x = \frac{(n-m)! * (m-1)!}{n!} \qquad \text{де } n > m.$$

В даному випадку кількість формальних параметрів обмежується одним – **iK**. Алгоритм, за яким працює функція, наведено на рис. 2.1, сама функція має такий вигляд:

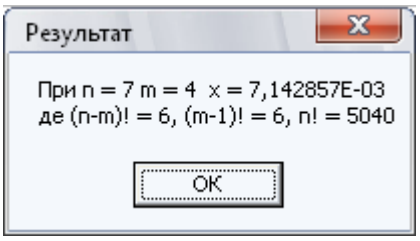
```
Public Function Factor(iK) As Single
'Функція обчислення факторіалу
Dim ii As Integer
   If iK < 0 Then
      MsgBox "Аргумент факторіалу – від'ємний. Помилка!", vbCritical, " Увага!"
      Factor = -1
      Exit Function
   ElseIf iK = 0 Or iK = 1 Then
      Factor = 1
      Exit Function
   End If
   Factor = 1
   For ii = 2 To iK
      Factor = Factor * ii
   Next ii
End Function
```




```
Public Sub Primer2_2()
   'Процедура звертанням до функції обчислення факторіалу
   Dim iNn As Integer, iMm  As Integer, x As Single
    MsgBox "Обчислення значення виразу x = (n-m)!(m-1)!/n!  где n, m - цілі і
                                                    n > m", , "Завдання"
   iNn = InputBox("Введіть значення n", " Введення")
   iMm = InputBox("Введіть значення m", " Введення")
   x = Factor(iNn – iMm) * Factor(iMm – 1) / Factor(iNn)
   MsgBox "При n = " & iNn & " m = " & iMm & "  x = " & x, , "Результат"
End Sub
```

Або:

```
Public Sub Primer2_2a()
   'Процедура звертанням до функції обчислення факторіалу
   Dim iNn As Integer, iMm  As Integer, x As Single
   Dim f1 As Single, f2 As Single, f3 As Single
    MsgBox "Обчислення значення виразу x = (n-m)!(m-1)!/n!  где n, m - цілі і
                                                    n > m", , "Завдання"
   iNn = InputBox("Введіть значення n", " Введення")
   iMm = InputBox("Введіть значення m", "
Введення")
   f1 = Factor(iNn - iMm)
   f2 = Factor(iMm - 1)
   f3 = Factor(iNn)
```



```
   x = f1 * f2 / f3
    MsgBox "При n = " & iNn & " m = " & iMm & "  x = " & x & vbNewLine & "де
              (n-m)! = " & f1 & ",   (m-1)! = " & f2 & ", n! = " & f3, , "Результат"
End Sub
```

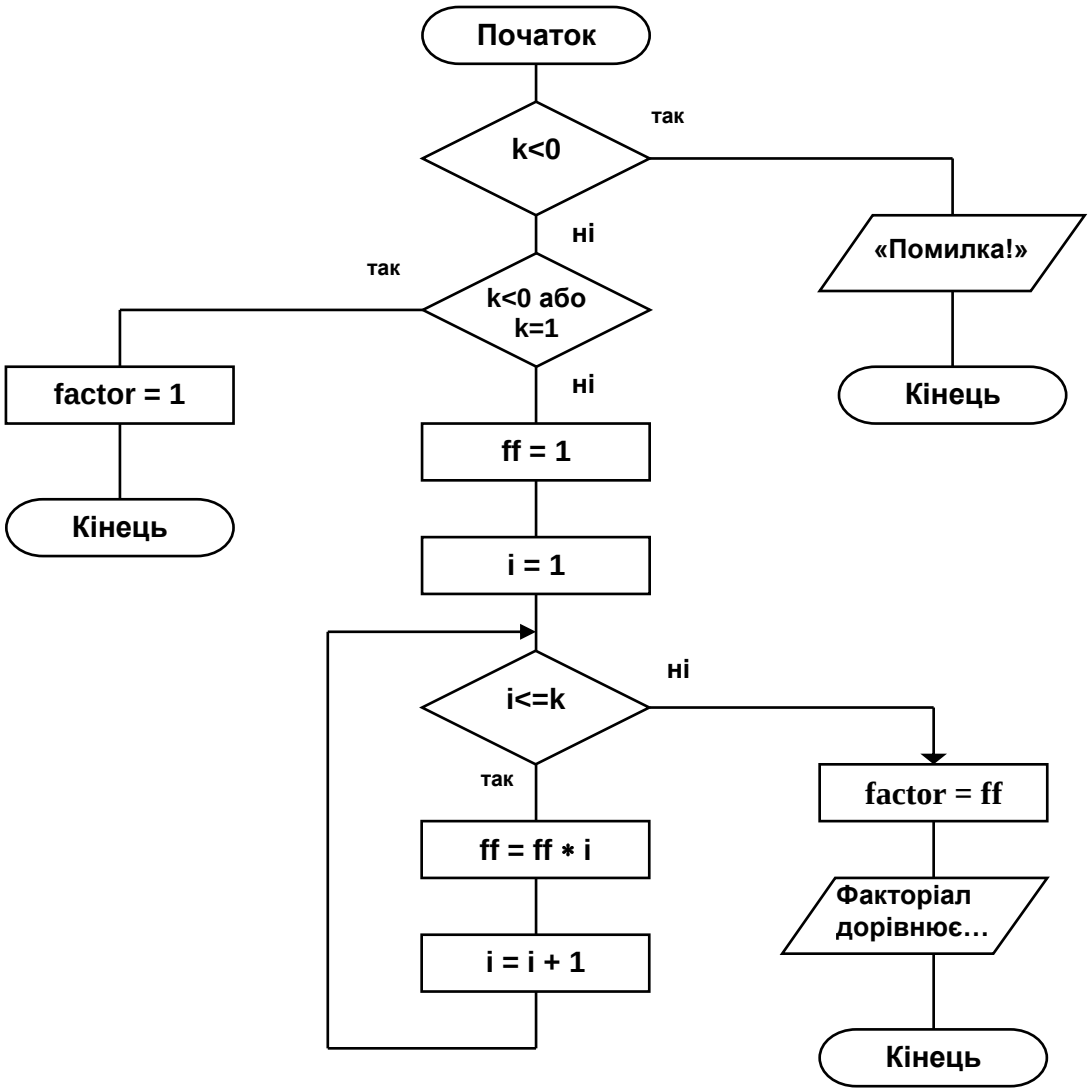


Рисунок 2.1 – Алгоритм обчислення факторіалу k!

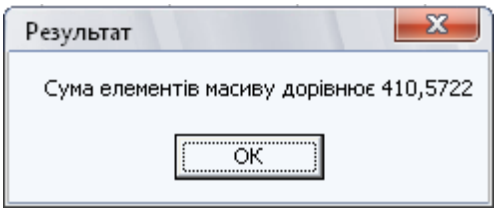
Наступний приклад ілюструє використання опису змінних «на рівні модуля».

```
Option Explicit
Dim iNumb As Integer, x() As Single

Public Function SumArray()
   'Функція обчислення суми елементів масиву
   Dim ii As Integer
   SumArray = 0
   For ii = 1 To iNumb
```

```
      SumArray = SumArray + x(ii)
   Next ii
End Function
```

```
Public Function PrintArray()
   'Функція друкування масиву
   Dim ii As Integer
   Range("A1").Select
   ActiveCell.Value = "Масив x"
   Range("A2").Select
   For ii = 1 To iNumb
      ActiveCell.Value = x(ii)
      ActiveCell.Offset(0, 1).Range("A1").Select
   Next ii
   ActiveCell.Offset(1, -iNumb).Range("A1").Select
End Function
```

```
Public Sub Primer2_3()
'Процедура зі звертанням до функції обчислення суми чисел натурального ряду
   Dim ii As Integer, Sum As Single
   iNumb = InputBox("Введіть розмір масиву", "Задання масиву")
   ReDim x(iNumb) As Single
   Randomize Timer
   For ii = 1 To iNumb
      x(ii) = Rnd * 100
   Next ii
   Sum = SumArray()
   PrintArray
   ActiveCell.Value = "Сума="
   ActiveCell.Offset(0, 1).Range("A1").Select
   ActiveCell.Value = Sum
   MsgBox " Сума елементів масиву дорівнює " & Sum, , "Результат"
End Sub
```



	A	B	C	D	E	F	G	H
1	Масив x							
2	94,22723	72,94994	27,25423	36,67414	60,16611	28,18507	27,39359	63,72189
3	Сума=	410,5722						

Але цей спосіб має певні недоліки, які зрозумілі з наступного прикладу, де треба знайти суми кількох масивів:

```
Option Explicit

Public Function SumArray(iNn, x() As Single) As Single
   'Функція обчислення суми елементів масиву
   Dim ii As Integer
   SumArray = 0
   For ii = 1 To iNn
```

```
    SumArray = SumArray + x(ii)
  Next ii
  MsgBox " Сума елементів масиву дорівнює " & SumArray, , "Результат"
End Function

Public Function PrintArray(iNn, x() As Single)
  'Функція друкування масиву
  Dim ii As Integer
  ActiveCell.Value = "Масив"
  ActiveCell.Offset(1, 0).Range("A1").Select
  For ii = 1 To iNn
     ActiveCell.Value = x(ii)
     ActiveCell.Offset(0, 1).Range("A1").Select
  Next ii
  ActiveCell.Offset(1, -iNn).Range("A1").Select
End Function

Public Sub Primer2_4()
'Процедура зі звертанням до функцій друкування та обчислення суми елементів масиву
  Dim ii As Integer, Sum As Single
  Dim iNumbA As Integer, iNumbB As Integer, iNumbC As Integer
  Dim a() As Single, b() As Single, c() As Single
  iNumbA = InputBox("Введіть розмір масиву A", "Задання масиву")
  iNumbB = InputBox("Введіть розмір масиву B", "Задання масиву")
  iNumbC = InputBox("Введіть розмір масиву C", "Задання масиву")
  ReDim a(iNumbA) As Single, b(iNumbB) As Single, c(iNumbC) As Single
  Randomize Timer
  For ii = 1 To iNumbA
     a(ii) = Int(Rnd * 100)
  Next ii
  Range("A1").Select
  PrintArray iNumbA, a()
  Sum = SumArray(iNumbA, a())
  ActiveCell.Value = "Сума="
  ActiveCell.Offset(0, 1).Range("A1").Select
  ActiveCell.Value = Sum
  ActiveCell.Offset(1, -1).Range("A1").Select
  For ii = 1 To iNumbB
     b(ii) = Int(Rnd * 100)
  Next ii
  PrintArray iNumbB, b()
  Sum = SumArray(iNumbB, b())
  ActiveCell.Value = "Сума="
  ActiveCell.Offset(0, 1).Range("A1").Select
  ActiveCell.Value = Sum
  ActiveCell.Offset(1, -1).Range("A1").Select
```

```
   For ii = 1 To iNumbC
      c(ii) = Int(Rnd * 100)
   Next ii
   PrintArray iNumbC, c()
   Sum = SumArray(iNumbC, c())
   ActiveCell.Value = "Сума="
   ActiveCell.Offset(0, 1).Range("A1").Select
   ActiveCell.Value = Sum
End Sub
```

Результат:

	A	B	C	D	E	F	G	H	I	J	K	L
1	Масив											
2	37	13	89	95	43	1	42	13	80	36	21	53
3	**Сума=**	**523**										
4	Масив											
5	62	14	44	35	80	88	58					
6	**Сума=**	**381**										
7	Масив											
8	6	53	35	41	13	96	42	36	5	15		
9	**Сума=**	**342**										

Приклад 4. Розрахувати значення $G = \sum_{k=M}^{N} \frac{(n+3)^3}{(n-5)^2 (a-2)}$

```
Public Sub Primer4()
   Dim m As Integer, n As Integer, a As Integer, numb As Integer, i As Integer
   Dim s As Single
   m = InputBox("Введіть початкове значення", "Введення даних")
   n = InputBox("Введіть кінцеве значення", "Введення даних")
   numb = InputBox("Скільки варіантів розрахунку?", "Введення даних")
   For i = 1 To numb
      a = InputBox("Введіть значення a", "Варіант " & i)
      s = Suma(m, n, a)
      MsgBox "При a = " & a & " Сума = " & s, , "Результат"
   Next i
End Sub

Public Function Chyselnik(i)
   Chyselnik = (i + 3) ^ 3
End Function

Public Function Znamennyk(k)
   Znamennyk = (k - 5) ^ 2
End Function
```

```
Public Function Suma(m, n, a)
   Dim k As Integer, dodanok As Single
   Suma = 0
   For k = m To n
      chys = Chyselnik(k)
      znam = Znamennyk(k)
      If znam <> 0 Then
         dodanok = chys / znam / (a - 2)
         Suma = Suma + dodanok
      End If
   Next k
End Function
```

Результат, що обчислюється (повертається) функцією користувача, може відноситися до будь-якого типу даних. Наприклад, він може бути не лише чисельним, але і символьним.

2.2 Вказівки до самостійної роботи

Самостійна робота є необхідним елементом підготовки до практичної і лабораторної робіт в комп'ютерному класі. Під час підготовки до виконання роботи вивчити та описати в протоколі лабораторної роботи такі поняття (визначення) і операції (команди):

- поняття функцій користувача та принципи їх організації;
- особливості їх роботи (обміну інформацією) при виклику функції та по завершенні її роботи.
- вправи по створенню програми, що використовує одну або декілька простих (з однієї формули) функцій користувача;
- вправи по створенню програми, що використовує одну або декілька функцій користувача (за певним алгоритмом[12]);
- проаналізувати як відбувається обмін даними при зверненні до функції користувача та при поверненні результату;
- розробити програму (відповідно своєму варіанту завдання – Додаток Д[12]), яка реалізує поставлену задачу у функції (функціях) користувача. Організувати виведення інформації в формі таблиці. Організувати також запис інформації в файл;
- налаштувати програму та перевірити її роботу;
- продемонструвати роботу програми викладачу.

[12] За вказівкою викладача.

2.3 Опис лабораторних засобів та обладнання

Лабораторна робота №2 виконується на персональному комп'ютері стандарту IBM PC під керуванням операційної системи MS Windows.

2.4 Заходи безпеки під час виконання практичної і лабораторної робіт

Заходи безпеки, яких треба дотримуватись при виконанні даної роботи, наведені у додатку А.

2.5 Вказівки до виконання лабораторної роботи №2

1. Відповідно до свого варіанту розробити програму, яка:
   - ✓ реалізує поставлену задачу за допомогою функції(й) користувача[13] – значення *суми*/ *добутку* повинно розраховуватись у функції користувача. Якщо значення чисельника або знаменника доданку/співмножника таке, що не дозволяє його розрахувати[14] (наприклад, воно дорівнює нулю), слід надрукувати відповідне повідомлення та не враховувати цей доданок/співмножник при розрахунку результату;
   - ✓ бажано застосувати окремі функції для розрахунку чисельника та знаменника *доданка/співмножника*;
   - ✓ в залежності від бажання користувача[15] передбачити виведення крім кінцевого результату також проміжних результатів у вигляді таблиці, що включає значення параметру *суми*/*добутку*, *доданків*/*співмножників* та поточних *сум*/*добутків* (див. примітку);
   - ✓ виведення результатів організувати як на лист MS Excel, так і; у вікна функції ***MsgBox*** та "***Immediate***". Організувати виведення інформації в файл[16].
   - ✓ виведення кінцевих результатів повинно відбуватись в головній процедурі;
   - ✓ результати, виведені на лист, слід відформатувати[17].
2. Запрограмувати порівняння значень функції, порахованих при двох різних значеннях параметру ***a***[18].
3. Налаштувати програму та перевірити її роботу на прикладі різних значень границь при обчисленні суми або добутку.
4. Продемонструвати роботу програми викладачу.

---

[13] При наявності в формулі *доданку/співмножника* невідомих констант, слід надати користувачу можливість їх задати (у головній програмі).
[14] Запрограмувати необхідні перевірки.
[15] Використати функцію **MsgBox** з кнопками «*Да*» та «*Нет*» або «*Yes*», «*No*» та «*Cancel*».
[16] При розв'язанні нового варіанту задачі, результати повинні записуватись в той же файл, що і попередні варіанти розрахунків не знищуючи їх.
[17] Для форматування створити окрему процедуру або функцію.
[18] Значення параметру ***a*** задаються користувачем довільно.

5. Оформити протокол лабораторної роботи, до складу якого треба включити теоретичні відомості, індивідуальні завдання опис та результати їх виконання (блок-схему алгоритму, тексти програм (виділити заголовки головної процедури та всіх функцій користувача, що входять до складу програми), роздруківки листа (листів) з таблицями, скріншоти вікон), висновки.

Примітка. Нижче наведено приклад створюваної таблиці:

Приклад таблиці:

Табл. 2. a = 4

i	Dod_i	S_i
-5	0,1667	0,1667
-4	0,0667	0,2333
-3	0,0000	0,2333
-2	0,3333	0,5667
-1	знаменник =0	
0	-9,0000	-8,4333
1	знаменник =0	
2	8,3333	-0,1000
3	4,5000	4,4000
4	3,2667	7,6667

2.6 Обробка та аналіз результатів. Оформлення звіту з лабораторної роботи

Протокол лабораторної роботи оформлюється згідно умов наведених вище (див. лабораторну роботу №1). Необхідно проаналізувати ефективність застосування в програмах функцій користувача та порівняти можливості обох варіантів функцій.

2.7 Контрольні запитання

1. Яку структуру має функція користувача?
2. Що таке формальні та фактичні параметри функцій користувача?
3. Вимоги до списків формальних і фактичних параметрів?
4. Як відбувається передача данних при зверненні до функції і як повертається отриманий результат?
5. Які основні відмінності між процедурами і функціями?
6. Як здійснити достроковий вихід з підпрограми або функції?

3. Організація ітераційних циклів

Мета та основні завдання роботи: дослідити особливості використання операторів організації циклів при реалізації ітераційних та рекурентних розрахунків у складі програмних засобів у мові програмування Visual Basic.

3.1 Основні теоретичні відомості

Ітераційні цикли – це цикли, перед виконанням яких кількість їх повторень невідома (не може бути розрахована). Припинення або повторення визначається в залежності від виконання (або не виконання) певної умови.

Залежно від того, коли робиться перевірка умови повторення циклу – перед тілом циклу або після, розрізняють циклічні алгоритми двох видів (рис. 3.1):

- цикли "з передумовою" (рис. 3.1 – а).
- цикли "з післяумовою" (рис. 3.1 – б);

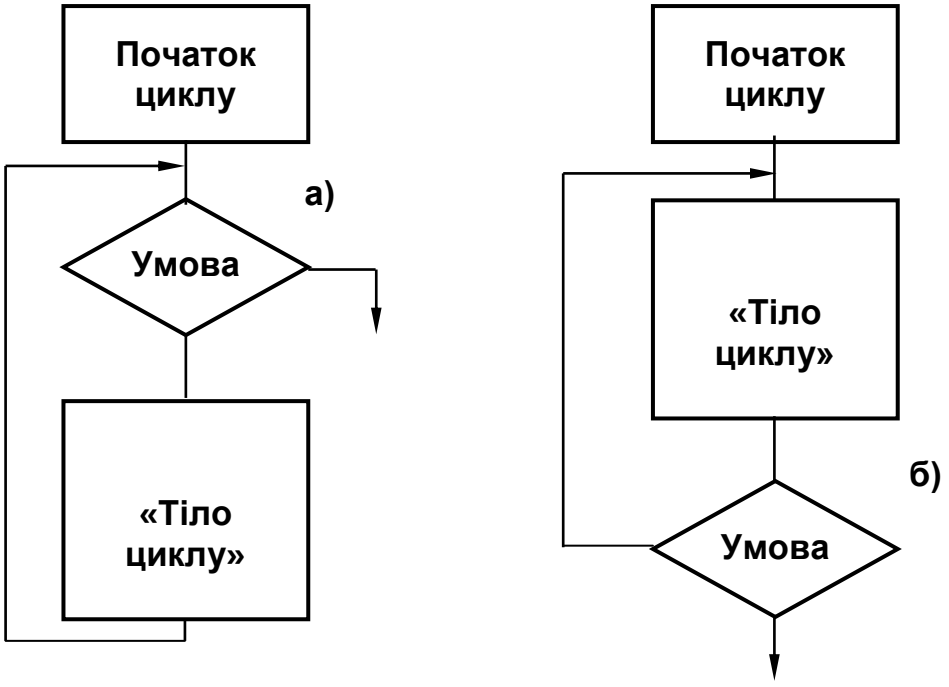


Рис. 3.1 Алгоритми побудови (а) циклу з передумовою і (б) цикл з післяумовою

В циклі «з післяумовою», оператори тіла циклу виконуються, принаймні, один раз, навіть якщо умова повторення циклу не дотримується, оскільки перевірка умови знаходиться у кінці циклу.

Для програмування ітераційних циклів використовують блок операторів **Do...Loop**. Користуючись циклом **Do...Loop** програміст має можливість самому обирати варіант побудови циклу – з передумовою або з післяумовою:

Варіант циклу з передумовою

Do <умова>

Тіло циклу

[Exit Do]

Loop

Варіант циклу з післяумовою

Do

Тіло циклу

[Exit Do]

Loop <умова>

Крім цього, умова, за якою перевіряється необхідність повторення або припинення циклу, може записуватись одним з двох наступних способів:

- **While <логічний вираз>**
- **Until <логічний вираз>**

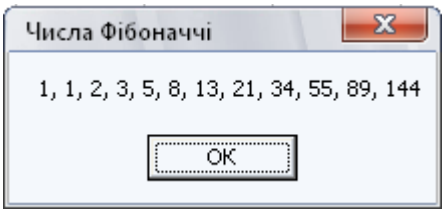
При використанні умови типу **While,** цикл виконується (повторюється) доки логічний вираз в умові приймає значення **True** (умова виконується). Якщо умова не виконується (логічний вираз має значення **False**), цикл припиняється.

При використанні умови типу **Until,** цикл припиняється, коли логічний вираз приймає значення **True** (умова виконується). Якщо логічний вираз має значення **False** (умова не виконується), цикл повторюється. Отже всього існує чотири варіанти запису циклу **Do...Loop**. (*Запишіть всі чотири варіанти*).

Оператор **Exit Do** дозволяє передчасно припинити виконання циклу за допомогою якоїсь додаткової умови.

Приклад 1. Знайти ***n*** чисел Фібоначчі (1, 1, 2, 3, 5, 8, 13, 21,...) користуючись формулою $F_n = F_{n-1} + F_{n-2}$ і виходячи з того, що $F_1 = 1$ та $F_2 = 1$.

```
Public Sub Fibonachi()
   Dim ii As Integer, Nn As Integer, iFib() As Integer
   Dim sMes As String
   Nn = InputBox("n=", "Числа Фібоначчі")
   ReDim iFib(Nn) As Integer
   ii = 3
   iFib(1) = 1
   iFib(2) = 1
   sMes = iFib(1) & ", " & iFib(2)
   Do While ii <= Nn
      iFib(ii) = iFib(ii - 1) + iFib(ii - 2)
      sMes = sMes & ", " & iFib(ii)
      ii = ii + 1
   Loop
   MsgBox sMes, , "Числа Фібоначчі"
   PrintArray Nn, iFib(), "Числа Фібоначчі"
End Sub
```



	A	B	C	D	E	F	G	H	I	J	K	L
1	Числа Фібоначчі											
2	1	1	2	3	5	8	13	21	34	55	89	144

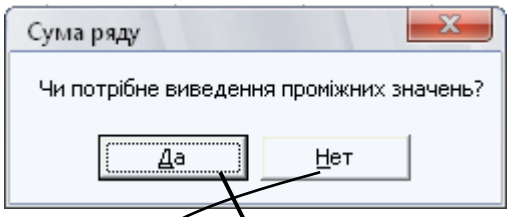
Приклад 2. Знайти із заданою точністю ε суму нескінченного ряду:

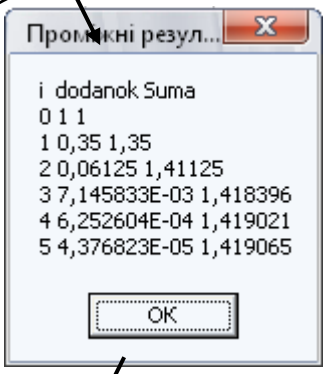
$$S = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots = e^x \qquad |x| < \infty$$

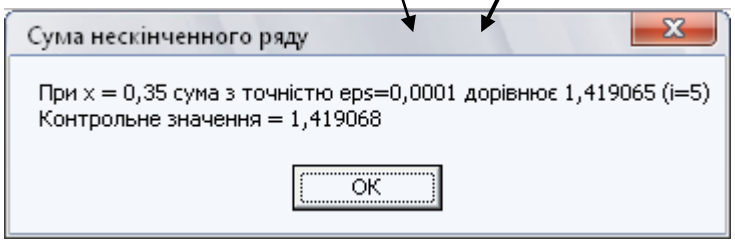
Умовою досягнення необхідної точності є $| S_{k+1} - S_k | \le \varepsilon$, де $S_k = \sum_{i=1}^{k} a_i$ $S_{k+1} = \sum_{i=1}^{k+1} a_i$ або $| a_{i+1} | \le \varepsilon$

Текст програми:

```
Option Explicit

Public Function SumaRjadu(x, eps, ii, iPrInd)
  Dim dod As Single, sMes As String
  SumaRjadu = 1
  ii = 0
  dod = 1
  If iPrInd = vbYes Then
    sMes = "i" & "  dodanok" & " Suma" & vbNewLine
    sMes = sMes & ii & " " & dod & " " & dod & vbNewLine
  End If
  Do
    ii = ii + 1
    dod = dod * x / ii
    SumaRjadu = SumaRjadu + dod
    If iPrInd = vbYes Then
      sMes = sMes & ii & " " & dod & " " & SumaRjadu & vbNewLine
    End If
  Loop Until Abs(dod) <= eps
  If iPrInd = vbYes Then
     MsgBox sMes, , "Проміжні результати"
  End If
End Function

Public Function KontrolValue(x)
   KontrolValue = Exp(x)
End Function

Public Sub Primer2()
  Dim eps As Single, x As Single, Suma As Single, SumContr As Single
  Dim iK As Integer, iPrInd As Byte
  eps = InputBox("Потрібна точність")
  x = InputBox("Значення x?")
  iPrInd = MsgBox("Чи потрібне виведення проміжних значень?", vbYesNo,
                                                    "Сума ряду")
```

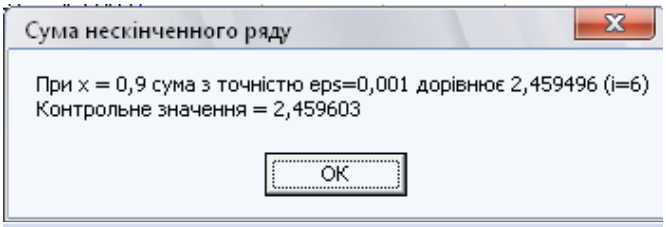


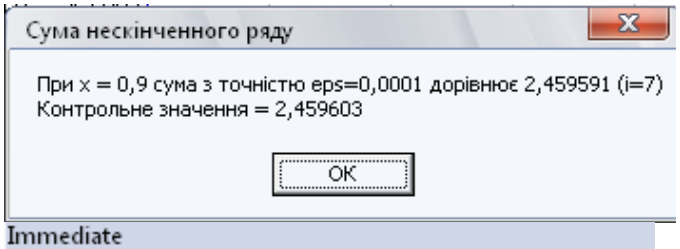




Suma = SumaRjadu(x, eps, iK, iPrInd)
SumContr = KontrolValue(x)

MsgBox "Сума з точністю eps=" & eps & " дорівнює " & Suma & " (i=" & iK & ")" & vbNewLine & "Контрольне значення = " & SumContr, , "Сума нескінченного ряду"
End Sub





Immediate

```
x =  0,9       eps =  0,001
Проміжні результати
  i    Dod                Sum
  1    0,9                1,9
  2    0,405              2,305
  3    0,1215             2,4265
  4    0,0273375          2,453837
  5    4,920749E-03       2,458758
  6    7,381123E-04       2,459496

Сума =  2,459496   Кільк.ітерацій =  6
Контрольне значення =  2,459603
```

Immediate

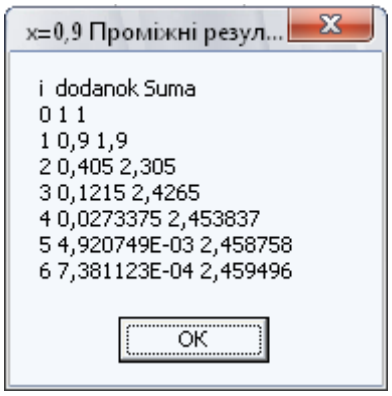
```
x =  0,9       eps =  0,0001
Проміжні результати
  i    Dod                Sum
  1    0,9                1,9
  2    0,405              2,305
  3    0,1215             2,4265
  4    0,0273375          2,453837
  5    4,920749E-03       2,458758
  6    7,381123E-04       2,459496
  7    9,490015E-05       2,459591

Сума =  2,459591   Кільк.ітерацій =  7
Контрольне значення =  2,459603
```

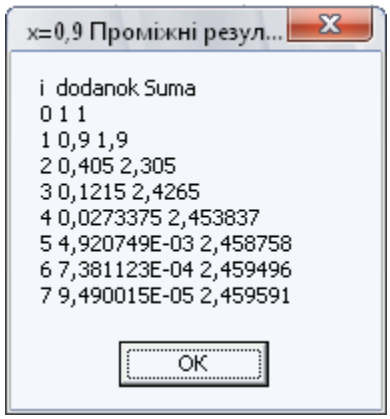
Immediate

```
x = -2,8       eps =  0,001
Проміжні результати
  i    Dod                Sum
  1   -2,8               -1,8
  2    3,92               2,12
  3   -3,658667          -1,538667
  4    2,561067           1,0224
  5   -1,434197          -0,4117974
  6    0,669292           0,2574946
  7   -0,2677168         -0,0102222
  8    9,370089E-02       8,347869E-02
  9   -2,915139E-02       0,0543273
  10   8,162389E-03       6,248969E-02
  11 -2,077699E-03       6,041199E-02
  12   4,847964E-04       6,089679E-02

Сума =  6,089679E-02   Кільк.ітерацій =  12
Контрольне значення =  6,081007E-02
```

Immediate

```
x = -2,8       eps =  0,0001
Проміжні результати
  i    Dod                Sum
  1   -2,8               -1,8
  2    3,92               2,12
  3   -3,658667          -1,538667
  4    2,561067           1,0224
  5   -1,434197          -0,4117974
  6    0,669292           0,2574946
  7   -0,2677168         -0,0102222
  8    9,370089E-02       8,347869E-02
  9   -2,915139E-02       0,0543273
  10   8,162389E-03       6,248969E-02
  11 -2,077699E-03       6,041199E-02
  12   4,847964E-04       6,089679E-02
  13 -1,044177E-04       6,079237E-02
  14   2,088354E-05       6,081326E-02

Сума =  6,081326E-02   Кільк.ітерацій =  14
Контрольне значення =  6,081007E-02
```





Порівняємо отримані результати:

x	Sk	S	iK	eps
0,9	2,459603	2,459496	6	0,001
		2,459591	7	0,0001
-2,8	6,081007E-02	6,089679E-02	12	0,001
		6,081326E-02	14	0,0001

3.2 Вказівки до самостійної роботи

Самостійна робота є необхідним елементом підготовки до практичної і лабораторної робіт в комп'ютерному класі. Під час підготовки до виконання роботи вивчити та описати в протоколі лабораторної роботи такі поняття (визначення) і операції (команди):

- Вивчити поняття ітераційних циклів і особливості операторів, які використовуються для їх реалізації. Самостійно вивчити оператор **While...Wend** та порівняти його з оператором **Do...Loop.**
- Вивчити приклади алгоритмів і програм, наведених в конспекті лекцій. Перевірити та проаналізувати їх роботу.
- Розробити алгоритм та написати програму для обчислення суми наступного ряду:

$$S \approx 1+\frac{5}{2}x+\frac{5\cdot 7}{2\cdot 4}x^2+\frac{5\cdot 7\cdot 9}{2\cdot 4\cdot 6}x^3+\ldots=(1-x)^{-5/2} \quad |x|<1$$

Перевірити роботу написаної програми при x=0,25 та при x=-0,8 для ε=0,001 та ε=0,0001

3.3 Опис лабораторних засобів та обладнання

Лабораторна робота № 6 виконується на персональному комп'ютері стандарту IBM PC під керуванням операційної системи MS Windows.

3.4 Заходи безпеки під час виконання лабораторної роботи

Заходи безпеки, яких треба дотримуватись при виконанні даної лабораторної роботи, наведені у додатку А.

3.5 Вказівки до виконання лабораторної роботи №3

1. Відповідно до свого варіанту завдання створити:
   - функцію, яка розраховує значення нескінченної суми (суми нескінченного ряду) для заданого значення аргументу з заданою точністю ε[19]. Використати ітераційний цикл **Do…Loop**. Передбачити визначення кількості необхідних ітерацій та точності отриманого результату, а також можливість (в залежності від бажання користувача[20]) виведення таблиці поточних значень – номера ітерації, доданку та суми (на кожній ітерації)[21];
   - функцію для визначення «*контрольного значення*» суми у відповідності до заданої формули;
   - підпрограму (головну), яка звертається до вищезгаданих функцій кілька разів з різними значеннями аргументу x та точності ε.
2. Налаштувати розроблену програму на комп'ютері.
3. Перевірити роботу програми з різними значеннями аргументу x (додатних та від'ємних, якщо $x \in [0, 1]$, то $|x| \to 0$ та $|x| \to 1$) та точності ε (наприклад, – 0.001 та 0.0001).
4. Проаналізувати отримані результати (процес збіжності ряду в залежності від значення x та від заданої точності ε).
5. Проаналізувати різницю у застосуванні функцій із статусом ***Public*** та функцій із статусом ***Private***.
6. Продемонструвати роботу програми та отримані результати викладачу.
7. Оформити протокол виконання роботи.

3.6 Обробка та аналіз результатів. Оформлення звіту

Протокол лабораторної роботи оформлюється згідно умов наведених вище (див. лабораторні роботи №1 та №2). Необхідно проаналізувати алгоритм роботи ітераційного циклу та закономірності накопичення нескінченної суми в залежності від значення x та заданої точності.

---

[19] Умова досягнення необхідної точності має вигляд:

$| S_{k+1} - S_k | \le \varepsilon$, де $S_k = \sum_{i=1}^{k} a_i$ $S_{k+1} = \sum_{i=1}^{k+1} a_i$

[20] Використати функцію **MsgBox** з кнопками «*Да*» та «*Нет*» або «*Yes*», «*No*» та «*Cancel*».

[21] Спосіб (способи) виведення результатів (на лист MS Excel, у вікно функції ***MsgBox*** або у вікно "***Immediate***", або запис у файл)студент обирає самостійно.

3.7 Контрольні запитання

1. Як програмуються циклічні алгоритми з невизначеним числом повторень циклу?
2. Чим відрізняються цикли з перед- та післяумовою?
3. Як працює оператор циклу **Do ... Loop** з умовою типу **While**?
4. Як працює оператор циклу **Do ... Loop** з умовою типу **Until**?
5. У якій частині циклу можна перевіряти умову його закінчення?
6. Чи може тіло оператора циклу з передумовою не виконатися жодного разу?
7. Чи може тіло оператора циклу з умовою виконуватися нескінченне число разів?
8. Як записується і як працює оператор циклу **While ... Wend**?
9. У яких випадках доцільно використати оператор **Do … Loop** з постумовою **While**?
10. Як здійснити достроковий вихід з циклу?

4. Обробка символьних даних

Мета та основні завдання роботи: дослідити особливості оперування з символьними змінними при створенні програмних засобів у мові програмування Visual Basic.

4.1 Основні теоретичні відомості

4.1.1. Дані символьного типу, їх введення та виведення

Символьною (*строковою*) *константою* є будь-яка послідовність символів, взятих в лапки. Наприклад:

"x=" **"Введіть значення a та b -"**
"alfa =" **"Таблиця залежності y=f(x)"**
"Іванов П.К." **"a:\work\sort.dat"**

Символьна змінна описується за допомогою оператора **DIM**. У VBA представлено два типи символьних змінних (рядків):

- Рядки фіксованої довжини, що оголошуються з певною кількістю символів. Наприклад:

  Dim sStringFixed As String * 100, MyName As String * 25
  DIM sFile1 As String * 50, sWrd As String * 20

  Максимальна довжина рядка в цьому випадку становить **65535** символів.
- Рядки змінної довжини, які теоретично можуть вмішати до 2 млрд. символів. Наприклад:

  Dim sStringDynamic As String, FileName As String, Task As String

При оголошенні рядка за допомогою оператора **Dim** можна визначити її довжину, якщо вона відома (задати тип рядка з фіксованою довжиною), або активізувати динамічну обробку довжини рядка (задати тип рядка змінної довжини).

Примітка. В старих версіях мови також використовувалось позначення типів змінних певним «*суфіксом*», яким закінчувалось ім'я змінної. Імена *символьних змінних* відрізняються від імен змінних інших типів за «*суфіксом*» **$**. При цьому оператор **Dim** не використовувався. Наприклад:

name$** **file1$ **mounse$** **d2$** тощо.

☞ Способи введення символьних даних:

А) за допомогою функції або методу InputBox:

InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])

<object>.InputBox(Prompt, Title, Default, Left, Top, HelpFile, HelpContextId, Type)

Наприклад:

```
sName = InputBox ("Ваше прізвище?")
sFile1 = InputBox ("Вкажіть і'мя файлу вихідних даних")
```

Б) за допомогою оператора Input:

```
Input #<номер файлу>, <список змінних>
```

Наприклад:

```
Dim MyString, MyNumber
Open "TESTFILE" For Input As #1    ' Відкриття файлу для читання
Do While Not EOF(1)                ' Доки не знайдений кінець файлу
   Input #1, MyString, MyNumber
                                   ' Читання даних у дві змінні ("Hello", 234).
   Debug.Print MyString, MyNumber  'Виведення даних у вікно "Immediate"
Loop
Close #1                           ' Закриття файлу
...
```

В) за допомогою оператора Line Input (введення цілого рядка):

```
Line Input #<номер файлу>, <ім'я змінної>
```

Наприклад:

```
Dim TextLine As String
Open "TESTFILE" For Input As #1   ' Відкриття файлу для читання
Do While Not EOF(1)               ' Доки не знайдений кінець файлу
   Line Input #1, TextLine        ' Читання рядка у змінну
   Debug.Print TextLine           'Виведення даних у вікно "Immediate"
Loop
Close #1                          ' Закриття файлу
...
```

Г) за допомогою функції Input:

```
Input(number, [#]filenumber)
```

або

```
sText1 = Input (<Кількість> [, <Номер_файлу>])
```

Наприклад:

```
Dim MyText As String
Open "TESTFILE" For Input As #1   ' Відкриття файлу для читання
...
MyText = Input (14, #1) ' Читання 14 символів з відкритого файлу у змінну
...
Debug.Print MyText                'Виведення даних у вікно "Immediate"
...
Close #1                          ' Закриття файлу
...
```

☞Друк символьних даних:

Виведення символьних даних можливо у вікно ***MsgBox***, або у вікно ***Immediate***, або на лист MS Excel. Принципово воно нічим не відрізняється від виведення даних інших типів. Приклади друкуваня символьних даних у вікно "***Immediate***":

```
Debug.Print "Лабораторна робота N4"
Debug.Print "Студент Петренко, гр. ХА-11"
Debug.Print "x="; x, "y="; y
Debug.Print sName1, sName2
Debug.Print a$[22], b$, c$
Debug.Print a$+b$
```

Друк у файл:

```
Print #3, sName1, sName2
Write #3, sName1, sName2
```

4.1.2. Основні операції з даними символьного типу

☞«Зчеплення» даних («конкатенація»)

Над строковимі змінними можна виконувати операцію додавання, точніше – *зчеплення* (операцію *конкатенації*) – обєднання декількох рядків (або констант, або змінних, або результатів, повернутих символьною функцією) в один рядок. Наприклад:

```
sText1 = "Студент Козаченко"
sText2 = " написав гарну програму"
sText = sText1 + sText2
```

В результаті змінна **sText** набуде значення:

```
"Студент Козаченко написав гарну програму"
```

☞Порівняння символьних даних (порівняння рядків)

При порівнянні між собою значень символьних змінних (за допомогою звичайних операцій відношення – **>**, **<**, ...), наприклад:

```
IF a$ > b$ Then
```

відбувається послідовне порівняння числових кодів символів, які є значеннями цих змінних. Тобто, два "рядки" порівнюються посимвольно до появи двох перших неспівпадаючих символів. Потім порівнюються коди цих неспівпадаючих символів (як звичайні числові значення) і визначається найбільший код. Відповідний рядок вважається більшим.

Примітка. Інтерпретація цих операцій залежить від установки опції **Option Compare**.

Директиви Option Compare Binary і Option Compare Text

За допомогою директиви **Option Compare Binary** можна задати режим бінарного порівняння операндів, тобто порівняння на основі збігу числових значень байтів. У цьому випадку порівняння чутливе до регістру.

[22] Використані «старі» позначення символьних змінних, щоб вказати ни їх тип.

Режим текстового порівняння здасться за допомогою директиви **Option Compare Text**. При цьому відмінність між великими та малими літерами не береться до уваги.

Наприклад:

"сло**в**арь" **<** "сло**н**" оскільки "**в**" (**162**) **<** "**н**" (**173**)

Якщо перший рядок коротший за другий, і усі його символи співпадають з відповідними символами другого рядка, то більшим вважається другий (довший) рядок:

"PROGR" < "PROGRAM"

Отже дві символьні змінні тільки тоді будуть дорівнювати одна одній, коли вони повністю ідентичні. Наприклад, в режимі **Option Compare Binary** значення тільки значення "**YES**" і "**YES**" дорівнюють одне одному, а значення "**YES**", "**YEs**", "**Yes**" і "**yes**" не дорівнюють одне одному. Це пояснюється тим, що коди маленьких і великих літер різні. А в режимі **Option Compare Text** всі ці значення дорівнюють одне одному.

Тому, якщо в програмі використовується діалоговий запит з відповіддю в символьній формі, необхідно враховувати можливість відповіді як прописними, так і рядковими буквами. Наприклад, такий фрагмент програми може мати вигляд:

```
.....
sAns = "y"
Do While sAns = "Y" Or sAns = "y"
.....
      sAns = InputBox("Наступний варіант? (Y/N))";
Loop
.....
```

Цей цикл повторюватиметься доки символьна змінна **sAns** має значення "**Y**" або "**y**". При введенні будь-якого іншого значення цикл припиняється.

Якщо потрібно локально перевизначити режим порівняння, заданий опцією для всього модуля, то можна використовувати вбудовану функцію **StrComp**, яка повертає результат порівняння рядків. Її синтаксис:

StrComp (string1, string2 [, compare])

Аргументи **string1** і **string2** – порівнювані рядки. Необов'язковий аргумент **compare** вказує спосіб порівняння рядків. Його значення за умовчанням **0** використовується для виконання двійкового порівняння; **1** задає посимвольного порівняння без урахування регістру. Якщо **string1** менше, ніж **string2**, то результат дорівнює **1**, якщо рядки рівні, то – **0**, якщо другий менше, то **–1**, якщо хоч один з рядків має значення **Null**, то результат також дорівнює **Null**.

Для попереднього прикладу:

Do While StrComp(sAns, "Y", 1) =0

Приклади результатів застосування функції **StrComp**:

Dim MyResult As Integer

Варіант порівняння	Змінна MyResult
MyResult = StrConp("Іванов", "іванов", vbTextCompare)	**0**
MyResult = StrConp("Іванов", "іванов", vbBinaryCompare)	**-1**
MyResult = StrConp("іванов", "Іванов", vbBinaryCompare)	**1**
MyResult = StrConp("іванов", "іванов", vbBinaryCompare)	**0**

4.1.3. Стандартні функції для обробки символьних даних.

Имена функций (стандартных, внутренних, внешних), результат вычисления которых относится к символьному типу, заканчиваются "суффиксом" $.

☞ Отримання символів коду ASCII (функція **Chr()**) :

```
nk = 78
Debug.Print "Цей код відповідає символу - "; Chr(nk)
```

Результат: **Цей код відповідає символу – N**

☞ Отримання ASCII-коду заданого символу (функція **Asc()**):

```
Str = InputBox("Задайте ,будь-який символ")
Debug.Print "Цей символ має код - "; Asc(Str)
```

```
Debug.Print Asc("f") → 102        Debug.Print Asc("file") → 102
Debug.Print Asc("Я") → 223        Debug.Print Asc("я") → 255
```

☞ Визначення довжини символьного рядка (функція **Len()**):

```
<Кількість_символів%>← Len(<Рядок>)
```

Наприклад:

```
Str = "Задача"
Debug.Print "В рядку "; Str; "-"; Len(Str); "символів"
```

Результат: **В рядку Задача - 6 символів**

```
sText1 = "Студент Палій"
sText2 = " написав гарну програму"
k = Len(sText1 + sText2)
Debug.Print "k="; k
```

Результат: **k=36**

☞ Видалення пробілів (функції **Ltrim()**, **Rtrim()** та **Trim()**):

```
<Рядок> ← Ltrim(<Рядок>)
<Рядок> ← Rtrim(<Рядок>)
<Рядок> ← Trim(<Рядок>)
```

Функції повертають заданий рядок відповідно без лівих, правих або і тих і тих пробілів одночасно.

☞Виділення підрядків.

Наступні функції призначені для виділення частини вихідного рядка (*підрядка*):

а) від лівого краю заданого рядка (функція **Left()**):

Left(<Рядок>, <Довжина>)

б) від правого краю заданого рядка (функція Rigth()):

Rigth(<Рядок>,<Довжина>)

в) з середини заданого рядка (функція Mid()):

Mid(<Рядок>, <Початок> [, <Довжина>])

де:

<Рядок> – вихідний рядок;
<Початок> – номер символу, з якого починається виділення підрядка;
<Довжина> – довжина підрядка (кількість символів).

Приклад:

```
sText = " Ми розробили гарну програму"
sText1 = Left(sText, 3)
          'Виділення 3-х перших символів                    _Ми
sText2 = Mid(sText, 4, 10)
          'Виділення 10-ти символів, починая з 4-го      _розробили
sText3 = Right(sText, 9)
          'Виділення 9-ти останніх символів                 _програму
Debug.Print "Яку"; sText3 + sText1 + sText2; "?"
```

Результат: **Яку програму Ми розробили?**

Заміна внутрішнього підрядка новим значенням (іншим підрядком) за допомогою оператора **Mid**:

Mid(<Рядок>, <Початок> [, <Довжина>])= <новий_підрядок>

Приклад:

```
sText = " Ми розробили гарну програму"
Mid(sText, 15, 6) = "погану"
Debug.Print sText
```

Результат: **Ми розробили поганупрограму**[23]

☞Створення рядка пробілів (функція **Space()**):

fsr = Space(<Довжина>)

[23] В звязку з тим, що відбувається заміна 6 символів, крім слова «**гарну**» було також видалено пробіл після нього.

Приклад:

Результат:

```
For i=1 To 5
      Debug.Print Space(i); i
Next i
```

```
 1
  2
   3
    4
     5
```

☞ Створення рядка з однакових символів (функція **String()**):

String(<Довжина>, <ASCII-код>)

або

String(<Довжина>, <Рядок>)

Приклад:

Debug.Print String(25, 42) або **Debug.Print String(25, "*")**

Результат: **********

☞ Перевірка збігу рядків (функція **Instr()**):

Instr([<Початок>,]<Рядок1>, <Рядок2>)

Функція дозволяє визначити чи входить підрядок <**Рядок2**> в рядок <**Рядок1**>. Якщо процедура пошуку завершена успішно (у тексті <**Рядок1**> знайдено фрагмент – <**Рядок2**>), то функція **Instr** повертає значення рівне номеру позиції, починаючи з якої виявлено збіг рядків. Якщо аналіз входження призводить до негативного результату, то функція **Instr** повертає **нуль**.

Приклад:

```
sText = "Програма написана мовою VisualBasic"
sText1 = "VisualBasic"
nk = InStr(sText, sText1)
Debug.Print "У вихідному рядку є слово "; sText1
Debug.Print "Воно починається з"; nk; "-ї позиції"
```

Результат:

```
У вихідному рядку є слово VisualBasic
Воно починається з 25 -ї позиції
```

☞ Приклад програми обробки символьних даних.

```
Public Sub Symb_Test3()
'Програма-шифратор, яка записує будь-який заданий текст у зворотному порядку
   Dim b(100) As String, a As String, k As String, m As String
   m = "Y"
   Do While m = "Y"
     Line Input #1, a
```

```
        n = Len(a)
        For i = 1 To n
           b(i) = Mid(a, i, 1)
        Next i
        k = b(n)
        For i = n - 1 To 1 Step -1
           k = k & b$(i)
        Next i
        Debug.Print "Вихідний текст:"
        Debug.Print a
        Debug.Print
        Debug.Print "Текст навпаки:"
        Debug.Print k
        Debug.Print
        m = InputBox("Продовжити роботу? (Y/N) ")
    Loop
End Sub
```

Результат:

Вихідний текст:
Завдання. Для довільної матриці A(n, n) обчислити вектор S.

Текст у зворотньому порядку:
.S роткев итилсичбо)n ,n(A іциртам їоньлівод ялД .яннадваЗ

4.2 Вказівки до самостійної роботи

Самостійна робота є необхідним елементом підготовки до практичної і лабораторної робіт в комп'ютерному класі. Під час підготовки до виконання роботи вивчити та описати в протоколі лабораторної роботи поняття даних символьного типу, особливості виконання операцій над ними, функції, які використовуються для їх обробки.

- Вивчити поняття даних символьного типу, особливості виконання операцій над ними, функції, які використовуються для їх обробки.
- Вивчити приклади обробки символьних даних і характерні прийоми, наведені в конспекті лекцій. Перевірити та проаналізувати їх роботу.
- Написати та налаштувати програми обробки символьних даних за наступними завданнями (Додаток Е)[24]. Перевірити різні способи введення та виведення символьних даних (в тому числі – запис у текстовий файл[25]).

[24] Виконати не менше ніж одне завдання з кожного підрозділу. Завдання, які помічені (*) – є обов'язковими.

[25] При розв'язанні нового варіанту задачі, результати повинні записуватись в той самий файл, що і попередні варіанти розрахунків не знищуючи їх.

– Перевірити роботу програм на 2х – 3х прикладах.
– Продемонструвати роботу програм викладачу.
– Тексти та результати роботи розроблених програм включити до протоколу лабораторної роботи разом з теоретичними відомостями.

4.3 Опис лабораторних засобів та обладнання

Лабораторна робота №10 виконується на персональному комп'ютері стандарту IBM PC під керуванням операційної системи MS Windows.

4.4 Заходи безпеки під час виконання лабораторної роботи

Заходи безпеки, яких треба дотримуватись при виконанні даної лабораторної роботи, наведені у додатку А.

4.5 Вказівки до виконання лабораторної роботи №4

1. Написати програму обробки символьного списку даних у відповідності до свого варіанту завдання. Для введення даних використати оператор **Line Input**. Для введення даних використати заздалегідь підготований файл(и). Передбачити виведення інформації в файл.
2. Налаштувати написану програму на комп'ютері та перевірити її роботу на 2х – 3х прикладах[26].
3. Продемонструвати роботу програми викладачу.
4. Оформить протокол лабораторної роботи.

4.6 Обробка та аналіз результатів. Оформлення звіту

Протокол лабораторної роботи оформлюється згідно умов наведених вище (див. лабораторні роботи №1 та №2). Необхідно проаналізувати характерні прийоми, які використовуються при обробці символьних даних, та особливості застосування відповідних функцій.

4.7 Контрольні запитання

1. Як оголошуються строкові змінні?
2. Яким чином відбувається об'єднуються рядків?
3. Як відбувається порівняння рядків?
4. Як визначити довжину рядка?
5. Які дії виконують функції **Left** і **Right**?
6. Яке максимальне число символів може містити строкова змінна?

[26] При розв'язанні нового варіанту задачі, результати повинні записуватись в той самий файл, що і попередні варіанти розрахунків не знищуючи їх.

7. За допомогою якої функції можна визначити номер коду будь-якого символу та з допомогою якої функції можна за кодом визначити символ?
8. Які функції дозволяють виділити частину рядка, відповідно, від лівого та від правого краю?
9. Як працює функція **Mid** та оператор з таким же ім'ям?
10. У якому випадку текстову змінну не можна перетворити в число?
11. Як працює оператор **Line Input**?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Основна література

1. Кашаев С.М. Офисные решения с использование MS Excel 2007 и VBA. [Текст] / СПб.: Питер, 2009. – 352 с.
2. Уокенбах Дж. Профессиональное программирование на VBA в Excel 2002. [Текст] / М.: Изд. дом '"Вильямс", 2003. – 764 с.

Додаткова література

3. Эйткен П. Разработка приложений на VBA в среде office XP. [Текст] / – М: "Вильямс", 2003. – 496 с.
4. Эйткен П. Интенсивный курс программирования в Excel за выходные. [Текст] / М.: Изд. дом «Вильямс», 2006. 432 стр.
5. Сергеев А.П. Использование MS Office Excel 2007. - М.: ООО "Изд. Дом Вильямс", 2007. 288 с.
6. Кузьменко В.Г. VBA 2000. Самоучитель. [Текст] / – М.: Бином, 2000. – 408 с.
7. Гарнаев А. Самоучитель VBA. Технология создания пользовательских приложений. [Текст] / – СПб.: BHV, 1999. 512 с.
8. Глушаков С.В., Сурядный А.С., Мачула О.В. MS Excel XP для профессионала. - Харьков, Фолио, 2007. 319 с.
9. Коттингхэм М. Excel 2000. Руководство разработчика. [Текст] / – К.: BHV, “Ирина”, 2000. – 704 с.
10. Каммингс С. VBA для чайников. [Текст] / 2-е изд. – М.: Вильямс, 2000. – 384 с.
11. Биллиг В.А. VBA в Office 2000. Офисное программирование. [Текст] / – М.: Издательс-ко-торговый дом «Русская Редакция», 1999. — 480 с.: ил.
12. Биллиг В.А. VBA и Office 97. Офисное программирование. [Текст] / В.А. Биллиг, М.И. Дехтярь – М.: «Русская Редакция» ТОО «Channel Trading Ltd.», 1998. – 720 с.
13. Харрис М. Освой самостоятельно программирование MS Excel 2000 за 21 день. – М.: Вильямс (SAMS), 2000. – 880 с.
14. Кергаль И. Методы программирования на Бейсике (с упражнениями). – М.: Мир, 1991. – 288 с.
15. Безносик Ю.А. Программирование на языке Qbasic для персональной ЭВМ [Текст] / Ю.А. Безносик, С.Г. Бондаренко, А.А. Квитка. – К.: ИСИО, 1996. – 204 с.
16. Бобровский С.И.. Программирование на языке Qbasic для школьников и студентов [Текст] / М.: ДЕСС КОМ, 2000. – 208 с.

Додаток А.

ЗАХОДИ БЕЗПЕКИ ПІД ЧАС ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

Цикл лабораторних робіт виконуються в комп'ютерних класах кафедри кібернетики хіміко-технологічних процесів хіміко-технологічного факультету та КБ ІС. Обладнання живиться електричним струмом напругою 220 В. Тому при виконанні лабораторних робіт слід дотримуватися заходів безпеки наступних інструкцій.

ІНСТРУКЦІЯ

з техніки безпеки при навчанні студентів на ПЕОМ в учбових лабораторіях кафедри кібернетики хіміко-технологічних процесів інженерно-хімічного факультету

Знання і суворе дотримання цих правил є обов'язковим для всіх осіб, допущених до роботи на ПЕОМ. Доведення їх до кожного зі студентів підтверджується особистим підписом кожного з них у контрольному листі з техніки безпеки. Особи, які не одержали такого інструктажу та не поставили підпис у контрольному листі з техніки безпеки, до роботи на ПЕОМ не допускаються.

Всі роботи в учбових лабораторіях кафедри кібернетики ХТП проводяться лише з дозволу викладача або співробітника кафедри.

Під час проведення занять в учбовій лабораторії не повинні знаходитися сторонні особи, в тому числі студенти інших груп. Студенти не повинні самовільно залишати учбову лабораторію під час занять.

При роботі на ПЕОМ треба пам'ятати, що в них використовується напруга, небезпечна для життя.

Всі особи, працюючі в учбових лабораторіях кафедри КХТП повинні бути ознайомлені з правилами надання першої медичної допомоги при ураженні електричним струмом.

Перед вмиканням ПЕОМ кожен з працюючих повинен отримати дозвіл викладача або співробітника кафедри.

У випадках виникнення короткого замикання, горіння, диму, вогню в апаратурі, пристрій необхідно негайно вимкнути з мережі та доповісти викладачеві або співробітникові кафедри. Самостійні дії по усуненню пошкодження забороняються.

У випадку виходу з ладу обладнання або програмного забезпечення, що зумовлені іншими причинами, доповісти викладачеві або співробітникові кафедри. Вимикати апаратуру при цьому не дозволяється. Самостійні дії по усуненню пошкодження забороняються.

Працюючі в учбових лабораторіях кафедри кібернетики ХТП несуть майнову та адміністративну відповідальність за збереження та використання обладнання, наданого для їх праці.

Категорично забороняється:

- самостійно вмикати та вимикати тумблери на щитку електроживлення;
- несанкціоновано вмикати електрообладнання;
- приносити та вмикати своє обладнання та пристрої, встановлювати власне програмне забезпечення;
- залишати без нагляду увімкнені пристрої та лабораторію;
- пересувати обладнання та комплектуючі;
- підключати та відключати інформаційні кабелі та кабелі живлення;
- використовувати власні носії інформації без дозволу викладачів або співробітників кафедри;
- знаходитись у учбовій лабораторії у верхньому одязі.

Після закінчення занять обладнання не вимикається. Робоче місце має бути прибране працюючим та перевірене викладачем чи співробітником кафедри

ІНСТРУКЦІЯ

про міри пожежної безпеки у лабораторіях, учбових та робочих приміщеннях кафедри кібернетики хіміко-технологічних процесів хіміко-технологічного факультету

Всі студенти повинні знати та ретельно виконувати «Загальні правила пожежної безпеки в НТУУ «КПІ».

Завідуючий кафедрою та завідуючий лабораторією відповідають за забезпечення пожежної безпеки всіх приміщень кафедри та за справність протипожежного обладнання та сигналізації.

Все електричне обладнання, яке знаходиться в лабораторіях та приміщеннях кафедри, повинно мати заземлення.

В усіх приміщеннях повинно дотримуватись чистоти, не займати приміщення непотрібними меблями, обладнанням та матеріалами.

Всі двері основних та додаткових виходів утримувати у стані швидкого відкривання.

Зберігання та використання горючих та легкоспалахуючих рідин у приміщеннях кафедри забороняється.

Ремонт електричного обладнання проводити у строгій відповідності з правилами пожежної безпеки

Всі електрозахисти повинні знаходитися у закритому положенні, не займаними сторонніми предметами.

Коридори, проходи, тамбури, евакуаційні виходи та підходи до першочергових засобів пожежогасіння, а також комунікаційні ніші повинні бути постійно вільними, чистими та нічим не займаними.

Відповідальні особи перед закриттям приміщень повинні ретельно оглянути їх, забезпечити прибирання виробничих відходів, перевірити якість перекриття води, газу, відключити напругу електромережі, перевірити стан пожежної сигналізації та засобів пожежогасіння.

Від усіх приміщень мати два комплекти ключів. Один комплект здавати черговому, а інший – зберігати в певному місці, яке відомо обслуговуючому персоналу.

Студенти повинні знати та ретельно виконувати «Загальні правила техніки безпеки в НТУУ «КПІ», про що вони ставлять свій підпис у відповідному контрольному листі з техніки безпеки перед початком проведення циклу лабораторних робіт. Студенти, які не пройшли інструктаж і не поставили підпис у контрольному листі , до роботи не допускаються.

Додаток Б.

Зразок титульного листа.

Міністерство освіти і науки України

Національний технічний університет України

"Київський політехнічний інститут"

Кафедра кібернетики хіміко-технологічних процесів

Звіт

з лабораторних робіт

Дисципліна: «*Комп'ютерні технології та програмування*»

Виконав: студент 1-го курсу ХТФ
гр. ХА-31
Сергієнко П.М.
Керівник: доц. Квітка О.О.

Київ 2014 р.

Додаток В.

Блок-схеми.

Блоки, що відповідають усім операціям, які використовуються в програмах, мають стандартизовані форму і розміри. Блоки, які використовуються найчастіше наведені в таблиці 1. Блоки розташовуються у блок-схемі згідно з логікою алгоритму. Блок-схема розпочинається з блоку із словом «*Початок*» і закінчується блоком «*Кінець*» (див. табл. 1 п.6). Усі блоки послідовно нумеруються. Номер проставляється в розриві лінії в лівому верхньому кутку кожного блоку (окрім блоків «*Початок*» і «*Кінець*»).

Блоки з'єднуються між собою горизонтальними і/або вертикальними лініями (лінії, розташовані під кутом, не допускаються) із стрілками на кінці, що відображають напрям обчислювального процесу. При цьому стрілки, спрямовані "вниз" і "праворуч" можуть бути опущені. Перетину стрілок бажано не допускати. При неможливості провести нерозривну стрілку між двома блоками, використовуються або "*з'єднувач*", – якщо блоки розташовані на одній сторінці, або "*міжсторінковий з'єднувач*" – якщо вони розташовані на різних сторінках (див. табл. 1, відповідно п.7 і п.8). При використанні "*з'єднувача*" на одному кінці розриву в "колі" проставляється номер наступного блоку, а на іншому – номер попереднього блоку. Коли необхідно з'єднати стрілкою блоки на різних сторінках у "*міжсторінковому з'єднувачі*" аналогічним чином проставляються не лише номери *подальшого* і *попереднього* блоків, але і номери сторінок, на яких вони розташовані (рис. 1).

У кожному "*розрахунковому блоці*" (див. табл. 1 п.1) допускається написання не більше однієї розрахункової формули, але можливе "об'єднання" декількох формул в одному блоці під загальною назвою (рис. 2), якщо вони об'єднані загальним змістом або однаковим призначенням.

Алгоритм, що реалізовується у вигляді окремого програмного модуля, є автономним і зображується у вигляді окремої блок-схеми (окремого малюнка). Зв'язок між різними модулями однієї програми і, відповідно, – між блок-схемами алгоритмів які в них реалізовані, здійснюється за допомогою "*блоку підпрограм*" (див. табл. 1 п.5), який відповідає операторові звернення до підпрограми в "*викликаючому модулі*".

Алгоритм наводиться у вигляді графічних фігур (блоків), кожний з яких має певне смислове значення і є частиною рішення задачі.

Розглянемо основні блоки:

b=1,5a

a

b

1. обчислювальний блок (виконання операції або групи операцій).

a

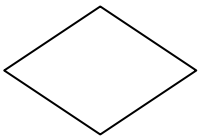
b = 0.25 a

2. блок введення-виведення

a

b

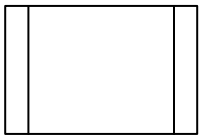
3. виведення на паперовий носій



a

4. логічний блок (вибір напряму виконання алгоритму залежно від деяких умов

b



a

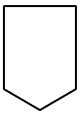
5. “підпрограма”

b

d= 0.5a

6. з'єднувач (зв'язок між перерваними лініями в межах однієї сторінки.

0,5a

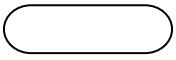


0.6a

7. міжсторінковий з'єднувач

--------[a

8. коментар (зв'язок між елементом схеми і поясненням).

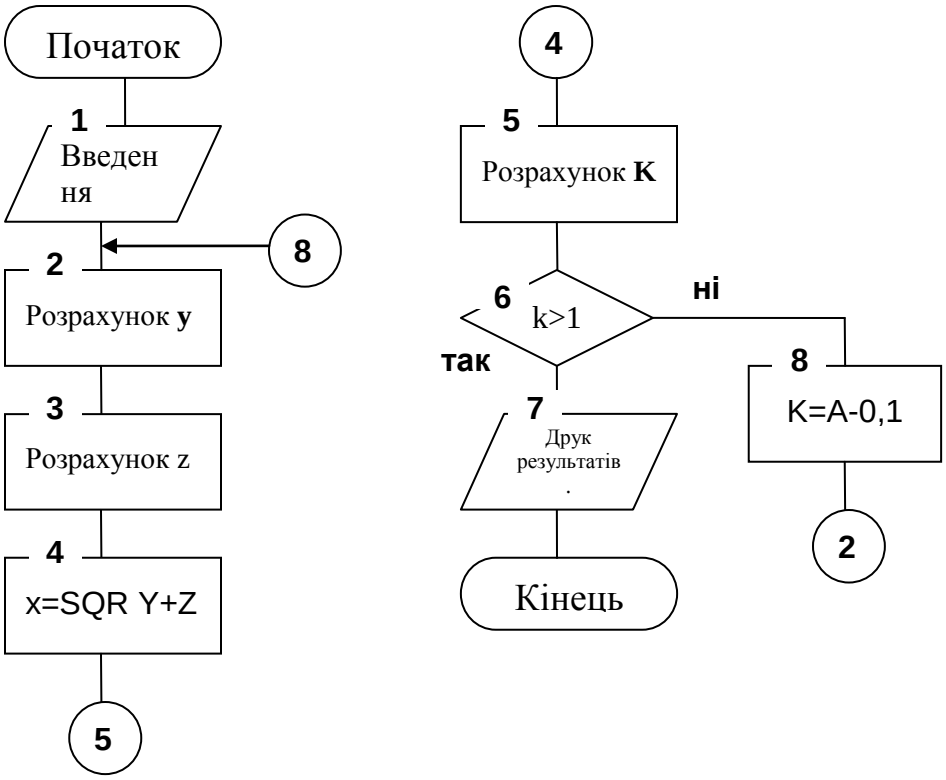


0.5a

9. початок, кінець

b

Приклад.


Початок
1
Введен
ня
8
2
Розрахунок y
3
Розрахунок z
4
x=SQR Y+Z
5
4
5
Розрахунок K
6
k>1
ні
так
8
K=A-0,1
7
Друк
результатів
.
2
Кінець

Додаток Г.

Завдання для самостійної роботи по темі «Програми складної структури»

Розробити програму для розв'язання задачі у відповідності до свого варіанту завдання. Поставлену задачу необхідно реалізувати в окремій процедурі. Передбачити виведення інформації в файл. Налаштувати програму та перевірити її роботу на 2-х – 3-х прикладах[27].

Варіанти індивідуальних завдань

1. Визначити та порівняти мінімальні за модулем значення елементів двох довільних матриць різної розмірності.
2. Для двох довільних матриць різної розмірності визначити та порівняти суми від'ємних непарних елементів.
3. Визначити та порівняти добутки елементів розташованих на побічних діагоналях двох довільних квадратних матриць різної розмірності.
4. Визначити та порівняти максимальні від'ємні елементи в двох довільних матрицях різної розмірності.
5. Визначити та порівняти суми елементів, кратних двом, розташованих на головних діагоналях двох довільних квадратних матриць різної розмірності.
6. Визначити та порівняти суми елементів, кратних трьом, що розташовані в непарних рядках двох довільних матриць різної розмірності.
7. Визначити та порівняти максимальні за модулем значення елементів двох довільних матриць різної розмірності.
8. Визначити та порівняти середні значення від'ємних елементів, кратних двом, розташованих на побічних діагоналях двох довільних квадратних матриць різної розмірності.
9. Визначити та порівняти максимальні парні елементи в двох довільних матрицях різної розмірності.
10. Визначити та порівняти мінімальні значення непарних елементів, що розташовані на головних діагоналях, для двох довільних квадратних матриць різної розмірності.
11. Визначити та порівняти середні значення від'ємних елементів двох довільних матриць різної розмірності.
12. Визначити та порівняти мінімальні додатні елементи в двох довільних матрицях різної розмірності.
13. Визначити та порівняти добутки елементів, кратних п'яти, що розташовані в парних стовпцях двох довільних матриць різної розмірності.

[27] При розв'язанні нового варіанту задачі, результати повинні записуватись в той самий файл, що і попередні варіанти розрахунків не знищуючи їх.

14. Визначити та порівняти мінімальні за модулем значення елементів, розташованих на головних діагоналях двох довільних матриць різної розмірності.
15. Для двох довільних матриць різної розмірності визначити та порівняти суми від'ємних непарних елементів, які розташовані на побічних діагоналях.
16. Визначити та порівняти максимальні від'ємні елементи розташовані на головних діагоналях в двох довільних матрицях різної розмірності.
17. Визначити та порівняти максимальні за модулем значення елементів, розташованих на побічних діагоналях двох довільних матриць різної розмірності.
18. Визначити та порівняти максимальні парні елементи, розташовані на головних діагоналях в двох довільних матрицях різної розмірності.
19. Визначити та порівняти мінімальні значення непарних елементів, що розташовані на побічних діагоналях, для двох довільних квадратних матриць різної розмірності.
20. Визначити та порівняти добутки елементів, кратних п'яти, що розташовані на головних діагоналях двох довільних матриць різної розмірності.
21. Визначити та порівняти мінімальні додатні елементи, розташовані на побічних діагоналях, в двох довільних матрицях різної розмірності.
22. Визначити та порівняти мінімальні за модулем значення елементів, розташованих на побічних діагоналях двох довільних матриць різної розмірності.
23. Записати в окремий масив максимальні елементи стовпців довільної матриці. Визначення та запис максимальних елементів в масив виконати в підпрограмі.
24. Записати в окремий масив максимальні елементи кожного з рядків довільної матриці. Визначення та запис максимальних елементів в масив виконати в підпрограмі.
25. Записати в окремий масив мінімальні елементи стовпців довільної матриці. Визначення та запис мінімальних елементів в масив виконати в підпрограмі.
26. Записати в окремий масив мінімальні елементи кожного з рядків довільної матриці. Визначення та запис мінімальних елементів в масив виконати в підпрограмі.
27. Записати в окремий масив середні значення елементів кожного зі стовпців довільної матриці. Визначення та запис середніх значень елементів в масив виконати в підпрограмі.

Додаток Д.

Завдання для самостійної роботи по темі «Створення та використання функцій користувача»

Розробити програму (відповідно своєму варіанту завдання), яка реалізує поставлену задачу у функції (функціях) користувача. Організувати виведення інформації в формі таблиці.

1. Обчислити та запам'ятати значення $\mathbf{z_i = F(x_i, y_i)}$ в $\boldsymbol{n}$ різних точках $\mathbf{(x_i, y_i)}$, а потім знайти суму всіх отриманих значень функції. Прийняти $\boldsymbol{n} = 14$; $\mathbf{x_{i+1} = x_i + \delta x}$ ($x_0 = 2.8$; $\delta x = 2.1$); $\mathbf{y_{i+1} = y_i + \delta y}$ ($y_0 = 1.7$; $\delta y = 0.3$);

$$F(x_i, y_i) = [\ln(x_i^2) - \ln(y_i^2)]\frac{\sqrt[3]{x_i - y_i}}{ctg(x_i)};$$

2. Обчислити значення: $y_{ij} = z_i * \cos^2 a_j + \sqrt{z_i^3 + a_j * z_i}$; $z_i = \cos(x_i + 1)$.

   Прийняти $x_i = 0{,}3;\ 0{,}8;\ 1{,}2$; $a_j = 1{,}1;\ 1{,}5;\ 1{,}9;\ 2{,}2$.

3. Обчислити значення: $y_{ij} = z_i^2 + \sqrt{z_i^2 - a_j}$; ãää $z_i = e^{1/x_i} + x_i^2$.

   Прийняти $x_i = 1{,}2;\ 1{,}8;\ 1{,}6;\ 1{,}4$; $a_j = 0{,}4;\ 0{,}5;\ 0{,}8$.

4. Обчислити та запам'ятати значення $\mathbf{z_i = F(x_i, y_i)}$ в $\boldsymbol{n}$ різних точках $\mathbf{(x_i, y_i)}$, де $\boldsymbol{n} = 14$; $\mathbf{x_{i+1} = x_i + \delta x}$ ($x_0 = 1.2$; $\delta x = 0.25$); $\mathbf{y_i = x_i^2}$;

$$F(x_i, y_i) = \lg^2(x_i * y_i) + \frac{\sqrt{x_i^2 + y_i^2}}{x_i + y_i}.$$

5. Обчислити та запам'ятати значення $\mathbf{z_i = F(x_i, y_i)}$ в $\boldsymbol{n}$ різних точках $\mathbf{(x_i, y_i)}$, де $\boldsymbol{n} = 15$; $\mathbf{x_{i+1} = x_i + \delta x}$ ($x_0 = 4.2$; $\delta x = 1.8$); $\mathbf{y_{i+1} = y_i + \delta y}$ ($y_0 = 1.82$; $\delta y = 0.33$);

$$F(x_i, y_i) = [\ln(x_i^2) - \ln(y_i^2)]\frac{\sqrt[3]{x_i - y_i}}{ctg(x_i)};$$

6. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в $\boldsymbol{n}$ різних точках (x_i, y_i), де $\boldsymbol{n} = 16$; x_i та y_i – довільні додатні числа;

$$F(x_i, y_i) = \sin^2(x_i^2 * y_i) + \cos^3(x_i * y_i^3) - \sqrt{\lg^3 x_i^2};$$

7. Обчислити та запам'ятати значення $\mathbf{z_i = F(x_i, y_i)}$ в $\boldsymbol{n}$ різних точках $\mathbf{(x_i, y_i)}$, де $\boldsymbol{n} = 16$; $\mathbf{x_i}$ – довільні додатні числа; $\mathbf{y_i = \lg^2 x_i^3 + (i\text{-}1)/10}$

$$F(x_i, y_i) = tg^2(x_i - y_i) * (x_i^2 + 0{,}5 * y_i).$$

8. Обчислити значення: $y_{ij} = z_i + \sqrt{z_i^2 + a_j^3 \cdot z_i^{0,3}};$ $z_i = e^{-5 \cdot x_i}$.

Прийняти x_i = 0,1; 0,4; 0,8; 1,2; a_j = 0,2; 0,5; 0,9.

9. Обчислити значення: $y_{ij} = (z_i^{\,2} + 0{,}33)/\mathrm{tg} z - b * \sqrt{\left| z_i^{\,2} - a_j \right|};\ z_i = e^{-1/x_i} + x_i^{\,2}.$

Прийняти x_i = 1,3; 1,5; 1,7; 1,9; a_j = 0,33; 1,12; 3,72; 5,16; b=3.68.

10. Обчислити значення:

$$y_{ij} = z_i^3 - (z_i + a_j * b)/(a_j^{\,0,4} + z_i^2);\ z_i = e^{-1/x_i} + x_i * b * \cos x_i.$$

Потім знайти суму всіх отриманих значень функції y_{ij}.
Прийняти x_i = 1,3; 1,5; 1,8; 2,0; a_j = 0,7; 0,85; 0,9; b = 5,75.

11. Обчислити та запам'ятати значення $\mathbf{z_i = F(x_i, y_i)}$ в n різних точках $\mathbf{(x_i, y_i)}$, а потім знайти добуток всіх отриманих значень функції. Прийняти n = 16; $\mathbf{x_{i+1} = x_i + \delta x}$ (x_0 = 1.3; δx = 0.2); $\mathbf{y_i = x_i^2}$;

$$F(x_i, y_i) = \lg^2(x_i * y_i) + \frac{\sqrt{x_i^{\,2} + y_i^{\,2}}}{x_i + y_i};$$

12. Обчислити та запам'ятати значення $\mathbf{z_i = F(x_i, y_i)}$ в n різних точках $\mathbf{(x_i, y_i)}$, де n = 15; $\mathbf{x_{i+1} = x_i + \delta x}$ (x_0 = 1.2; δx = 0.3); $\mathbf{y_i}$ – довільні додатні числа;

$$F(x_i, y_i) = e^{-1/x} + x^y - y^x$$

13. Обчислити та запам'ятати значення:

$$y_{ij} = A_i^2 + A_i \cdot B_i \cdot C_j - B_i^2 \cdot \sqrt{A_i + B_i};\quad A_i = sin^2 x_i;\quad B_i = e^{1/x_i} + 3.$$

а потім знайти суму всіх отриманих значень функції y_{ij}.
Прийняти x_i = 1,8; 1,4; 1,1; 1,5; C_j = 3,11; 2,21; 2,87.

14. Обчислити та запам'ятати значення $\mathbf{z_i = F(x_i, y_i)}$ в n різних точках $\mathbf{(x_i, y_i)}$, а потім знайти суму перших п'яти значень та добуток двох останніх. Прийняти n = 18; $\mathbf{x_{i+1} = x_i + \delta x}$ (x_0 = 1.2; δx = 0.55); $\mathbf{y_i}$ – довільні додатні числа;

$$F(x_i, y_i) = e^{-2/x} + 0{,}8 \cdot x^y - 1{,}7 \cdot y^x.$$

15. Обчислити значення: $y_i = a_i^{b_i} + b_i^{a_i};\quad a_i = e^{-1/x_i} + x_i \cdot C;\quad b_i = cos^2 x_i + x_i^{0,3}.$

Прийняти x_i = 0,5; 0,8; 0.9; 1,2; 1,3; 1.5; 1,7; 1,8; 1,9; 2.1; 2.3; 2.4; C=1,86;.

16. Обчислити та запам'ятати значення $\mathbf{z_i = F(x_i, y_i)}$ в n різних точках $\mathbf{(x_i, y_i)}$, де n = 18; $\mathbf{x_i}$ – довільні додатні числа; $\mathbf{y_i}$ – довільні від'ємні числа;

$$F(x_i, y_i) = 12 \cdot x_i^{0.65} - (1{,}9 \cdot x_i^2 - 0{,}85 \cdot y_i^{0.8}) \cdot tg^2(x_i) / y_i.$$

17. Обчислити та запам'ятати значення:

$$y_{ij} = (A_i \cdot \sin B_j + B_j \cdot \cos A_i) \cdot (A_i^{0,5} + \sqrt[3]{B_j});$$
$$A_i = \cos^2 x_i + x_i^3; \quad B_j = \sin^2 a_j + a_j^{0,5}.$$

Прийняти x_i = 1,2; 0,4; 1,3; 0,8; a_j = 0,6; 0,9; 1,9.

18. Обчислити значення: $y_{ij} = z_i + (a_j * z_i * b + b^{0,3})/\sqrt{z_i^2 + 1}$; $z_i = e^{-1/x_i} + x_i * b$.

Прийняти x_i = 1,2; 1,6; 1,9; 1,95; a_j = 0,9; 0,95; 0,99; b=1.68.

19. Обчислити значення: $y_{ij} = z^{0,3} + z_i \cdot a_j$; $z_i = x_i^2 + \ln x_i$.

Прийняти x_i = 1,9; 1,5; 1,4; 1,2; a_j = 0,2; 1,1; 1,4.

20. Обчислити та запам'ятати значення $\mathbf{z_i = F(x_i, y_i)}$ в $\boldsymbol{n}$ різних точках $\mathbf{(x_i, y_i)}$, а потім знайти суму перших п'яти значень та добуток двох останніх. Прийняти $\boldsymbol{n}$ = 16; $\mathbf{x_{i+1} = x_i + \delta x}$ (x_0 = 1.5; δx = 0.65); $\mathbf{y_i}$ – довільні додатні числа;

$$F(x_i, y_i) = 0{,}8 \cdot x^y - 1{,}7 \cdot y^x + \ln\frac{x}{y}.$$

21. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в $\boldsymbol{n}$ різних точках (x_i, y_i), де $\boldsymbol{n}$ = 16; $x = x_i * \delta x$ (x = 1.8; δx = 1.2); y_i – довільні додатні числа ($y_i < 1$);

$$F(x_i, y_i) = \frac{\sqrt{1 - y_i}}{x_i + \operatorname{tg} x_i} * (1 + e^{-x_i * y_i}).$$

22. Обчислити значення:

$$y_{ij} = z_i^3 - (z_i + a_j * b)/(a_j^{0,3} + z_i^2); \; z_i = e^{-2/x_i} + x_i * b * \cos(x_i).$$

Прийняти x_i = 1,2; 1,6; 1,9; 2,2; a_j = 0,75; 0,8; 0,9; 0,95; b = 6,74.

23. Обчислити та запам'ятати значення $z_i = F(x_i, y_i)$ в $\boldsymbol{n}$ різних точках (x_i, y_i), де $\boldsymbol{n}$ = 18; $x_{i+1} = x_i + e^{\delta x}$ (x_0 = 2; δx = 0,1); y_{i+1}= cy*(1 – y_i) (y_0 =3; cy= -2);

$$F(x_i, y_i) = 15^{2x-3y} + \sin^2(x^3 + y^4) + x^2 y^3.$$

24. Обчислити значення: $y_{ij} = 0{,}88 \cdot z_i + \sqrt{\alpha \cdot z_i^2 + a_j^{2,5} \cdot z_i^{0,45}}$; $z_i = \lg(3{,}8 \cdot x_i)$.

Прийняти x_i = 1,1; 1,4; 1,8; 2,2; 2,5; 3,1; a_j = 0,2; 0,5; 0,9.

25. Обчислити та запам'ятати значення:

$$y_{ij} = A_i^{2,2} + 1{,}5 \cdot A_i \cdot B_i \cdot C_j - B_i^{1,5} \cdot \sqrt{A_i + 2B_i}; \quad A_i = 1{,}8 \cdot \cos^3 x_i; \quad B_i = e^{2/x_i} + 2{,}6.$$

а потім знайти суму всіх отриманих значень функції y_{ij}.

Прийняти x_i = 2,2; 1,9; 1,7; 1,4; 1,2; C_j = 2,31; 2,86; 3,57.

26.Обчислити значення: $y_i = a_i^{2b_i} + b_i^{3a_i}$; $a_i = e^{2,5/x_i} + 3x_i \cdot C$; $b_i = sin^2 x_i + x_i^{0,5}$.

Прийняти x_i = 0,3; 0,7; 0.9; 1,1; 1,2; 1.4; 1,7; 1,8; 1,9; 2.2; 2.5; 2.7; C=2,15;.

Додаток Е.

Завдання для самостійної роботи по темі «Обробка символьних даних»

Виділення окремих символів (по одному)

1. Надрукувати задане слово в стовпчик. (*Використати функцію* ***Mid***). (*****)

1а. Надрукувати задане слово так, щоб кожна буква була розташована у наступних рядку та стовпці.

Визначення кількості символів

2. Визначити кількість букв у заданому слові (*потім додатково – кожному в реченні*). (*****)
3. Визначити, яке з набору окремо введених слів є найкоротшим (найдовшим) та надрукувати його.

Пошук заданого символу та визначення його порядкового номеру

4. Порахувати кількість слів у введеному довільному реченні. (*****)
5. Визначити яке зі списку[28] введених слів є найкоротшим та надрукувати його.
6. Розбити ведений список довільних слів[2] на окремі слова і надрукувати їх кожне з нового рядка.
7. Розбити введене довільне речення на окремі слова і надрукувати їх кожне з нового рядка. (*****)

Приклади використання функцій Left, Right, Mid на окремих словах

8. Замінити букву у слові "БЕНЗИН" на іншу і одержати та надрукувати слово "БЕНЗОЛ".
9. Зі слова "ПАМИР" виділити та надрукувати слово "МИР".
10. Зі слова "МАКУХА" отримати та надрукувати два нових слова.
11. З двох слів "МАК" та "УХА" отримати та надрукувати одне, яке утворено внаслідок їх злиття.
12. У слові "БАНКА" поміняти місцями склади та надрукувати результат.
13. Розділити довільне слово на склади та надрукувати їх (за умови, що кожний склад має по дві літери).
14. Розділити довільне слово на склади та надрукувати їх (за умови, що кожний склад має по три літери).

Порівняння символів

15. Скласти програму, що порівнює дві послідовно введені букви та розташовує їх за алфавітом.

---

[28] Список слів вводиться як речення – разом (*слова записуються через кому*) на відміну від набору слів, де вони вводяться окремо.

Кодування символів

16. Розробити програму, що слово, введене прописними буквами (великі літери), переробляє у слово, записане рядковими буквами (малі літери) і навпаки.

Заміна символів в «слові»

17. Поміняти між собою перші букви у двох заданих словах.
18. Поміняти між собою перші склади у двох заданих словах (за умови, що кожний склад має по дві літери).
19. Поміняти між собою перші склади у двох заданих словах (за умови, що кожний склад має по три літери).