МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Хіміко-технологічний факультет
Кафедра кібернетики хіміко-технологічних процесів

ОБЧИСЛЮВАЛЬНА МАТЕМАТИКА ТА ПРОГРАМУВАННЯ

МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ
«VBA ДЛЯ ЗАВДАНЬ ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ»
ЧАСТИНА 1
ДЛЯ СТУДЕНТІВ ХІМІКО-ТЕХНОЛОГІЧНОГО ФАКУЛЬТЕТУ
СПЕЦІАЛЬНОСТІ
161 «ХІМІЧНІ ТЕХНОЛОГІЇ ТА ІНЖЕНЕРІЯ»

Рекомендовано Вченою радою хіміко-технологічного факультету
КПІ ім. Ігоря Сікорського

Київ
КПІ ім. Ігоря Сікорського
2017

Обчислювальна математика та програмування: метод. вказівки до викон. лаб. роб. «VBA для завдань обчислювальної математики» для студ. хіміко-технологічного факультету спеціальності 161 «Хімічні технології та інженерія» Ч. 1/ Уклад: С. Г. Бондаренко, О. В. Сангінова, А. О. Абрамова, С. І. Заєць – Київ: КПІ ім. Ігоря Сікорського. – 100 с.

Гриф надано вченою радою
хіміко-технологічного факультету КПІ ім. Ігоря Сікорського,
протокол № 5 від 23.05.2017

Електронне мережне навчальне видання

Обчислювальна математика та програмування

методичні вказівки до виконання лабораторних робіт
«VBA для завдань обчислювальної математики»
Частина 1
для студентів хіміко-технологічного факультету
спеціальності 161 «Хімічні технології та інженерія»

Укладачі: Бондаренко Сергій Григорович, к.т.н, доц.
Сангінова Ольга Вікторівна, к.т.н, доц.
Абрамова Алла Олександрівна, к.т.н, доц.
Заєць Сергій Іванович, зав. лаб.

Відповідальний редактор: А. О. Абрамова

Рецензент: В. І. Супрунчук

Зміст

Передмова

У технічній документації фірма Microsoft називає Visual Basic не інакше як системою програмування. Ця система програмування призначена для написання програм, що працюють під керуванням операційної системи Windows.

Visual Basic є продуктом багаторічної еволюції мови програмування Basic. Після появи системи Windows фірма Microsoft розробила графічний варіант мови Basic. В основному, ця мова призначалася для створення програм, що працюють у режимі діалогу з користувачем, тобто у візуальному режимі. Тому мова й названа Visual Basic. По мірі розвитку комп'ютерних технологій Visual Basic поступово перетворювався у потужний інструмент для розроблення прикладних програмних додатків.

Слово Basic у назві вказує лише на те, що синтаксис програм і оператори опираються на мову високого рівня Basic (Beginner's All Purpose Symbolic Instruction Code – символьна мова програмування загального призначення для початківців). Але при цьому Visual Basic дуже помітно відрізняється від звичайної мови Basic.

Технологія роботи у середовищі Visual Basic базується на ідеях візуального й об‘єктно-орієнтованого програмування. Візуальна мова програмування означає, що взаємодія із системою програмування при складному налагодженні програм реалізується діалоговими засобами графічного інтерфейсу користувача.

Під *об‘єктно-орієнтованим програмуванням* розуміють об'єднання даних і процедур їхньої обробки в об'єкти. Це спрощує створення й перетворення складних структур даних і дій над ними. Інакше кажучи, середовище Visual Basic є автоматизованою графічною оболонкою над об‘єктно-орієнтованою версією мови Basic.

Середовище програмування Visual Basic for Application (VBA) створене фірмою Microsoft на базі середовища програмування Visual Basic (VB).

VBA це об‘єктно-орієнтована мова, що призначена для програмування в рамках конкретної прикладної програми, і в цьому його головна відмінність від універсальних мов програмування (наприклад, від VB).

Найчастіше VBA використають для програмування в прикладних програмах Microsoft Office. Він також вбудований у такі прикладні програми, як Visio, AutoCad, CorelDraw, WordPerfect і т.д.

Кожна з прикладних програм Microsoft Office містить велику кількість об'єктів, які відрізняються один від одного. Кожний з об'єктів має свої властивості, методи й події. Хоча основні властивості VBA залишаються незмінними для різних прикладних програм Microsoft Office (вони об'єднані базовими операторами мови VB), але при цьому кожна прикладна програма має свої специфічні команди й об'єкти. Це й відрізняє VBA від універсального середовища програмування VB. Крім того, програми, розроблені в середовищі VB, після компіляції можуть експлуатуватися незалежно від середовища розроблення, а програми, розроблені у VBA, є компонентом певного документа однієї з прикладних програм, і вони можуть виконуватися тільки в середовищі своєї прикладної програми. Таким чином, програма, розроблена на VBA, входить до складу деякого документу Microsoft Office і призначена для автоматизації оброблення інформації в рамках цього документа.

Зазвичай в прикладній програмі (Microsoft Word, Excel, PowerPoint, Access) є вбудований редактор VB. У деяких програмах (OutLook, Internet Explorer) використається вкорочена версія VBA – VBA Script.

Вивчення мови VBA направлене на розширене використання прикладної програми Microsoft Excel і призначено для створення користувачем власних програм на VBA.

ЛАБОРАТОРНА РОБОТА № 1
АЛГОРИТМІЗАЦІЯ ТА ПРОГРАМУВАННЯ ІТЕРАЦІЙНИХ ТА РЕКУРЕНТНИХ ЗАЛЕЖНОСТЕЙ

Мета та основні завдання роботи: дослідити роботу і можливості операторів циклу While – Wend і Do – Loop. Вивчити алгоритм роботи циклів з передумовою та з післяумовою. Ознайомитися з правилами організації вкладених циклів.

Вивчити та стисло описати у протоколі лабораторної роботи:

- правила запису та роботу операторів циклу While – Wend і Do – Loop;
- розглянути алгоритми циклів з передумовою та з післяумовою.

1.1 Основні теоретичні відомості

Ітераційні цикли

Цикли For – Next використовують, коли заздалегідь відомо скільки разів, повинен відпрацювати цикл. Якщо це заздалегідь не відомо, то краще використовувати інші варіанти операторів циклу.

Циклічні структури, в яких кількість повторів завчасно не визначена, називається *невизначеним* або *ітераційним циклом*. Їх організація потребує включення у “тіло” циклу умови, яка буде визначати закінчення циклу.

Цикли While – Wend

Інструкція **While – Wend** виконує послідовність операторів “тіла” циклу, поки задана умова має значення ІСТИНА. Цикл **While – Wend** є циклом з передумовою. Його відносять до ітераційних циклів.

Формат:

```
While <умова >
```

“Тіло” циклу – оператори мови VBA

Wend

Якщо умова має значення True (Істина), виконуються всі оператори “тіла” циклу, що записані до оператора Wend. Потім управління повертається до інструкції While і знову перевіряється умова. Якщо умова як і раніше має значення True (Істина), то процес виконання операторів “тіла” циклу повторюється. Якщо умова вже не має значення True (Істина), а має – False (Хибність), то цикл закінчується і програма виконується з оператора, що записаний за ключовим словом Wend. Таким чином, виконання операторів “тіла“ циклу буде проводиться до тих пір, поки умова записана в операторові While буде істиною.

При використанні конструкції While – Wend треба стежити за тим, щоб до моменту першої перевірки логічної умови всі змінні, які входять в неї, були визначені (етап підготовки циклу).

Якщо умова не є істиною із самого початку, то конструкція циклу ігнорується (конструкція While – Wend є циклом з передумовою).

Цикли While – Wend можуть мати будь-яку глибину вкладеності. Кожен оператор While (початок циклу) повинен бути закінчений оператором Wend (кінець циклу).

Приклад: надрукувати цілі числа від 1 до 100 (рис. 1.1).

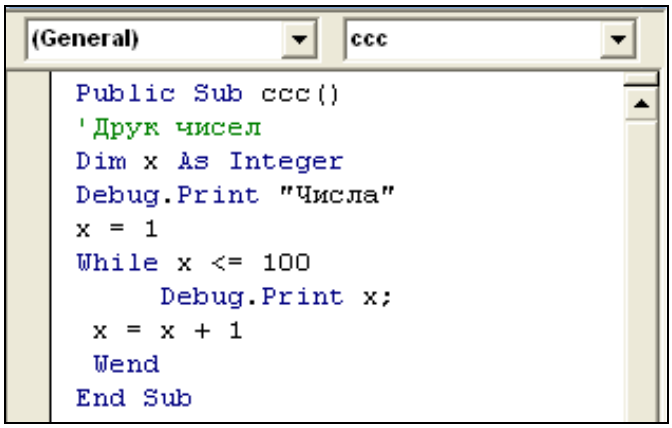


Рис.1.1 – Код програми

Результат виконання цієї програми представлено на рис.1.2.

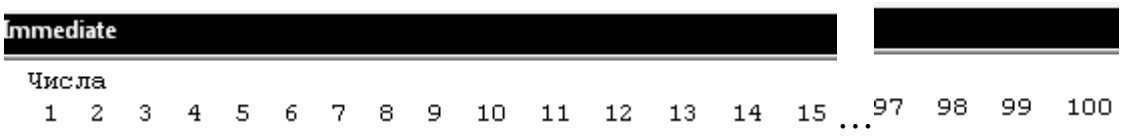


Рис.1.2 – Результат виконання програми

Дія цієї програми така ж сама, як і дія програми з використанням циклу For – Next. Порівняння цих програм дає можливість побачити синтаксичні відмінності циклів For – Next і While – Wend.

При використанні конструкції While – Wend треба організувати зміну значення параметра циклу в “тілі” циклу (в прикладі *x*), бо ця дія не виконується автоматично, як у циклі For – Next. Також початкове значення цієї змінної ***x*** задається перед конструкцією циклу While – Wend (в прикладі *x=1*), а не записується в заголовку циклу, як в циклі For – Next.

Цикли Do – Loop

Інструкція Do – Loop забезпечує більш структурований і гнучкий спосіб організації циклів, ніж конструкція While – Wend. В ній можна здійснювати перевірку, як на істинність умови, так і на її хибність. Існує можливість перевірки умови циклу, як на початку циклу, так і в кінці (вибір здійснює програміст). Таким чином користувач може будувати цикли з передумовою або цикли з післяумовою в залежності від своїх потреб.

Формат (загальний):

```
DO [{While|Until} <умова>]
“Тіло” циклу - оператори мови VBA
[Exit Do]
Loop [{While|Until} <умова>]
```

Зазвичай, щоб уникнути плутанини при роботі з цим циклом, його синтаксис записують окремо – для циклу с передумовою, і для циклу с післяумовою (табл.1.1).

Таблиця 1.1

Цикли Do – Loop з передумовою та з післяумовою

Цикли Do – Loop з передумовою	**Цикли Do – Loop з післяумовою**
Do {While|Until} <умова> “Тіло” цикла – оператори мови VBA [Exit Do] Loop	Do “Тіло” цикла– оператори мови VBA [Exit Do] Loop {While|Until} <умова>

Ключовими словами While або Until визначається спосіб перевірки умови. Конструкція While <умова> примушує повторювати “тіло“ циклу до тих пір, поки умова залишається істинною. Конструкція Until <умова> приводить до повторення “тіла“ циклу при хибності умови до тих пір, поки умова стане не істинною.

У конструкцію “тіла“ циклу Do – Loop може бути включений і оператор Exit Do, який перериває виконання циклу і програма продовжується з оператора, що записаний за ключовим словом Loop. Операторів Exit Do може бути будь-яка кількість і вони можуть розташовуватися в будь-якому місці “тіла“ циклу.

При використанні конструкції Do – Loop треба стежити за тим, щоб до моменту першої перевірки логічної умови всі змінні, які входять в неї були визначені (етап підготовки циклу).

Цикли Do – Loop можуть мати будь-яку глибину вкладеності. Кожен оператор Do (початок циклу) повинен бути закінчений оператором Loop (кінець).

Якщо умова, що забезпечує роботу циклу, не є істиною із самого початку (наприклад, While <умова>), то конструкція циклу з передумовою ігнорується, а цикл з післяумовою виконується тільки один раз.

Приклад: Надрукувати цілі числа від 1 до 100 (рис. 1.3).

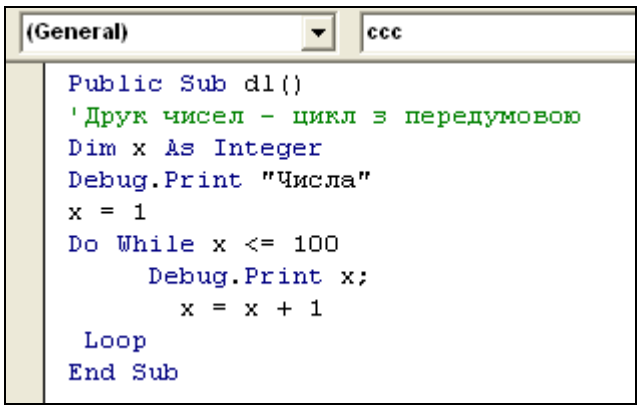


Рис.1.3 – Цикл з передумовою

Результат виконання цієї програми представлено на рис.1.4.

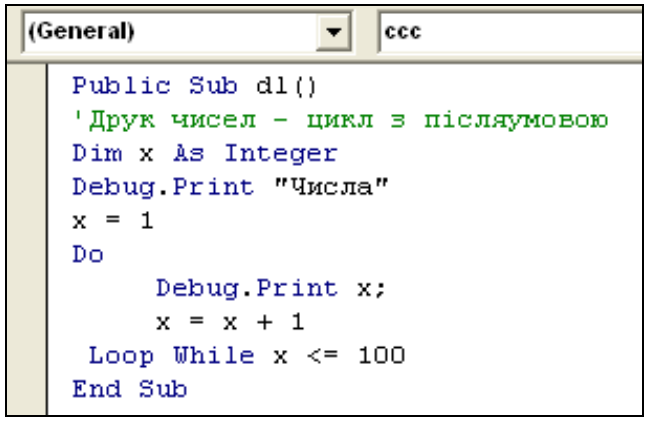


Рис.1.4 – Цикл з післяумовою

Приклад: обчислити суму нескінченого ряду з точністю ε = 0.001 при $|x|<1$. Формула ряду:

$$S = e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots \quad . \qquad (1.1)$$

Для складання алгоритму і програми розрахунку формалізуємо умову задачі. Позначимо елемент ряду y_i. Проаналізуємо яким чином змінюється сума ряду при збільшенні кількості доданків n. Проаналізувавши отримані дані визначимо, що задана точність буде досягнута, коли значення члена ряду, що додається, стане менше або буде дорівнювати значенню ε (табл.1.2).

Таблиця 1.2

Розрахунок ряду

Сума ряду	Точність ε	Елемент ряду y_i
$S_0 = 1$	–	–
$S_1 = 1 + \frac{x}{1!}$	$e_1 = \lvert S_1 - S_0 \rvert = \frac{x}{1!}$	$y_1 = \frac{x}{1!}$
$S_2 = 1 + x + \frac{x^2}{2!}$	$e_2 = \lvert S_2 - S_1 \rvert = \frac{x^2}{2!}$	$y_2 = \frac{x^2}{2!}$
$S_3 = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}$	$e_3 = \lvert S_3 - S_2 \rvert = \frac{x^3}{3!}$	$y_3 = \frac{x^3}{3!}$
…	…	…
	$e_n = \lvert S_n - S_{n-1} \rvert = \frac{x^n}{n!}$	$y_n = \frac{x^n}{n!}$

Для програмування такої задачі краще застосувати один з розглянутих ітераційних циклів. Аналіз даних та прийомів програмування показує, що при програмуванні даної задачі, краще застосувати цикл Do – Loop з післяумовою. Умовою закінчення такого циклу слід обрати: $y_n = \frac{x^n}{n!} \le e$.

Реалізація алгоритму рішення задачі у VBA представлена на рис.1.5.

```
(General)                                   ccc
Public Sub sr()
'Обчислення суми нескінченого ряда
Dim eps As Single, S As Single
Dim n As Integer, nf As Integer
Dim x As Single, y As Single
eps = Val(InputBox("Введіть точність eps", "Вікно введення", "0.001"))
x = Val(InputBox("Введіть x"))
Debug.Print "Контрольна сума ряду="; Exp(x)
S = 1 ' Початкове значення суми
n = 1 ' Перший член ряда
nf = 1 ' Початкове значення факторіала
Do
    nf = nf * n ' Розрахунок факторіала
    y = x ^ n / nf ' Розрахунок члена ряда
   S = S + y ' Наступна сума
    n = n + 1 ' Наступний номер
Loop While y >= eps
Debug.Print "При x = "; x, "сума ряду ="; S
Debug.Print "Просумовано"; n; "членів ряда"
End Sub
```

Рис.1.5 – Програма розрахунку ряду

Результат виконання цієї програми представлено на рис.1.6.

```
Immediate
Контрольна сума ряду= 1,64872127070013
При x =  0,5  сума ряду = 1,648698
Просумовано 6 членів ряда
```

Рис.1.6 – Результати виконання програми

Примітка. Зважаючи на те, що при помилковому програмуванні невизначених циклів найпоширенішою помилкою є створення нескінченного циклу, то необхідно вміти виконувати переривання таких процедур. Іноді таке переривання можна виконати за допомогою клавіші Esc (інколи допомагає одне натискання клавіші, а іноді клавішу слід притримати до здійснення переривання). Коли це не спрацьовує, слід застосувати комбінацію клавіш Ctrl + Break.

З метою більш поглибленого вивчення теоретичних основ даної лабораторної роботи рекомендується використати конспект лекцій з курсу та список рекомендованої літератури для даних методичних вказівок.

1.2 Опис лабораторних засобів та обладнання

Лабораторна робота виконується на персональному комп'ютері стандарту IBM PC під керуванням операційної системи MS Windows зі стандартним пакетом MS Office.

1.3 Заходи безпеки під час виконання лабораторної роботи

Заходи безпеки, яких треба дотримуватись при виконанні лабораторної роботи «Алгоритмізація та програмування ітераційних та рекурентних залежностей» наведені у Додатку А.

1.4 Послідовність виконання роботи

Згідно із отриманим завданням виконати наступне:

1. Вивчити принципи організації ітераційних циклічних алгоритмів, правила запису та роботи операторів циклу: Do…Loop (з різними варіантами умов) і циклу While…Wend.
2. Розробити алгоритм та написати програму, яка включає: обчислення суми нескінченного ряду для заданого значення x с заданою точністю e [1] (у відповідності до свого варіанту завдання, що містяться у Додатку Б).
3. Передбачити: обчислення та друк контрольного значення суми ряду (у відповідності з формулою, що представлена у вигляді ряду)[2] та друкування таблиці результатів розрахунку – № ітерації, величина доданку, поточне значення суми ряду.
4. Створити, відредагувати та налаштувати написану програму на комп'ютері.
5. Продемонструвати роботу програми викладачу.

---

[1] Значення x *і* ε вводити з клавіатури. Використати ітераційний цикл Do…Loop.
[2] Друкується для перевірки результату роботи програми.

6. Дослідити процес збіжності ряду в залежності від значення x та точності, що задається (наприклад, для двох значень x виконати розрахунки при e= 0.001 і e=0.0001[1]). Зробити висновки про результати проведеного дослідження.
7. Оформити протокол лабораторної роботи (алгоритм рішення задачі навести у вигляді блок-схеми).

Примітка. Умова досягнення необхідної точності має вигляд:

$| S_{k+1} - S_k | \le e$

де $S_k = \sum_{i=1}^{k} a_i$, $S_{k+1} = \sum_{i=1}^{k+1} a_i$, де a_i – член ряду.

1.5 Оброблення та аналізування результатів. Оформлення звіту

При оформленні звіту з лабораторної роботи до заздалегідь підготованого протоколу (див. завдання до самостійної роботи) додаються роздруковані аркуші з результатами виконаної роботи та рисунок блок-схеми алгоритму.

Контрольні запитання

1. Яка відмінність арифметичних та ітераційних циклічних алгоритмів?
2. Який формат запису оператору циклу While – Wend?
3. Дайте характеристики циклу While – Wend.
4. Яка кількість спрацювань циклу While – Wend, якщо умова, що забезпечує роботу циклу, не є істиною із самого початку?
5. Які синтаксичні відмінності циклів For – Next і While – Wend?
6. Який формат запису оператору циклу Do – Loop з передумовою?

[1] За згодою викладача можна вибрані інші значення точності ε.

7. Який формат запису оператору циклу Do – Loop з післяумовою?
8. Що дає використання оператору Exit Do у “тілі“ циклу Do – Loop?
9. В яких місцях циклу Do – Loop може бути записаний оператор Exit Do?
10. Яким буде спосіб перевірки умови в циклі Do – Loop при використанні ключових слів While або Until?
11. Яка різниця в роботі циклів з передумовою та з післяумовою?

ЛАБОРАТОРНА РОБОТА № 2
ОДНОВИМІРНІ МАСИВИ. ХАРАКТЕРНІ ПРИЙОМИ ПРОГРАМУВАННЯ

Мета та основні завдання роботи. Дослідити роботу циклічних структур для оброблення даних представлених у вигляді масивів. Ознайомитися з основними прийомами введення та виведення масивів. Вивчити характерні прийоми програмування.

Вивчити та стисло описати у протоколі лабораторної роботи:

– поняття статичних і динамічних масивів;

– прийоми введення, виведення та оброблення даних, що представлені у вигляді масивів;

– характерні прийоми програмування для даних, що представлені у вигляді масивів.

2.1 Основні теоретичні відомості

Масиви у VBA

У програмах на алгоритмічних мовах разом з простими змінними використовуються і індексовані змінні (масиви).

Масив (array) – це кінцевий набір однотипних даних, позначених одним і тим же загальним ім'ям. Правила утворення імен масивів ті ж, що і у імен простих змінних. Масиви – це зручний і широко поширений спосіб зберігання однотипних даних. Масиви корисні для створення списків даних, для зберігання табличної інформації і для багато чого іншого. Всі елементи даних, що зберігаються у масиві, мають бути одного типу. Наприклад, якщо створений масив для зберігання даних типу Integer, то кожен елемент масиву має бути числом типу Integer.

Масив дає можливість зберігати і обробляти всі свої елементи, використовуючи одну змінну. При цьому скорочується загальна кількість імен змінних, і з'являється можливість використання циклічних структур (в основному таких, як For – Next або For – Each) для оброблення елементів в масиві. У такий спосіб досягається значна ефективність при програмуванні. Так можна за допомогою простого фрагменту програми обробити велику кількість даних. Таким чином, можливість об'єднання груп елементів у масив дозволяє, з одного боку, полегшити масове оброблення даних, а з іншого, спростити ідентифікацію елементів масиву.

Масив складається з елементів. Кожен елемент масиву ідентифікується ім'ям масиву і індексом (індексами) записаними в круглих дужках. Індекси розділяють комами. Індекси можуть бути числовими виразами, змінними і константами. Якщо індекс є результатом обчислення виразу і має дробову частину, то в якості індексу береться значення виразу, що округлене до цілої частини.

Масив з одним індексом можна представити як рядок значень і називають його *одновимірним масивом*. Додавання другого індексу дає таблицю значень, а масив називають *двовимірним*. *Тривимірний масив* можна представити як куб. Фактично VBA дає можливість задавати до 60 розмірностей у масиві.

В основному при програмуванні використовують одновимірні масиви (з одним індексом) (табл.2.1) і двовимірні масиви (з двома індексами)(табл.2.2).

Таблиця 2.1.

Одновимірні масиви

Математика	Програмування
x_i , $i=1,2,3,\ldots,n$	*x(i)* , індекс *i* потрібно визначити
це ряд – $x_1,x_2,\ldots,x_n$	це ряд – x*(1),x(2),...,x(n)*

Таблиця 2.2.

Двовимірні масиви

Математика	Програмування
таблиця	значення
a_{11} a_{12} a_{21} a_{22}	A(1,1) A(1,2) і т.д.

Важливою характеристикою масиву є його розмірність (розмір) – загальна кількість елементів у ньому. Робота з масивами починається з їх *оголошення*. Це дозволяє розмістити елементи масиву в пам'яті. Для однієї змінної з заданим ім'ям пам'ять резервується в залежності від її типу. В цю область пам'яті буде записано її значення, масив має одне ім'я і декілька елементів. Тому при оголошенні масиву треба вказати не тільки тип даних, що зберігаються в ньому, а і кількість елементів масиву. І тоді VBA зарезервує пам'ять для зберігання елементів масиву.

Це можна зробити за допомогою оператора оголошення **Dim**. Оголошення масиву треба зробити перш, ніж він зустрінеться у програмі. Рекомендовано оголошувати масиви на початку програми. Структура оголошення масиву відрізняється від структури оголошення змінної тим, що тут потрібно вказувати різновид масиву і межі зміни індексів.

Використання оператора Dim для оголошення масивів у VBA

```
Dim <ім'я масиву1(розмірність)> [As <тип1>]
[, ім'я масиву2(розмірність) [As <тип 2 >]]…
```

Dim – оператор оголошення;

Ім'я масиву – ім'я, що формується за правилами утворення імен простих змінних;

Розмірність – загальна кількість елементів у масиві має наступний синтаксис:

[I_{1min} To] I_{1max} , [[I_{2min} To] I_{2max}] …

I_{1min}, I_{2min}, …– визначає нижню межу для індексів масиву (необов'язковий аргумент);

I_{1max}, I_{2max}, … – визначає верхню межу для індексів масиву (обов'язковий аргумент).

Тобто, розмірність можна вказати діапазонами зміни індексів або максимальним значенням відповідного індексу.

Приклади:

```
Dim A1(1 To 20) As Single
```

Оголошується одновимірний масив дійсних чисел одиночної довжини з ім'ям A1. Межа зміни індексу масиву задається в діапазоні від 1 до 20.

```
Dim B(5 To 10) As Single, C(0 To 100) As Integer
```

Оголошується два одновимірних масиви. Один – одновимірний масив дійсних чисел одиночної довжини з ім'ям B, а другий – одновимірний масив цілих чисел одиночної довжини з ім'ям C. Межа зміни індексу масиву B задається в діапазоні від 5 до 10, а масиву C – від 0 до 100.

```
Dim MyArray(1 To 10, 1 To 10) As Single
```

Оголошується двовимірний масив дійсних чисел одиночної довжини з ім'ям MyArray. Межа зміни першого і другого індексів масиву задається в діапазоні від 1 до 10.

У VBA за замовчуванням нумерація елементів у масиві починається з 0. Така система нумерації досить поширена в програмуванні і називається нумерацією з нульовою базою. В цьому випадку індекс будь-якої розмірності першого елементу дорівнює 0.

Оскільки нумерація з нульовою базою не дуже зручна (всі звикли до рахунку з 1, а не з 0), то в VBA є інструкція – **Option Base**, що дозволяє вказати нижню межу індексів масиву (0 або 1).

Оскільки за замовчуванням використовується значення 0, інструкція Option Base не є обов'язковою. Однак якщо Option Base використовується, то вона повинна з'являтися у модулі перед будь-якою процедурою. Інструкція Option Base може задаватися в модулі тільки один раз і повинна передувати опису масивів, що включають розмірності.

Інструкція має два варіанти написання:

- Option Base 0 – індекси масивів починаються з 0;
- Option Base 1 – індекси масивів починаються з 1.

Інструкцію Option Base не можна розміщувати всередині процедури. У модулі необхідно мати тільки одну інструкцію Option Base, її дія має силу для всіх масивів, описаних в модулі: байдуже, чи на рівні процедури, чи на рівні модуля. Якщо не використовується інструкція Option Base, то за замовчуванням нумерація масиву починається з 0.

Таким чином, у VBA є дві можливості, щоб призначити перший номер елементів масиву. По-перше, можна вказувати найменший номер індексів при описі масиву за допомогою оператора оголошення **Dim**, а по-друге, можна використовувати інструкцію **Option Base**.

Приклади:

```
Option Base 1
Dim Name(1000) As String
```

Межа зміни індексу масиву Name задається в діапазоні від 1 до 1000.

```
Dim A(25) As Integer, B1(75, 15) As Long
```

Межа зміни індексу масиву A задається в діапазоні від 1 до 25, а масиву B1 в діапазоні від 1 до 75 і другого індексу в діапазоні від 1 до 15.

```
Dim H(55) As String
```

Межа зміни індексу масиву H задається в діапазоні від 1 до 55.

Статичні і динамічні масиви

Масиви, що не змінюють число своїх елементів, називаються **статичними масивами**. Проте бувають ситуації, коли на початку роботи з масивом невідома кількість елементів в масиві, або ж, в процесі роботи ця кількість може змінюватися. Для такого роду випадків передбачені масиви, які називаються **динамічними масивами**, в яких можна змінювати кількість елементів при виконанні програми, точно вказуючи необхідне число елементів без марного витрачання пам'яті.

Динамічний масив створюється за допомогою оператора оголошення Dim. Але при цьому ім'я масиву супроводжується пустими круглими дужками, де параметр розмірність не вказується.

Приклад: оголошується одновимірний динамічний масив

```
Dim ArrayVar() As Single
```

У момент свого оголошення такий масив не містить жодного елементу.

Щоб використовувати масив по ходу виконання програми і додавати в нього елементи необхідно користуватися **інструкцією ReDim**:

Формат:

```
ReDim [Preserve] <ім'я масиву1(розмірність)> [As <тип1>]
[,<ім'я масиву2(розмірність)> [As type]]...
```

Ключове слово **Preserve** – необов'язковий аргумент. Його використання приводить до того, що дані вже існуючі у масиві, зберігаються після зміни його розмірності.

З оператором ReDim пов'язано одне важливе обмеження: якщо змінюємо розмір масиву, який вже містить дані, всі існуючі дані будуть повністю втрачені. Після виконання оператора ReDim вміст динамічного масиву встановлюється на значення за умовчанням, таке, як нуль (null).

Ключове слово Preserve, можна використовувати для збереження даних в масиві при зміні його розміру.

Приклади:

```
ReDim ArrayVar(50) As Single
```

Встановлює розмірність динамічного масиву, що дорівнює 50 елементам.

```
Dim A() As Single
```

Оголошує динамічний масив.

```
ReDim A(20)
```

Встановлює розмірність одновимірного динамічного масиву, що дорівнює 20 елементам.

```
ReDim A(1 To 30)
```

Змінює розмірність масиву до 30 елементів.

```
ReDim Preserve A(1 To 40)
```

Змінює розмірність масиву до 40 зі збереженням вмісту попереднього масиву.

```
Dim AB() As Single
```

Оголошує динамічний масив.

```
ReDim AB(4, 4)
```

Робить масив двовимірним масивом.

```
ReDim AB(3, 7)
```

Змінює розмірність двовимірного масиву.

```
ReDim Preserve AB(3, 8)
```

Змінює останній розмір масиву, зі збереженням вмісту попереднього масиву.

Примітка. При оголошенні масивів слід пам'ятати, що включення розмірності до оголошення масиву створює статичний масив з фіксованим числом елементів, а пропуск інструкції розмірність створює динамічний масив.

Інструкція Option Base впливає на загальне число елементів у масиві. Інструкцію ReDim можна використовувати необхідну кількість разів для

зміни у динамічному масиві кількості елементів і розмірностей. За допомогою інструкції ReDim можна змінити і тип даних, але тільки у тому випадку, коли динамічний масив зберігається у змінній типу Variant.

Функції LBound, Ubound

Функція LBound повертає перше значення індексу статичного або динамічного масиву, а функція UBound повертає верхнє граничне значення індексу статичного або динамічного масиву.

Формат:

```
LBound (ім'я масиву [, розмірність])
UBound (ім'я масиву [, розмірність])
```

Розмірність – ціле число (необов'язковий аргумент), що вказує розмірність яку повертає функція (нижню або верхню). Визначає вимірювання масиву, для якого треба отримати верхню або нижню межу (для першого вимірювання слід вказати 1, для другого – 2, для третього – 3 і так далі). За відсутності аргументу розмірності повертається межа для першого індексу масиву.

Приклад:

```
Dim TMP() As Single
……
ReDim TMP(20)
……...
For i = 0 To UBound(TMP)
……
```

В прикладі верхня межа масиву визначається за допомогою функції UBound. Ця методика достатньо гнучка, бо якщо надалі масив збільшиться або зменшиться, то цикл For – Next автоматично підстроїться під новий розмір масиву.

Очищення і видалення масивів за допомогою оператора Erase

Оператор **Erase** дозволяє для статичних масивів виконувати очищення від всіх його елементів, а для динамічних масивів виконувати видалення із пам'яті масиву разом із його вмістом.

Коли масив заповнений, дані в масиві залишаються до тих пір, поки користувач не присвоїть їм нові значення (або поки VBA не звільниться від масиву). Часто буває, що в подальших обчисленнях динамічний масив не буде використовуватися і тому недоцільно зберігати його в пам'яті комп'ютера, оскільки це може позначитися на швидкості роботи програми. Або ж може знадобитися очистити всі значення в статичному масиві, змінюючи числові значення на 0, а строкові – на порожні рядки. Це можна здійснити за допомогою операторів циклів (в тілі циклу присвоїти елементам масиву значення нуля, або порожнього рядка).

Оператор Erase дозволяє зробити таку процедуру значно простіше.

Формат:

```
Erase <ім'я масиву1>  [,<ім'я масиву2> …]
```

Даний оператор обнуляє масив з вказаним ім'ям (якщо масив статичний) або очищає пам'ять від масиву з указаним ім'ям (якщо масив динамічний).

Приклад:

```
Erase ArrayVar
```

Масив з ім'ям буде очищений або видалений.

При спробі доступу до динамічного масиву після його видалення за допомогою оператора Erase буде виведено вікно з повідомленням про помилку. Якщо ж доступ потрібен, то динамічний масив треба створити знову за допомогою інструкції ReDim.

Дія оператора Erase для статичних масивів залежить від типу даних, що зберігаються у масиві, як це показано в таблиці 2.3.

Таблиця 2.3

Дія оператора Erase для статичних масивів

Тип статичного масиву	**Дія оператора *Erase***
Будь-який числовий тип	Обнуляє елементи масиву
Будь-який строковий тип	Присвоює елементам масиву значення рядка нульової довжини, а для рядків фіксованої довжини – всім символам присвоює значення пропуску
Тип Variant	Присвоює елементам масиву значення Empty
Тип Object	Присвоює елементам масиву значення Nothing
Будь-який призначений для користувача тип	Кожній змінній в визначеному користувачем типу присвоює значення : числовому типу – значення 0; строковому типу – значення рядка нульової довжини; типу Variant – значення Empty; типу Object – значення Nothing

Робота з масивами в VBA

Одновимірні масиви і робота з ними

У програмі до елемента масиву звертаються за іменем і індексом, що записаний у круглих дужках.

A(5) – це 5-й елемент масиву.

Якщо записано A(i), то програма в залежності від числового значення індексу, обирає потрібний елемент масиву. Якщо i=5, то це 5-й елемент масиву.

А якщо значення індексу не було визначене, то це буде нульовий елемент масиву A(0). Якщо встановлено Option Base 0, то це може бути

тільки логічною помилкою, а якщо встановлено Option Base 1, то помилка буде синтаксичною і логічною одночасно.

Введення елементів масиву

Найпростіший спосіб введення значень елементів масиву це введення з клавіатури за допомогою функції *InputBox*. І якщо здійснювати введення без використання циклічних структур, то записується воно приблизно так:

```
Public Sub vvv()
Dim A(1 To 8) As Integer, i As Integer
'Заповнення масиву
A(1) = Val(InputBox("Введіть число", " Заповнення масиву"))
A(2) = Val(InputBox("Введіть число", " Заповнення масиву"))
A(3) = Val(InputBox("Введіть число", " Заповнення масиву"))
A(4) = Val(InputBox("Введіть число", " Заповнення масиву"))
.........................
End Sub
```

Елементам масиву можна присвоїти потрібні значення за допомогою оператора присвоювання:

```
a(1)=5
a(2)=3
a(3)=0
a(4)=4
……..
```

Якщо треба ввести велику кількість даних, то такі процедури мало кому сподобаються. З використанням операторів циклу процедура введення сприймається не такою обтяжливою:

```
Public Sub vvv()
Dim A(1 To 10) As Integer, i As Integer
```

```
' Заповнення масиву
For i = 1 To 10
   A(i) = Val(InputBox("Введіть число", "Заповнення масиву"))
Next i
 'Повідомлення про закінчення введення
MsgBox ("Введення масиву закінчено")
End Sub
```

Коли на початку роботи з масивом невідома кількість елементів, або ж, в процесі роботи ця кількість може змінюватися, то у цьому випадку слід застосовувати динамічні масиви.

Приклад:

```
Option Base 1
Public Sub vvv()
Dim A() As Integer, i As Integer
Dim N As Integer
'Заповнення масиву
N = Val(InputBox("Введіть розмірність масиву N="))
ReDim A(N)
For i = 1 To N
A(i) = Val(InputBox(" Введіть A( " & i & ") елемент масиву "))
Next i
'Повідомлення про закінчення введення
MsgBox ("Введення масиву закінчено")
End Sub
```

Для перевірки роботи розроблених програм і економії часу на введення даних масив можна заповнити випадковими числами.

Приклад:

Заповнити масив випадковими числами в діапазоні від 0 до 100.

```
Public Sub vvv()
Dim A(1 To 10) As Integer, i As Integer
'Заповнення масиву
```

```
Randomize
For i = 1 To 10
     A(i) = Int(Rnd * 100)
Next i
'Повідомлення про закінчення введення
MsgBox ("Введення масиву закінчено")
End Sub
```

Процедура **Randomize** запускає (ініціалізує) датчик випадкових чисел, функція **Rnd** без аргументу повертає випадкове дійсне число з діапазону (0,1), а функція **Int( )** повертає цілу частина дійсного числа.

Щоб отримати значення випадкових чисел в інтервалі від мінімуму (*min*) до максимуму (*max*) використовується формула: *Int((max-min+1)*Rnd +min).*

Приклад: отримати випадкове число від 1 до 6 із використанням процедури Randomize.

```
Dim MyValue
Randomize ' Ініціалізує генератор випадкових чисел.
MyValue = Int((6 * Rnd) + 1)' Повертає випадкове число від 1 до 6.
```

Зручним способом визначення одновимірних масивів є **функція Array**, що перетворює список елементів, розділених комами, в масив із цих значень.

Формат:

Array(список аргументів)

Такий запис дозволяє автоматично створити масив потрібного розміру, відразу завантажити в нього задані значення і повернути його як змінну типа Variant. Список аргументів функції Array – це розділений комами список значень, які присвоюються елементам масиву. Якщо аргументи не вказані, створюється пустий масив.

Приклад:

```
Dim A As Variant
A = Array(10, 20, 30)
```

```
B = A(2)
```

У даному прикладі змінна A створюється як одновимірний масив, що складається з трьох елементів (10, 20, 30), а змінна B набуває значення другого елементу масиву A(2) і дорівнює 20.

Також введення даних в масив можна здійснити з файлу даних і з робочого листа Excel.

Виведення елементів масиву

Процедура виведення елементів масиву також здійснюється з використанням операторів циклу. Зручно виводити елементи масиву у вікно Immediate за допомогою оператора Debug.Print.

Приклад:

```
Debug.Print "Result"
For i = 1 To UBound(A)
     Debug.Print  A(i);
Next i
```

Якщо потрібно вивести тільки один (заданий) елемент масиву, оператори циклу не потрібні і оператор Debug.Print записується в програмному рядку.

Приклад:

```
i = 5
Debug.Print A(i)
```

або

```
Debug.Print A(11)
```

Перший оператор Debug.Print виводить 5-й елемент масиву, а другий – 11-й елемент масиву.

При використанні функції MsgBox для виведення елементів масиву існує необхідність у формуванні повідомлення виводу (prompt).

Приклад:

```
prompt = ""    'Початкове повідомлення - пустий рядок
For i = 1 To n
    prompt = prompt & CStr(A(i)) & " "
Next i
MsgBox prompt
```

Приклад:

```
answer = ""
For i = 1 To UBound(A)
  answer = answer & A(i) & " "
Next i
MsgBox answer
```

Також виведення даних з масиву можна здійснити у файл даних і на робочий лист Excel.

Приклад:

Задано цілочисельний масив А з 10 елементів. Піднести всі елементи масиву до квадрату. Передбачити друк початкового і отриманого масивів.

(General) mac

```
Option Base 1
Public Sub mac()
Dim A(10) As Integer, i As Integer
' Оброблення цілочисельного масиву
 For i = 1 To 10
      A(i) = Val(InputBox("Введіть число", "Заповнення масиву"))
 Next i
Debug.Print "Початковий масив"
For i = 1 To 10
     Debug.Print A(i);
Next i
For i = 1 To 10
          A(i) = A(i) ^ 2
Next i
Debug.Print
Debug.Print "Змінений масив"
For i = 1 To 10
 Debug.Print A(i);
Next i
End Sub
```

Рис. 2.1 – Код програми

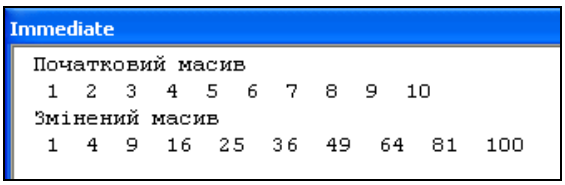


Рис. 2.2 – Результати роботи програми

Приклад:

У цілочисельному масиві А зменшити всі елементи на 1. Початковий і отриманий масиви надрукувати.

```
(General)                                    mac
Public Sub mac()
Dim A() As Integer, i As Integer
    ' Оброблення цілочисельного масиву
N = Val(InputBox("Введіть розмірність масиву N="))
ReDim A(N)
 For i = 1 To N
      A(i) = Val(InputBox(" Введіть A( " & i & ") елемент масиву "))
 Next i
 Debug.Print "Початковий масив"
For i = 1 To N
     Debug.Print A(i);
Next i
For i = 1 To N
          A(i) = A(i) - 1
Next i
Debug.Print
 Debug.Print "Змінений масив"
   For i = 1 To UBound(A)
     Debug.Print A(i);
   Next i
End Sub
```

Рис. 2.3 – Код програми

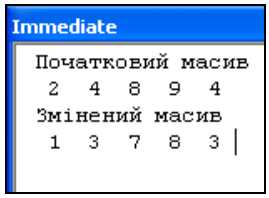


Рис. 2.4 – Результати роботи програми при n=5

Характерні прийоми програмування

1.Обчислення суми елементів масиву.

Для обчислення суми елементів множини потрібно першому значенню суми присвоїти значення нуля – S=0 (перед циклом), а потім накопичувати

суму за формулою (у циклі): **S=S+Y** (S – проміжне значення суми; Y– елемент множини).

Приклад:

Обчислити суму елементів масиву А з шістьох елементів – А(6).

Нехай елементи масиву: 3, 5, 4, 2, 1, 10.

```
(General)                                   suma
Option Base 1
Public Sub suma()
Dim A(6) As Integer, i As Integer
    ' Пошук суми
    For i = 1 To 6
          A(i) = Val(InputBox("Введіть число"))
    Next i
Debug.Print "Початковий масив"
For i = 1 To 6
     Debug.Print A(i);
Next i
S = 0
For i = 1 To 6
          S = S + A(i)
Next i
Debug.Print
Debug.Print "S ="; S
End Sub
```

Рис. 2.5 – Код програми

```
Immediate
 Початковий масив
  3   5   4   1   2   10
 S = 25
```

Рис. 2.6 – Результати роботи програми

2.Обчислення добутку елементів масиву.

Для рішення задачі обчислення добутку елементів множини потрібно присвоїти початковому значенню добутку значення одиниці – D=1 (перед циклом), а потім накопичувати добуток за формулою (в циклі): **D = D * Y** (D– проміжне значення добутку; Y – елемент множини).

Примітка. Якщо початковому значенню добутку присвоїти значення 0 то і добуток всіх елементів множини буде дорівнювати 0.

Приклад:

Знайти добуток елементів масиву. Обчислити добуток елементів масиву А з шістьох елементів – А(6). Нехай елементи масиву: 3, 5, 4, 2, 1, 10.

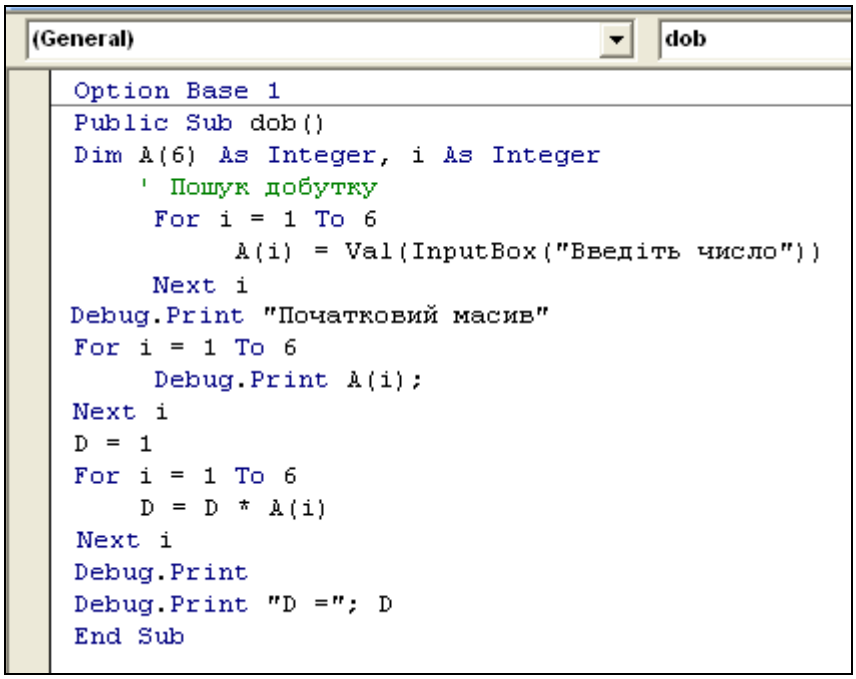


Рис. 2.7 – Код програми

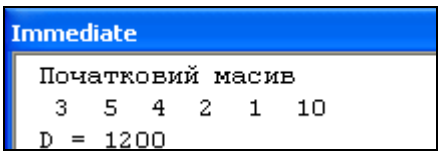


Рис. 2.8 – Результати роботи програми

Програмування процесів пошуку, сортування і злиття

Процеси пошуку пов'язані із знаходженням одного або декількох елементів множини, які задовольняють деякій ознаці (або деяким ознакам одночасно). Ознаки, за якими здійснюють пошук можуть бути самими різними.

Наприклад:

– пошук від’ємних значень;

– пошук найбільшого або найменшого значень;

– пошук значень кратних деякому числу;
– пошук значень, що не перевищують (перевищують) деяку константу;
– пошук значень, що належать заданому інтервалу.

Зазвичай множина задана у вигляді масиву (одновимірного або двовимірного). Через це процес пошуку неминуче пов'язаний з перебором елементів масиву, і перевірці кожного з елементів на ознаку, що досліджується. Масив обробляється із застосуванням циклічних структур. Обрані елементи масиву можна надрукувати, зберегти у файл, записати в новий масив.

Пошук найбільшого або найменшого значень елементів масиву

Алгоритм пошуку

Спочатку обираємо «еталон». Потім порівнюємо по черзі всі елементи масиву з цим «еталоном». «Еталон» зберігає своє значення, якщо він витримує порівняння з поточним елементом, і замінюється цим елементом, якщо він не витримує порівняння. При цьому необхідно пам’ятати, що потрібно не тільки обрати «еталон», а й надати йому початкове значення.

Пошук максимального елемента (max)

Для пошуку максимального елемента масиву рекомендується до початку циклу задати початкове значення «еталона» у вигляді дуже маленького числа (меншого за всі елементи) масиву, а потім вже у циклі робити порівняння.

Приклад:

В масиві із 6 елементів знайти найбільший елемент. В якості «еталона» обираємо дуже маленьке число – (-1E-34). Збережемо це значення в змінній MAX (MAX= -1E-34 - «еталон»).

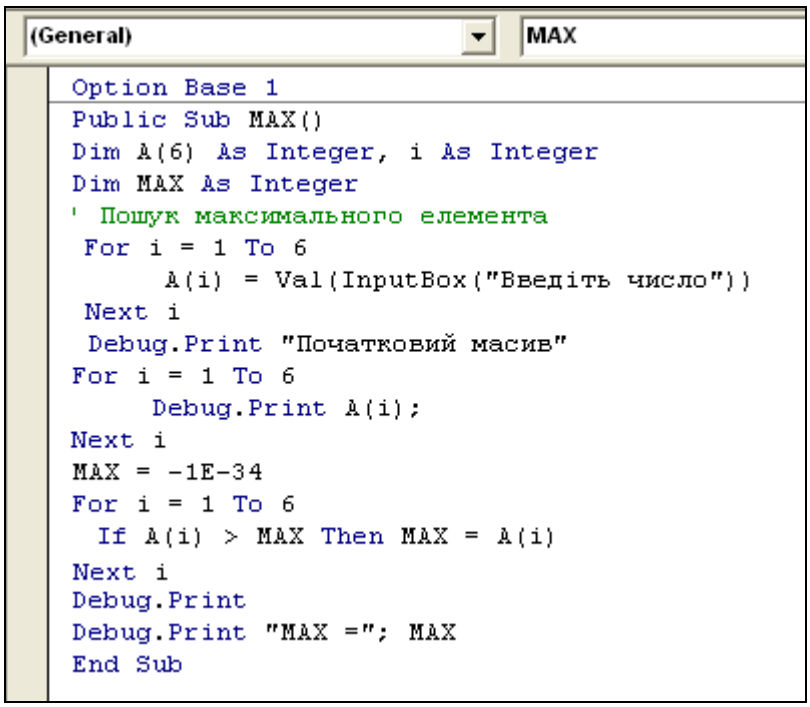


Рис.2.9 – Код програми

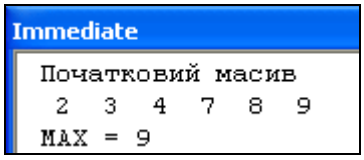


Рис.2.10 – Результат виконання програми

Як видно з наведеного алгоритму, значення змінної MAX («еталон») замінюється при першому порівнянні на перший елемент масиву A(1) і наступне порівняння вже відбувається з ним. Таким чином, в якості «еталона» можна використати і перший елемент масиву A(1). Тоді перебір елементів масиву починається з другого елемента.

```
MAX= A(1)
For i = 2 To 6
     If  A(i) > MAX Then  MAX = A(i)
Next i
```

Пошук мінімального елемента (min)

Для пошуку мінімального елемента масиву рекомендується до початку циклу задати початкове значення «еталона» у вигляді дуже великого числа, а потім вже у циклі робити порівняння.

Приклад:

В якості «еталона» обираємо дуже велике число – (1E38). Збережемо це значення в змінній MIN (MIN = 1E38 - «еталон»).

Фрагмент програми.

```
MIN = 1E38
For i = 1 To 6
      If  A(i) < MIN Then MIN = A(i)
Next i
```

Аналогічно в якості «еталона» можна використати і перший елемент масиву A(1).

Фрагмент програми.

«Еталон» це перший елемент множини.

```
MIN = A(1)
For i = 2 To 6
      If A(i) < MIN Then MIN = A(i)
Next i
```

Для того, щоб винайти не тільки значення максимуму (мінімуму) в масиві, а і його місцезнаходження (позиція елемента в масиві) потрібно обрати змінну (або змінні при багатовимірному масиві) для зберігання координати, і при зміні «еталона» одночасно змінювати відповідні координати (тобто запам'ятовувати координати числа, що присвоюється «еталону».

Приклад:

Фрагмент програми пошуку мінімального елемента і його позиції в масиві.

```
MIN = A(1)
Index = 1  ' позиція мінімуму – одиниця
```

```
For i = 2 To 6
   If A(i) < MIN Then
MIN = A(i)
Index = i
   End If
Next i
```

Пошук значень кратних деякому числу

Зазвичай для вирішення задач такого класу використовують наступні стандартні математичні функції:

1. Функція **Fix(N)**. Повертає цілу частину числа N. Функція Fix не округляє число, а відкидає будь-яку дробову частину. Якщо N є негативним числом, то функція Fix повертає найближче негативне ціле число, що більше або рівне числу N.
2. Функція **Int(N)**. Повертає цілу частину числа N. Функція Int не округляє число, а відкидає будь-яку дробову частину. Якщо N є негативним числом, то функція Int повертає найближче негативне ціле число, рівне числу N або не перевищує його.

Іншими словами можна сказати, ціла частина числа залишається, а дробова відкидається.

Також можна використати і арифметичні оператори:

1. \ – цілочисельне ділення ($N_1 \setminus N_2$). Ділить N_1 на N_2 і відкидає будь-яку дробову частину так, щоб результат був цілим числом.
2. Mod – ділення по модулю (N_1 Mod N_2). Ділить N_1 на N_2 і повертає тільки залишок операції ділення.

Число X кратне деякому числу m, якщо залишок операції ділення дорівнює нулю. Тому, якщо X Mod m = 0, то число X кратне числу m (а, якщо не дорівнює нулю, – то не кратне). При використанні цілочисельного ділення та функцій Int і Fix необхідно порівняти результат звичайного ділення X/m з результатом цілочисельного ділення X\m (або результатом виконання функцій Fix(X/m) або Int(X/m)). Якщо різниця цих результатів дорівнює

нулю, то число *X* кратне числу *m*, а якщо ні, то число *X* не кратне числу m. При обробці елементів масиву треба цю перевірку виконати почергово для кожного елемента масиву.

Приклад:

В масиві з 9 елементів знайти елементи кратні 5.

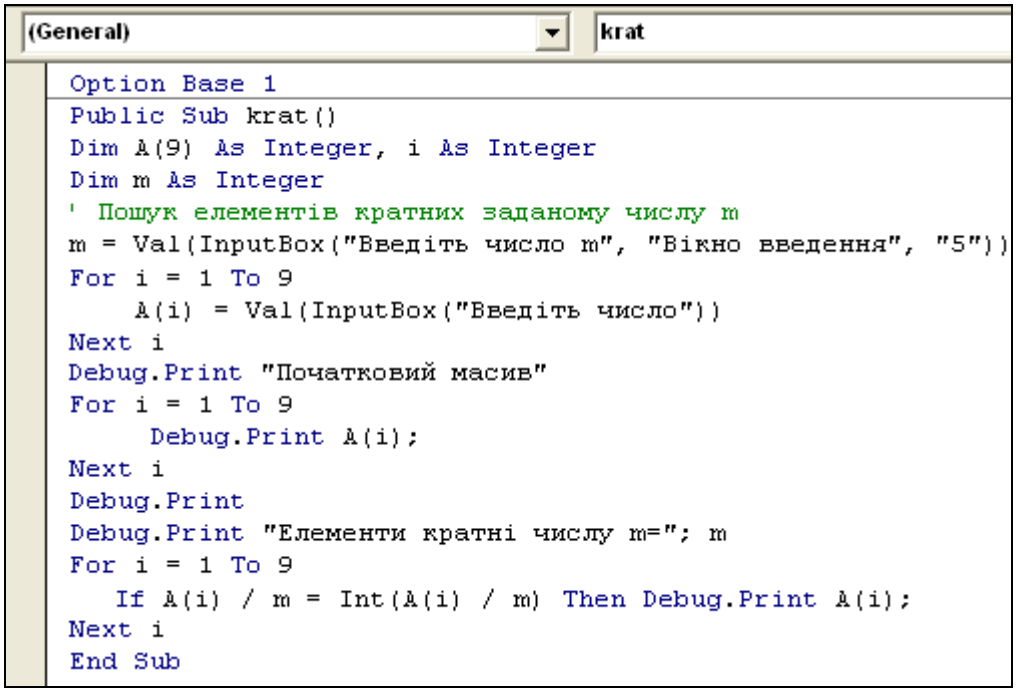


Рис.2.11 – Код програми

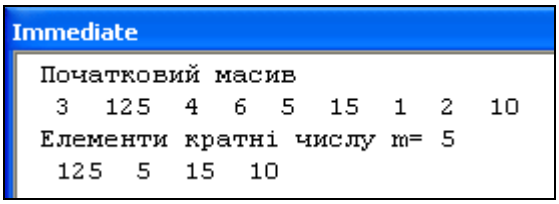


Рис.2.12 – Результат виконання програми

Аналогічно будуть працювати функція Fix, оператор Mod і оператор цілочисельного ділення.

Наприклад:

Фрагмент програми із застосуванням функції Fix:

```
For i = 1 To 9
  If A(i) / m = Fix(A(i) / m) Then Debug.Print A(i);
Next i
```

Фрагмент програми із застосуванням оператора цілочисельного ділення:

```
For i = 1 To 9
  If A(i) / m = A(i) \ m Then Debug.Print A(i);
Next i
```

Фрагмент програми із застосуванням оператора ділення по модулю Mod:

```
For i = 1 To 9
  If A(i) Mod m = 0 Then Debug.Print A(i);
Next i
```

Якщо завданням пошуку є пошук елементів не кратних деякому числу, то описану вище умову слід перевіряти на нерівність нулю або порівнювати результати ділення.

Наприклад:

Фрагмент програми знаходження елементів не кратних деякому числу m із застосуванням функції Int:

```
Debug.Print "Елементи не кратні числу m="; m
For i = 1 To 9
    If A(i) / m <> Int(A(i) / m) Then Debug.Print A(i);
Next i
```

Пошук від’ємних значень

Для пошуку від’ємних елементів масиву треба виконати почергово для кожного елемента масиву перевірку знака числа.

Фрагмент програми знаходження від’ємних елементів масиву:

```
For i = 1 To 9
    If A(i) < 0 Then Debug.Print A(i);
Next i
```

Пошук значень, що належать заданому інтервалу

При вирішені такої задачі необхідно виконати почергово для кожного елемента масиву перевірку на належність числа до заданого інтервалу.

Фрагмент програми для знаходження елементів масиву, що належать інтервалу [4, 11.7]:

```
f = 4
d = 11.7
For i = 1 To 9
    If A(i) > f And A(i) < d Then Debug.Print A(i);
Next i
```

Фрагмент програми для знаходження елементів масиву, що не перевищують 7.5:

```
For i = 1 To 9
    If A(i) <= 7.5 Then Debug.Print A(i);
Next i
```

В розглянутих прикладах ті елементи в масиві, які задовольняли задану умову, виводились на друк. Однак ці елементи можна записати в новий масив для подальшої обробки. При створенні нового масиву слід правильно визначити його розмірність і потім записати в нього елементи, що задовольняють задану умову пошуку. Для послідовного запису елементів масиву слід використати лічильник. Початкове значення такого лічильника дорівнює нулю (наприклад, $k=0$). Коли потрібний елемент знайдено, лічильник змінюється на одиницю ($k= k+1$) і на позицію k в новий масив записується знайдений елемент.

Якщо умов пошуку декілька, то для кожного з чисел виконується група перевірок.

Приклад:

В масиві А з 10 елементів знайти від’ємні елементи кратні 3 і записати їх в новий масив В.

```
(General)                                    krat
Option Base 1
Public Sub krat()
Dim A(10) As Integer, i As Integer
Dim m As Integer, B(10) As Integer
Dim k As Integer
' Пошук від'ємних елементів кратних заданому числу m
m = Val(InputBox("Введіть число m", "Вікно введення", "3"))
For i = 1 To 10
    A(i) = Val(InputBox("Введіть число"))
Next i
Debug.Print "Початковий масив A"
For i = 1 To 10
     Debug.Print A(i);
Next i
Debug.Print
k = 0
For i = 1 To 10
    If A(i) < 0 And A(i) / m = Int(A(i) / m) Then
     k = k + 1
     B(k) = A(i)
    End If
Next i
Debug.Print "Отриманий масив B"
For i = 1 To k
     Debug.Print B(i);
Next i
End Sub
```

Рис.2.13 – Код програми

```
Immediate
Початковий масив A
 3 -12  4  6  5 -15  1  2  10 -33
Отриманий масив B
-12 -15 -33
```

Рис.2.14 – Результат виконання програми

Впорядкування масивів (сортування)

Процес сортування. Полягає в зміні порядку розташування елементів множини відповідно до якоїсь ознаки або закономірності. Для виконання сортування елементів масиву потрібно виконати відповідну перестановку його елементів. Власне перестановка полягає в обміні значень двох елементів масиву. Обмін значеннями двох елементів масиву виконується за допомогою

проміжної третьої змінної, яка застосовується для зберігання одного з елементів масиву.

Наприклад:

Слід поміняти місцями два елементи масиву x(i) і x(к). Введемо проміжну змінну R і виконаємо наступну послідовність операторів:

```
R=x(i) ' зберігання значення x(i) в змінній R
x(i)=x(k) ' заміна значення x(i) на значення x(к)
x(k)=R ' заміна значення x(к) на значення x(i) збережене в змінній R.
```

Задачі сортування, що найчастіше зустрічаються, полягають в розташуванні елементів заданої послідовності за зростанням (збільшенням значень елементів масиву) і спаданням (зменшенням значень елементів масиву). Як правило процесу сортування передує процес пошуку.

Алгоритмів рішення подібних задачі, як і їх програмних реалізацій може бути достатньо велика кількість. Зупинимося на найбільш вживаних.

Алгоритм сортування – принцип максимуму або мінімуму

Наприклад, слід розташувати елементи масиву в порядку спадання.

В цьому випадку на першому місці буде найбільший елемент, на другому місці – найбільший елемент з тих, що залишились і так далі. Це означає, що можна визначити максимальний елемент і поставити його на перше місце. Потім з тих елементів, що залишилися, знову визначити максимальний елемент і поставити його на друге місце і так далі. При сортуванні за зростанням спочатку слід визначити мінімальний елемент, поставити його на перше місце, а далі все аналогічно сортуванню за спаданням. Таким чином, при виконанні сортування рух відбувається від першого елемента до останнього. При іншій реалізації алгоритму можна поставити потрібні елементи в кінець масиву і рухатись при подальшому пошуку в напрямку першого елемента.

Приклад:

Нехай задано масив A(5), що містить елементи {8,12,21,3,115}. Розташувати елементи масиву А в порядку спадання.

(General) sort

```
Option Base 1
Public Sub sort()
     Dim A() As Integer, i As Integer
Dim N As Integer
Dim k As Integer, R As Integer
N = Val(InputBox("Введите розмірність масива N="))
ReDim A(N) As Integer
For i = 1 To N
     A(i) = Val(InputBox("Введіть число"))
Next i
Debug.Print "Початковий масив"
For i = 1 To N
     Debug.Print A(i);
Next i
' Сортування за спаданням
k = 1
Do
  m = A(k)
  Index = k
  For i = k To N
       If A(i) > m Then
         m = A(i)
         Index = i
       End If
  Next i
  If Index <> k Then
  R = A(Index)
  A(Index) = A(k)
  A(k) = R
  End If
  k = k + 1 ' Зміщаємось до другого елемента
Loop While k < N
Debug.Print
Debug.Print "Відсортований масив"
For i = 1 To N
     Debug.Print A(i);
Next i
End Sub
```

Рис.2.15 – Код програми

Immediate

```
Початковий масив
 8   12   21   3   115
Відсортований масив
 115   21   12   8   3
```

Рис.2.16 – Результат виконання програми

Аналогічним чином вирішується задача впорядкування масиву за зростанням (збільшенням елементів). Тільки у цьому випадку треба було б здійснити пошук мінімального елементу і поставити його на перше місце і так далі.

В наведеній програмі сортували масив з 5 елементів {8,12,21,3,115}. Пошук у введеному масиві здійснювався при такий зміні параметра циклу *i*:

від 1 до 5 і після перестановок отримали – 115 12 21 3 8

від 2 до 5 і після перестановок отримали – 115 21 12 3 8

від 3 до 5 і після перестановок отримали – 115 21 12 8 3

від 4 до 5 і після перестановок отримали – 115 21 12 8 3

З наведених даних видно, що хоча і після третього пошуку всі елементи масиву вже були відсортовані в порядку спадання, був виконаний і 4-й пошук максимального елемента при зміні параметра циклу від 4 до 5. Таким чином, наведений алгоритм виконує більше процедур пошуку, чим це потрібно. Навіть у випадку, коли би всі елементи масиву з самого початку були б розташовані в порядку спадання, все одно було б виконано стільки ж процедур пошуку.

Алгоритм сортування – “метод бульбашки”

Один з алгоритмів, що використовується для вирішення задач сортування, відомий як “метод бульбашки”. Він полягає в тому, що послідовно порівнюються два сусідніх елемента масиву і вони міняються місцями, якщо їх розташування не відповідає необхідному порядку. Ця процедура повторюється до тих пір, поки “усі пари сусідів” не будуть задовольняти задану умову сортування. Для програмного визначення моменту закінчення сортування вводять лічильник перестановок, якому надають початкове значення нуль (наприклад, *fl=0*). При кожній виконаній перестановці лічильник перестановок змінюється (наприклад, *fl=1* або *fl = fl+ 1*). Процедура сортування закінчується, якщо перестановки були відсутні і лічильник перестановок не змінив своє значення (*fl=0*).

Приклад:

Нехай задано масив A(5), що містить елементи {8,12,21,3,115}. Розташувати елементи масиву А в порядку спадання.

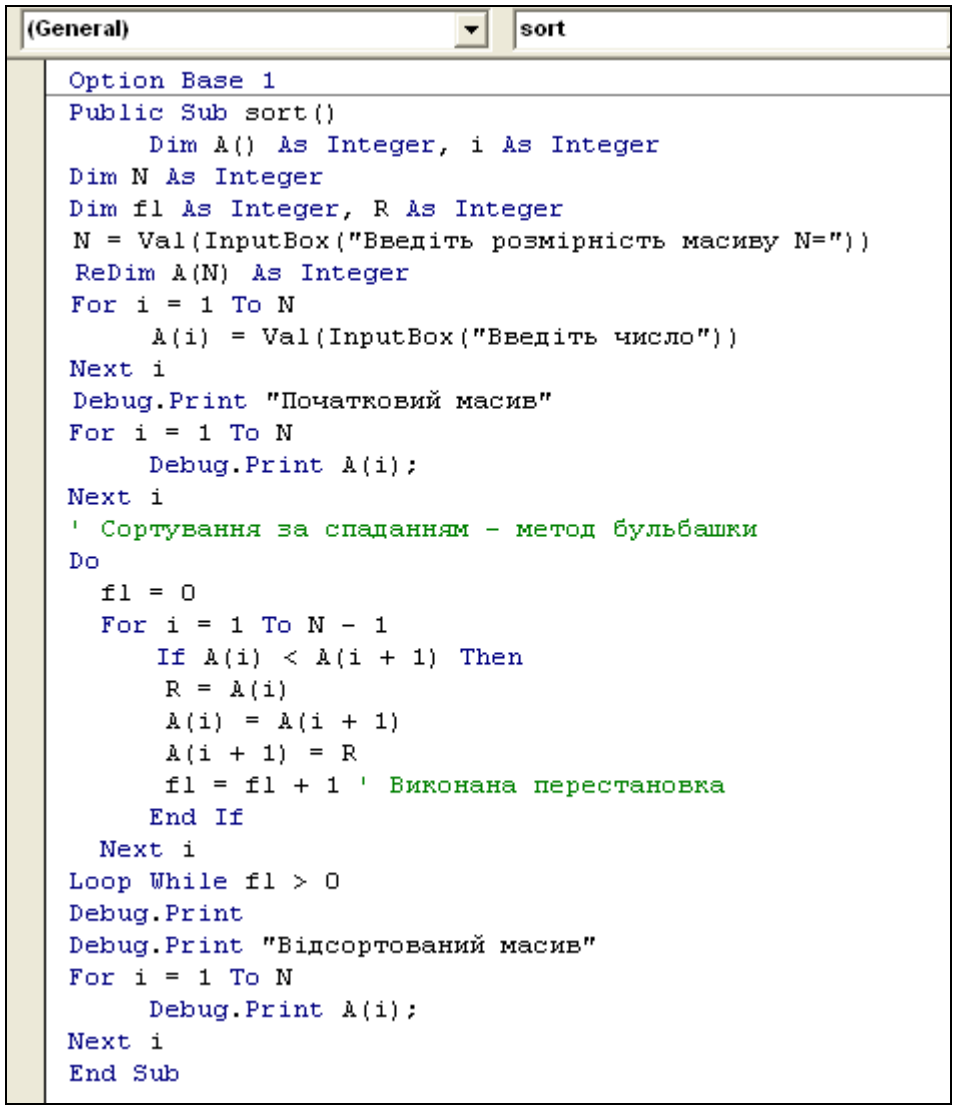


Рис.2.17 – Код програми

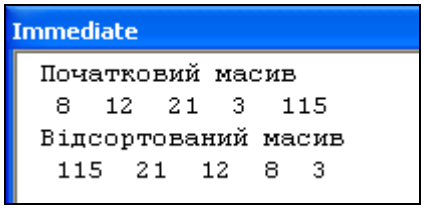


Рис.2.18 – Результат виконання програми

Якщо перед програмним рядком Loop While fl > 0 записати оператори виведення результатів проміжного сортування у вигляді наступних програмних рядків:

```
Debug.Print "Кількість перестановок ="; fl,
For j = 1 To N
     Debug.Print A(j);
Next j
```

то побачимо і проміжні результати сортування.

Початковий масив

```
8   12   21   3   115
```

Кількість перестановок =3 12 21 8 115 3

Кількість перестановок =2 21 12 115 8 3

Кількість перестановок =1 21 115 12 8 3

Кількість перестановок =1 115 21 12 8 3

Кількість перестановок =0 115 21 12 8 3

Відсортований масив

```
115   21   12   8   3
```

Процеси злиття

Під процесом злиття звичайно розуміють утворення деякої нової множини з усіх (або деяких) елементів раніше існуючих множин. Якщо в нову множину включаються всі елементи початкових множин (відбувається просте злиття декількох множин), то розмір нової множини визначається, як сумарний розмір існуючих множин. А якщо в нову множину включаються елементи початкових множин, що підбираються за якоюсь ознакою (чи ознаками), то розмір нової множини краще теж обирати, як сумарний розмір існуючих множин, бо в деяких випадках в нову множину можуть бути включені всі елементи існуючих множин. Але в кожному випадку треба ввести додаткову змінну (наприклад, *poz*), яка вказує позицію в новому масиві, на яку буде здійснено збереження елемента із початкової множини. Цій змінній надають початкове значення нуль (наприклад, *poz=0*), а потім перед записом змінюють її значення (наприклад, *poz = poz + 1*).

Приклад:

Створити новий масив *Z*, в записати від'ємні елементи двох масивів *X* і *Y*.

(General) st

```
Option Base 1
Public Sub st()
Dim X() As Integer, Y() As Integer
Dim Z() As Integer, i As Integer
Dim N As Integer, M As Integer
Dim poz As Integer
N = Val(InputBox("Введіть розмірність N масиву X"))
M = Val(InputBox("Введіть розмірність M масиву Y"))
ReDim X(N) As Integer
ReDim Y(M) As Integer
ReDim Z(N + M) As Integer
' Введення і друк значень масиву X
For i = 1 To N
    X(i) = Val(InputBox("Введіть число"))
Next i
Debug.Print "Початковий масив X' '
For i = 1 To N
    Debug.Print X(i);
Next i
' Введення і друк значень масиву Y
 For i = 1 To M
     Y(i) = Val(InputBox("Введіть число"))
 Next i
Debug.Print
Debug.Print "Початковий масив Y"
For i = 1 To M
    Debug.Print Y(i);
Next i
' Формування масиву Z
poz = 0
For i = 1 To N
     If X(i) < 0 Then
        poz = poz + 1
        Z(poz) = X(i)
    End If
Next i
For i = 1 To M
     If Y(i) < 0 Then
       poz = poz + 1
       Z(poz) = Y(i)
    End If
Next i
Debug.Print
Debug.Print "Сформований масив Z"
For i = 1 To poz
     Debug.Print Z(i);
Next i
End Sub
```

Рис.2.19 – Код програми

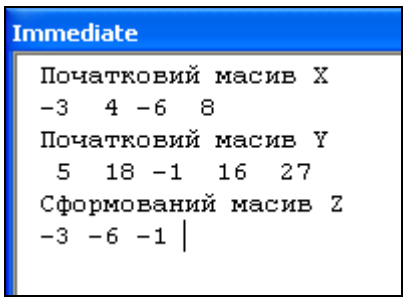


Рис.2.20 – Результат виконання програми

Алгоритми зміни кількості елементів у масиві

Часто виникає необхідність додавання нового елемента до вже існуючого масиву на задане місце (не обов'язково в кінець або на початок).

Необхідно пам'ятати, що збільшення розмірів масиву можливе тільки до розміру передбаченого в операторі *Dim*.

Нехай до масиву **x** який містить **n** елементів на **k**-те місце потрібно вставити новий елемент, що має значення **a** (тобто потрібно, щоб x**(k)=a)**. Якщо в програму помістити такий запис, то замість елемента, що знаходиться на **k**–тому місці буде записано значення **a**, а попередній елемент, що знаходився на **k**-тому місці буде «затертий» новим значенням і втрачений. Тоді, щоб виконати процедуру вставки алгоритмічно правильно, потрібно перші (**k**-1) елементів залишити без змін, а інші з – **k**-го по **n**-ий перемістити на один елемент вправо (перемістити вміст комірок пам'яті). І вже потім на «звільнене» **k**-те місце записати потрібний елемент (x**(k)=a**).

Щоб вміст жодного з елементів масиву не було втрачено, то зсув повинен здійснюватися від кінця до місця вставки. Тобто:

X(n+1)=x(n)

X(n)=x(n-1)

X(n-1)=x(n-2)

……..

X(k+1)=x(k)

А після виконання цього зсуву на **k** -те місце записати:

X(k)=a

Фрагмент програми для реалізації алгоритму вставки елемента в масив:

```
For I=N To k step -1
X(I+1)=X(I)
Next I
X(k)= a
```

Приклад:

В масив X на третю позицію вставити елемент, значення якого дорівнює 1000.

```
(General)                                   st
Option Base 1
Public Sub st()
Dim X() As Integer, i As Integer
Dim N As Integer, A As Integer
Dim poz As Integer
N = Val(InputBox("Введіть розмірність N масиву X"))
poz = Val(InputBox("На яку позицію відбудеться вставка"))
A = Val(InputBox("Яке число вставити"))
ReDim X(N) As Integer
' Введення і друк значень масиву X
For i = 1 To N
    X(i) = Val(InputBox("Введіть число"))
Next i
Debug.Print "Початковий масив X"
For i = 1 To N
   Debug.Print X(i);
Next i
ReDim Preserve X(N + 1) As Integer
' Зсув елементів масиву X
For i = N To poz Step -1
X(i + 1) = X(i)
Next i
X(poz) = A
Debug.Print
Debug.Print "Сформований масив"
For i = 1 To UBound(X)
  Debug.Print X(i);
Next i
End Sub
```

Рис.2.21 – Код програми

```
Immediate
Початковий масив X
 8   12   21   3   115
Сформований масив
 8   12   1000   21   3   115
```

Рис.2.22 – Результат виконання програми

При вирішенні задачі видалення **k**-го елемента з заданого масиву **x**, який містить **n** елементів необхідно на **k**-те місце записати **k+1** елемент, на **k+1** -те місце записати **k+2** елемент і так далі. Таким чином, зсув повинен виконуватися від місця видалення елемента (**k)** до кінця масиву. Тобто значення елементів з **(k+1)**-го по **n**-ий (вміст комірок пам'яті) здвигаються на один елемент вліво. При такому зсуві починаючи з позиції **k** наступні елементи «затирають» значення попередніх. Розмір масиву не зменшиться і **n-1** та **n**-ий елементи будуть мати однакові значення. Але після видалення елемента довжина масиву повинна зменшиться на 1. Цю проблему можна вирішити переоголошенням масиву за допомогою оператора ReDim (якщо масив динамічний) з використанням ключового слова Preserve (для зберігання значень попереднього масиву) або при обробці масиву задавати розмірність масиву меншу на одиницю.

Алгоритм видалення:

X(k)=x(k+1)

x(k+1)= x(k+2)

……

x(n-1)= x(n)

Фрагмент програми для реалізації алгоритму видалення елемента з масиву:

```
For I=K To N-1
 X(I)=X(I+1)
Next I
```

Приклад:

В масиві X видалити елемент, що знаходиться на третій позиції.

```
(General)                                   ts
Option Base 1
Public Sub ts()
Dim X() As Integer, i As Integer
Dim N As Integer, poz As Integer
N = Val(InputBox("Введіть розмірність N масиву X"))
poz = Val(InputBox("Позиція елементу для видалення"))
ReDim X(N) As Integer
' Введення і друк значень масиву X
For i = 1 To N
    X(i) = Val(InputBox("Введіть число"))
Next i
Debug.Print "Початковий масив X"
For i = 1 To N
     Debug.Print X(i);
Next i
' Видалення елемента з масиву X
For i = poz To N - 1
X(i) = X(i + 1)
Next i
ReDim Preserve X(N - 1) As Integer
Debug.Print
Debug.Print "Сформований масив"
For i = 1 To UBound(X)
   Debug.Print X(i);
Next i
End Sub
```

Рис.2.23 – Код програми

```
Immediate
 Початковий масив X
  8   12   21   3   115
 Сформований масив
  8   12   3   115
```

Рис.2.24 – Результат виконання програми

Примітка. В двовимірних масивах видалити або вставити можна цілий рядок або стовпець. Алгоритм такої дії аналогічний алгоритму видалення або вставки в одновимірних масивах. При цьому треба стежити, щоб двовимірний масив був оголошений належним чином (з запасом місця).

З метою більш поглибленого вивчення теоретичних основ даної лабораторної роботи рекомендується використати конспект лекцій з курсу та список рекомендованої літератури для даних методичних вказівок.

2.2 Опис лабораторних засобів та обладнання

Лабораторна робота виконується на персональному комп'ютері стандарту IBM PC під керуванням операційної системи MS Windows зі стандартним пакетом MS Office.

2.3 Заходи безпеки під час виконання лабораторної роботи

Заходи безпеки, яких треба дотримуватись при виконанні лабораторної роботи «Одновимірні масиви. Характерні прийоми програмування» наведені у Додатку А.

2.4 Послідовність виконання роботи

Згідно із отриманим завданням виконати наступне:

1. Вивчити поняття одновимірного масиву, прийоми роботи з одновимірними масивами (опис, введення та виведення масиву, операції з окремими елементами масиву).
2. Вивчити характерні прийоми програмування та типові алгоритми (пошук елементів масиву у відповідності з заданою ознакою: знак числа; кратність одного числа іншому; найменший або найбільший елемент; обчислення суми, добутку та інше).
3. У відповідності до свого варіанту завдання (Додаток В) розробити алгоритм та написати програму для обробки одновимірного масиву, що складається з 15 – 20 довільних дійсних чисел у відповідності до свого варіанту завдання. Передбачити друк введеного масиву та отриманих результатів. Для розрахунку використати довільні значення елементів масиву (можна скористатися датчиком випадкових чисел для заповнення масиву або ввести масив з клавіатури).
4. Набрати, відредагувати та налаштувати програму в середовищі VBA.
5. Продемонструвати роботу програми викладачу.

6. Оформити протокол лабораторної роботи. Алгоритм задачі навести у вигляді блок-схеми.

2.5 Оброблення та аналізування результатів. Оформлення звіту

При оформленні звіту з лабораторної роботи до заздалегідь підготованого протоколу (див. завдання до самостійної роботи) додаються роздруковані аркуші з результатами виконаної роботи та рисунок блок-схеми алгоритму.

Контрольні запитання

1. Дайте визначення поняття «масив». Як записати елемент масиву у середовищі програмування VBA?
2. Поняття «одновимірного масиву» та «двовимірного масиву».
3. Який оператор використовується для оголошення масивів у VBA?
4. Яке призначення у функцій LBound, Ubound?
5. Поняття статичного і динамічного масивів.
6. Який оператор використовується для очищення і видалення елементів масиву?
7. Як ввести елементи одновимірного масиву в VBA?
8. Як вивести елементи одновимірного масиву в VBA?
9. Алгоритми пошуку суми та добутку елементів одновимірного масиву.
10. Алгоритм сортування елементів одновимірного масиву за “методом бульбашки”.
11. Алгоритм сортування елементів одновимірного масиву за пошуком максимуму або мінімуму.
12. Що представляють собою процеси злиття даних у VBA?
13. Алгоритми зміни кількості елементів у масиві.

ЛАБОРАТОРНА РОБОТА № 3
ДВОВИМІРНІ МАСИВИ. ХАРАКТЕРНІ ПРИЙОМИ ПРОГРАМУВАННЯ

Мета та основні завдання роботи. Дослідити роботу циклічних структур для оброблення даних представлених у вигляді двовимірних масивів. Ознайомитися з основними прийомами введення-виведення і оброблення двовимірних масивів. Вивчити характерні прийоми програмування.

Вивчити та описати (стисло) в протоколі лабораторної роботи:

- поняття двовимірних масивів;
- прийоми введення-виведення та оброблення двовимірних;
- характерні прийоми програмування для двовимірних масивів.

3.1 Основні теоретичні відомості

Двовимірні масиви

До елемента двовимірного масиву звертаються за іменем і двом індексам, що записані у круглих дужках. Перший індекс ідентифікує рядок, а другий стовпець.

a_{11}	a_{12}	a_{13}	a_{14}
a_{21}	a_{22}	a_{23}	a_{24}
a_{31}	a_{32}	a_{33}	a_{34}
a_{41}	a_{42}	a_{43}	a_{44}

Досить часто виникають потреби оброблення елементів квадратної матриці (двовимірного масиву при рівній кількості рядків і стовпців), що знаходяться на головній або побічній діагоналях матриці. Якщо елементи матриці знаходяться на головній діагоналі, то індекси рядка і стовпця

співпадають – I = J. Якщо елементи матриці знаходяться на побічній діагоналі, то індекси рядка і стовпця зв'язані між собою співвідношенням: J = N +1 – I, де N – розмір квадратної матриці.

Для введення і виведення двовимірних масивів та виконання інших операцій з ними використовують конструкцію вкладені цикли. В зовнішньому циклі змінюють рядок (або стовпець), а у внутрішньому стовпець (або рядок) в залежності від потреб.

Наприклад:

```
For I=1 To 4
For J=1 To 4
     оператори введення/виведення/інші
Next J
Next I
```

В наведеній конструкції обробляється двовимірний масив розміру 4x4. Зовнішній цикл позначає 1 рядок (або стовпець) і обробляються всі його елементи, потім обробляється 2 рядок (або стовпець) і так далі.

Введення елементів двовимірного масиву

Найпростіший спосіб введення значень для елементів масиву з клавіатури за допомогою функції InputBox.

Приклад:

```
Option Base 1
Public Sub vvv()
Dim A(4,4) As Integer, i As Integer, j As Integer
' Заповнення масиву
For i = 1 To 4
      For j = 1 To 4
         A(i,j) = Val(InputBox("Введіть число", "Заповнення масиву"))
      Next j
Next i
' Повідомлення про закінчення введення
```

```
MsgBox ("Введення масиву закінчено")
End Sub
```

Також, як і одновимірний масив, двовимірний статичний масив можна заповнити випадковими числами.

Приклад:

Заповнити двовимірний масив випадковими числами в діапазоні від 0 до 100.

```
Option Base 1
Public Sub vvv()
Dim A(4,4) As Integer, i As Integer, j As Integer
' Заповнення масиву
          Randomize
For i = 1 To 4
      For j = 1 To 4
         A(i,j) = Int(Rnd * 100)
      Next j
Next i
' Повідомлення про закінчення введення
MsgBox ("Введення масиву закінчено")
End Sub
```

Процедура **Randomize** запускає датчик випадкових чисел, функція **Rnd** без аргументу повертає випадкове дійсне число з діапазону (0,1), а функція **Int( )** повертає цілу частину дійсного числа.

Коли на початку роботи з масивом невідома кількість елементів в ньому, або ж, в процесі роботи ця кількість може змінюватися, то в цьому випадку слід застосовувати динамічні масиви.

Приклад:

```
Option Base 1
Public Sub vvv()
Dim A() As Integer, i As Integer, j As Integer
Dim N As Integer, M As Integer
' Заповнення масиву
```

```
            N = Val(InputBox("Введіть кількість рядків N="))
            M = Val(InputBox("Введіть кількість стовпців M="))
  ReDim A(N,M) As Integer
   For i = 1 To N
        For j = 1 To M
           A(i,j) = Val(InputBox("Введіть число", "Заповнення масиву"))
        Next j
   Next i
  ' Повідомлення про закінчення введення
  MsgBox ("Введення масиву закінчено")
  End Sub
```

Також введення даних в двовимірний масив можна здійснити з файлу даних і з робочого листа Excel.

Виведення елементів масиву

Процедура виведення елементів масиву здійснюється з використанням операторів циклу. Зручно виводити елементи масиву у вікно Immediate за допомогою оператора Debug.Print.

Приклад:

```
For i = 1 To 4
    For j = 1 To 4
        Debug.Print A(i, j);
    Next j
  Debug.Print
Next i
```

Приклад виведення двовимірного масиву за допомогою наведеного фрагменту програми

```
 3  4  5  6
 1  3  6  7
 9  8  6  5
 3  5  6  8
```

Якщо не помістити між ключовими словами *Next j* і *Next i* оператор Debug.Print, то виведення масиву буде здійснюватись в один рядок (це не дає можливості побачити звичне представлення матриці у вигляді таблиці). Оператор Debug.Print, що розміщений після виконання внутрішнього циклу, в якому друкується перший рядок двовимірного масиву, виводить пустий рядок і тому при наступному виконанні внутрішнього циклу новий рядок двовимірного масиву буде друкуватись з нового рядка.

Якщо потрібно вивести тільки один (заданий) елемент двовимірного масиву, оператори циклу не потрібні і оператор Debug.Print записується в програмному рядку.

Приклад:

```
i = 2 : j = 3
Debug.Print  A(i, j)
Debug.Print A(1,1)
```

Перший оператор Debug.Print виводить елемент двовимірного масиву A(2,3), що знаходиться у другому рядку і третьому стовпці. а другий – елемент масиву A(1,1), що знаходиться у першому рядку і першому стовпці.

При використанні функції MsgBox для виведення елементів масиву потрібно спочатку сформувати повідомлення виводу (prompt), а після цього застосувати функцію MsgBox.

Приклад:

```
prompt = "" ' Початкове повідомлення - пустий рядок
For i = 1 To UBound(A,1)
       For j = 1 To UBound(A,2)
           prompt = prompt & A(i, j) & " "
       Next j
      prompt = prompt & Chr(13)
Next i
MsgBox prompt
```

Тут у внутрішньому циклі у повідомленні виводу – prompt формується перший рядок двовимірного масиву. Після його формування оператор присвоювання prompt = prompt & Chr(13) переносить виведення в наступний рядок і так далі (Chr(13) використовують для розділення рядків). Після формування повідомлення виводу – prompt у вікно буде виведений двовимірний масив.

Наприклад:

Рис.3.1 – Результат виконання програми

Також виведення даних з двовимірного масиву можна здійснити у файл даних і у робочий лист Excel.

Робота з двовимірними масивами

Для роботи з двовимірними масивами використовують циклічні структури (конструкція вкладені цикли). Елементи двовимірного масиву обробляють по рядкам (або стовпцям) в залежності від потреб.

Приклад:

У заданій матриці A(3,3) всі елементи піднести до кубу.

```
(General)                                    prim1
Option Base 1
Public Sub prim1()
Dim A(3, 3) As Integer, i As Integer, j As Integer
          'Введення масиву
For i = 1 To 3
      For j = 1 To 3
          A(i, j) = Val(InputBox("Введіть число"))
      Next j
Next i
Debug.Print "Початкова матриця"
For i = 1 To 3
    For j = 1 To 3
          Debug.Print A(i, j);
    Next j
    Debug.Print
Next i
For i = 1 To 3
    For j = 1 To 3
          A(i, j) = A(i, j) ^ 3
    Next j
Next i
Debug.Print "Отримана матриця"
For i = 1 To 3
   For j = 1 To 3
     Debug.Print A(i, j);
   Next j
  Debug.Print
Next i
End Sub
```

Рис.3.2 – Код програми

```
Immediate
Початкова матриця
 1   2   3
 4   5   6
 7   8   9
Отримана матриця
 1   8   27
 64   125   216
 343   512   729
```

Рис.3.3 – Результат виконання програми

Приклад:

У заданій матриці A(3,3) всі елементи другого рядка замінити на 1.

```
(General)                                   prim1
Option Base 1
Public Sub prim1()
Dim A(3, 3) As Integer, i As Integer, j As Integer
' Введненя масива
For i = 1 To 3
     For j = 1 To 3
         A(i, j) = Val(InputBox("Введіть число"))
     Next j
Next i
Debug.Print "Початкова матриця"
For i = 1 To 3
    For j = 1 To 3
         Debug.Print A(i, j);
    Next j
    Debug.Print
Next i
' Перетворення в матриці
For i = 1 To 3
    For j = 1 To 3
        If i = 2 Then A(i, j) = 1
    Next j
Next i
Debug.Print "Отримана матриця"
For i = 1 To 3
    For j = 1 To 3
     Debug.Print A(i, j);
    Next j
    Debug.Print
Next i
End Sub
```

Рис.3.4 – Код програми

```
Immediate
 Початкова матриця
  3   4   5
  6   5   3
  7   6   5
 Отримана матриця
  3   4   5
  1   1   1
  7   6   5
```

Рис.3.5 – Результат виконання програми

Фрагмент програми, що виконує перетворення в матриці можна спростити наступним чином (вказуванням номеру рядку і зміною в циклі тільки номерів стовпців):

```
For j = 1 To 3
     A(2, j)=1
```

```
  Next j
```

Приклад:

У заданій матриці A(3,3) всі елементи головної діагоналі замінити на 0.

```
(General)                                   prim1
Option Base 1
Public Sub prim1()
Dim A(3, 3) As Integer, i As Integer, j As Integer
' Введеняя масиву
For i = 1 To 3
      For j = 1 To 3
         A(i, j) = Val(InputBox("Введіть число"))
      Next j
Next i
Debug.Print "Початкова матриця"
For i = 1 To 3
    For j = 1 To 3
         Debug.Print A(i, j);
    Next j
    Debug.Print
Next i
' Перетворення в матриці
For i = 1 To 3
    For j = 1 To 3
        If i = j Then A(i, j) = 0
    Next j
Next i
Debug.Print "Отримана матриця"
For i = 1 To 3
    For j = 1 To 3
      Debug.Print A(i, j);
    Next j
    Debug.Print
Next i
End Sub
```

Рис.3.6 – Код програми

```
Immediate
 Початкова матриця
  1   2   4
  3   5   6
  7   8   6
 Отримана матриця
  0   2   4
  3   0   6
  7   8   0
```

Рис.3.7 – Результат виконання програми

Фрагмент програми, що виконує перетворення в матриці можна спростити наступним чином: зміною в циклі тільки номерів рядків, оскільки на головній діагоналі індекси рядка і стовпця рівні.

```
For i = 1 To 3
     A(i, i)=1
Next i
```

Характерні прийоми програмування

Характерні прийоми програмування розглянуті для одновимірних масивів мають антологічне застосування і для двовимірних масивів.

Приклад:

Фрагмент програми пошуку максимального елемента і його позицій в двовимірному масиві.

```
MAX= A(1,1))    ' максимум - перший елемент
IndexRow = 1      ' рядок максимуму - одиниця
IndexCol = 1      ' стовпець максимуму - одиниця
For i = 1 To 4
      For j = 1 To 4
           If  A(i,j) > MAX Then
              MAX = A(i,j)
              IndexRow = i
              IndexCol = j
           End If
      Next j
Next i
```

З метою більш поглибленого вивчення теоретичних основ даної лабораторної роботи рекомендується використати конспект лекцій з курсу та список рекомендованої літератури для даних методичних вказівок.

3.2 Опис лабораторних засобів та обладнання

Лабораторна робота виконується на персональному комп'ютері стандарту IBM PC під керуванням операційної системи MS Windows зі стандартним пакетом MS Office.

3.3 Заходи безпеки під час виконання лабораторної роботи

Заходи безпеки, яких треба дотримуватись при виконанні лабораторної роботи «Двовимірні масиви. Характерні прийоми програмування» наведені у Додатку А.

3.4 Послідовність виконання роботи

Згідно з отриманим завданням виконати наступне:

1. Вивчити поняття двовимірного масиву, прийоми роботи з двовимірними масивами (опис, введення та виведення двовимірних масивів, операції з окремими елементами масиву).
2. Вивчити характерні прийоми програмування та типові алгоритми (пошук елементів масиву у відповідності з заданою ознакою: знак числа; кратність одного числа іншому; найменший або найбільший елемент; обчислення суми, добутку та інше).
3. У відповідності до свого варіанту завдання (Додаток Д) розробити алгоритм та написати програму для оброблення двовимірного масиву. Передбачити друк введеного двовимірного масиву та отриманих результатів. Для розрахунку використати довільні значення елементів двовимірного масиву (можна скористатися датчиком випадкових чисел для заповнення масиву або ввести масив з клавіатури).
4. Створити, відредагувати та налаштувати програму в середовищі VBA.
5. Продемонструвати роботу програми викладачу.
6. Оформити протокол лабораторної роботи. Алгоритм задачі навести у вигляді блок-схеми (за вказівкою викладача).

3.5 Оброблення та аналізування результатів. Оформлення звіту

При оформленні звіту з лабораторної роботи до заздалегідь підготованого протоколу (див. завдання до самостійної роботи) додаються роздруковані аркуші з результатами виконаної роботи та рисунок блок-схеми алгоритму (за вказівкою викладача).

Контрольні запитання

1. Розкрийте поняття «двовимірний масив».
2. Як правильно записати елемент двовимірного масиву у VBA?
3. Як працюють вкладені цикли?
4. Як оголошують двовимірний масив у VBA?
5. Як можна ввести елементи двовимірного масиву у VBA?
6. Як можна вивести елементи двовимірного масиву у VBA?
7. Як за допомогою функцій LBound, UBound визначити нижню та верхню границі двовимірного масиву?
8. Поясніть для чого потрібен наведений програмний рядок

```
ReDim Preserve AB (3, 8)?
```

9. Як визначити суму та добуток елементів двовимірного масиву?
10. Як зв’язані індекси побічної діагоналі квадратної матриці?

ЛАБОРАТОРНА РОБОТА № 4
РОБОТА З ФАЙЛАМИ ДАНИХ

Мета та основні завдання роботи. Дослідити можливості обміну інформацією між програмою і файлами даних. Ознайомитися з основними прийомами введення-виведення інформації в файл.

Вивчити та описати (стисло) в протоколі лабораторної роботи:

– робота з файлами даних: отримання дескриптора файлу, відкриття файлу, читання або запис даних, закриття файлу;

– режими доступу до даних;

– відмінності роботи операторів Print# і Write#.

4.1 Основні теоретичні відомості

Робота з файлами даних

Зручним способом зберігання інформації є файли. Так у файлі можна зберігати як початкову інформацію, так і результати роботи програми. При цьому, початкова інформація може бути використана декількома програмами, а результати розрахунків, що записані в файл, можуть бути використані іншими програмами в якості початкової інформації.

У VBA реалізовано три типи доступу до файлів:

– послідовний – для читання і запису текстових файлів;

– довільний – для читання і запису тексту із записами структурованої довжини;

– двійковий (бінарний) – для читання і запису довільно структурованих файлів.

Файл послідовного доступу розглядається як послідовність рядків довільної довжини, розділених спеціальними символами. Читання і запис у файл проводиться порядково.

Файл довільного доступу складається із записів фіксованої довжини і розмір запису вказується при його відкритті. Це дозволяє локалізувати будь-який запис у файлі по його номеру.

Бінарний файл є окремим випадком файлу довільного доступу. Розмір запису в бінарному файлі вважається рівним 1 байту

Найчастіше використовується послідовний доступ. При цьому, дані у файлі зберігаються в неструктурованому вигляді.

Робота з файлами даних завжди складається з декількох етапів:

- отримання дескриптора файлу;
- відкриття файлу;
- читання або запис даних;
- закриття файлу.

Для того, щоб програма могла обмінюватися даними з якимось файлом його потрібно відкрити. Для цього використовують оператор відкриття – **Open** який організовує канал зв'язку між дисковим файлом і програмою.

Формат:

```
Open  "ім'я файлу" For < режим роботи > As [#] < номер каналу >
```

Ім'я файлу – визначає назву файлу, який необхідно відкрити. Ім'я файлу пишеться в лапках і вказує повний шлях до файлу, що відкривається – диск:\шлях\ім'я файлу (наприклад: "C:\MT\referat.txt").

Режим роботи – встановлює дозволений режим доступу до даних, які зберігаються в файлі. Для позначення режиму можливо застосувати наступні ключові слова:

– Input – файл відкривається лише для читання (для введення даних з дискового файлу в програму);

– Output – файл відкривається для запису (виведення даних з програми у дисковий файл);

– Append – файл відкривається лише для до запису (нове речення додається в кінець вже існуючого).

Номер каналу – це номер, по якому VBA ідентифікує відкритий файл. Він може бути будь-яким цілим числом в діапазоні від #1 до #255 і сприймається як номер файлу у всіх процедурах обміну. (У VBA допустиме використання номерів 1-511 при різних варіантах доступу до файлів).

Файл, що відкривається для читання (For Input), потрібно створити і записати на диск до початку роботи програми. Якщо такого файлу не існує і робиться спроба для його відкриття в програмі, то VBA видає повідомлення про помилку. При **відкритті файлів для запису** або додавання (For Output або For Append), VBA створює новий файл. Якщо файл з вказаним ім'ям існує, то в режимі Output його вміст видаляється, а в режимі Append файл відкривається для додавання записів (дописування).

Якщо в програмі відкрита велика кількість файлів і важко визначити які з каналів вільні (довго і уважно передивлятись програмний код великої програми), то можна обрати вільний канал, який можна використовувати для відкриття файлів, за допомогою функції **FreeFile.**

Формат:

```
FreeFile[(RangeNumber)]
```

FreeFile повертає номер вільного каналу, який можна використовувати для роботи з файлом. Якщо вільних каналів немає, то виникає помилка.

Аргумент функції RangeNumber (ДіапазонНомерів) дозволяє визначити діапазон значень, з якого вибирається черговий вільний номер каналу. Якщо його значення дорівнює 0 (за умовчанням), то повертається номер каналу з діапазону 1 – 255, якщо 1, то з діапазону 256 – 511.

Така операція має назву – **отримання дескриптора файлу**.

Приклад:

```
NF=FreeFile
```

В даному прикладі змінній NF присвоюється ціле значення, яке можна використовувати для відкриття файлу. Тоді, оператор Open можна записати:

```
Open "D:\USERS\ DATA\TEXT.TXT" For Input As NF
```

В кінці рядка вказано номер каналу, що був повернутий функцією **FreeFile**.

Після того як обмін даними з файлом закінчений, канал зв'язку необхідно закрити. Це виконується оператором **Close**.

Формат:

```
Close [ [# <номер каналу1>] [, [# <номер каналу2>] …]
```

Якщо цей оператор написати без всяких аргументів, він закриє всі відкриті файли. Якщо треба закрити файл, що обмінювався інформацією по каналу номер 2, то слід записати **Close #2**.

Приклади:

Close – закриє всі відкриті файли.

Close # 3,# 5 – закриє файли з каналами обміну 3,5.

….

F1=FreeFile : F2=FreeFile : F3=FreeFile

….

Close F1, F2, F3 – закриє файли з каналами обміну F1, F2, F3, які були повернуті функцією FreeFile.

Читання або запис даних

Доступ до файлу можливий в середині тексту програми між операторами Open і Close.

Введення даних (зчитування) із файлу

Введення даних здійснюється оператором Input. Він прочитує дані з відкритого файлу послідовного доступу і присвоює їх змінним, що вказані в його списку.

Формат:

```
Input # <номер каналу> , <список введення>
```

або

```
Input # <номер каналу>, V1 [, V2] [, V3]…
```

де V1, V2, V3… – це список введення змінних, яким слід присвоїти значення, прочитані з файлу. Змінні в списку розділяються комами.

Кількість змінних в списку оператора Input, їх тип і порядок запису повинні відповідати характеру і послідовності типів даних, що записані в файлі. Основний принцип – читати як записував.

Для читання з файлу рядкових даних можна також використовувати:

– **оператор Line Input #**;

Прочитує рядок символів з відкритого файлу послідовного доступу і присвоює її одній змінній типа String або Variant .

Формат:

```
Line Input #<номер каналу>, Ім'я змінної
```

Результатом роботи цього оператора є присвоєння Змінній значення – всього чергового текстового рядка файлу.

– **функцію Input**;

Функція Input – повертає значення типа String, що містить вказане число символів, що зчитані з файлу, відкритого в режимі Input або Binary.

Формат:

```
Input(<Число Символів >, # <номер каналу>)
```

Число Символів – це кількість символів, які треба прочитати з вхідного файлу. Число Символів задає число символів, що повертається. Якщо аргумент **Число Символів** рівний 1, то проводиться посимвольне читання даних. Функція повертає текст у вигляді символьного рядка.

Виведення (запис) даних в файл

Виведення даних здійснюється оператором Print# або оператором Write#.

Формат:

```
Print #<номер каналу> [, <список виведення>]
```

або

```
Print #<номер каналу>, [P1] [{ ; | ,} P2] ……
```

де P1, P2 – список змінних, значення яких виводяться в файл (записуються у файл даних).

При використанні оператора Print елементи списку значень в цьому операторі мають бути розділені або крапкою з комою, або комою. Від цього залежить, як вони будуть записані в текстовий рядок файлу:

– значення записуватимуться підряд, без проміжків між ними (роздільник крапка з комою);

– значення записуватимуться в 14-символьні зони виводу(роздільник кома).

Крім того, в списку значень оператора можуть бути присутніми функції:

§ Spc(n) − для вставки п пропусків між значеннями в текстовому рядку;

§ Tab(n) − для вказівки номера позиції *n* рядка для запису наступного значення.

У список виведення можуть бути включені вирази будь-якого типу (числові або символьні).

В файл послідовного доступу можна записувати дані і оператором **Write**. Його робота аналогічна роботі оператора Print.

Формат:

```
Write#<номер каналу> [, <список виведення>]
Write#<номер каналу>, [P1] [, P2] ……
```

де P1, P2 – список змінних, значення яких виводяться до файлу (записуються у файл даних).

При використанні оператора Write роздільником в списку виведення є кома. Елементи цього списку записуються в один текстовий рядок файлу. На відміну від оператора Print, оператор Write вставляє коми між елементами списку виведення. Елементи типа String записуються в лапках. Після запису останнього елементу списку виведення записується символ переходу на новий рядок.

Дані, що записані в послідовний файл за допомогою оператора Write#, зазвичай легко можна прочитати за допомогою оператора Input#.

Якщо список виведення в операторах відсутній, то в файл буде записано пустий рядок.

Створення файлу вихідних даних

Файл вихідних (початкових) даних обов’язково повинен бути створений до початку роботи програми, яка зчитує з нього дані (спроба відкрити неіснуючий файл приведе до помилки).

Дані записуються в рядках і розділяються комою (або проміжком). Наприклад, початкові дані, що записані в файлі data.txt попереднього прикладу можна записати, як показано нижче.

3,4	3 4
1,2,5,6	1 2 5 6
2,4,7,9	2 4 7 9
3,6,4,5	3 6 4 5

При цьому дані, які читаються “наступним” оператором введення Input #, записуються обов’язково з нового рядка. А дані, котрі відносяться до одного оператора введення, можуть записуватися як в одному так і в декількох рядках.

Файл початкових даних може бути створений за допомогою текстового редактора (наприклад, Блокнот).

Файл, призначений для запису результатів програми, заздалегідь створювати не потрібно. Він створюється системою автоматично при використанні режиму роботи Outpput при відкритті файлу.

Приклад 1:

Нехай задано в файлі послідовного доступу data.txt, що збережений на диску D записані значення змінних A, B, C (8,12,21). Прочитати ці дані, виконати необхідні розрахунки і результати роботи програми записати в новий файл –rez.txt.

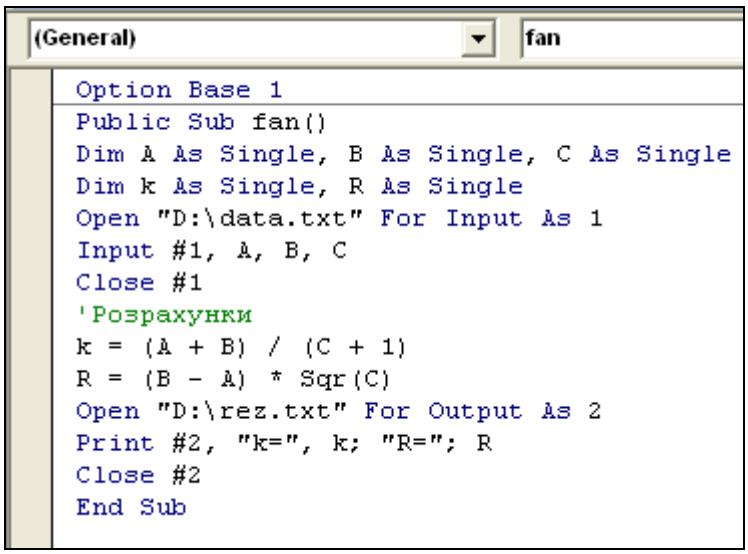


Рис.4.1 – Код програми

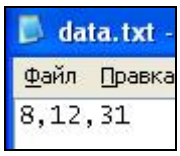


Рис.4.2 – Початкові дані, що записані в файлі data.txt

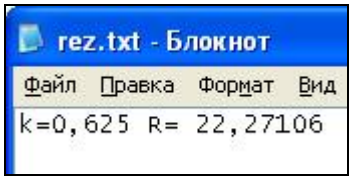


Рис.4.3 – Результати, що записані в файлі rez.txt

Якщо для запису в новий файл використати оператор Write (Write #2, "k=", k, "R=", R), то результати будуть записані наступним чином:

"k=",.625,"R=",22.27106

Як видно з наведеного приклада, виведення символу «,» в файл оператором Write не завжди корисне.

Приклад 2:

Задано цілочисельний масив A з 10 елементів. В новий масив B записати квадрати елементів масиву A. Підрахувати суму елементів масиву B. Початковий масив ввести з файлу, а отриманий масив і суму його елементів записати в новий файл.

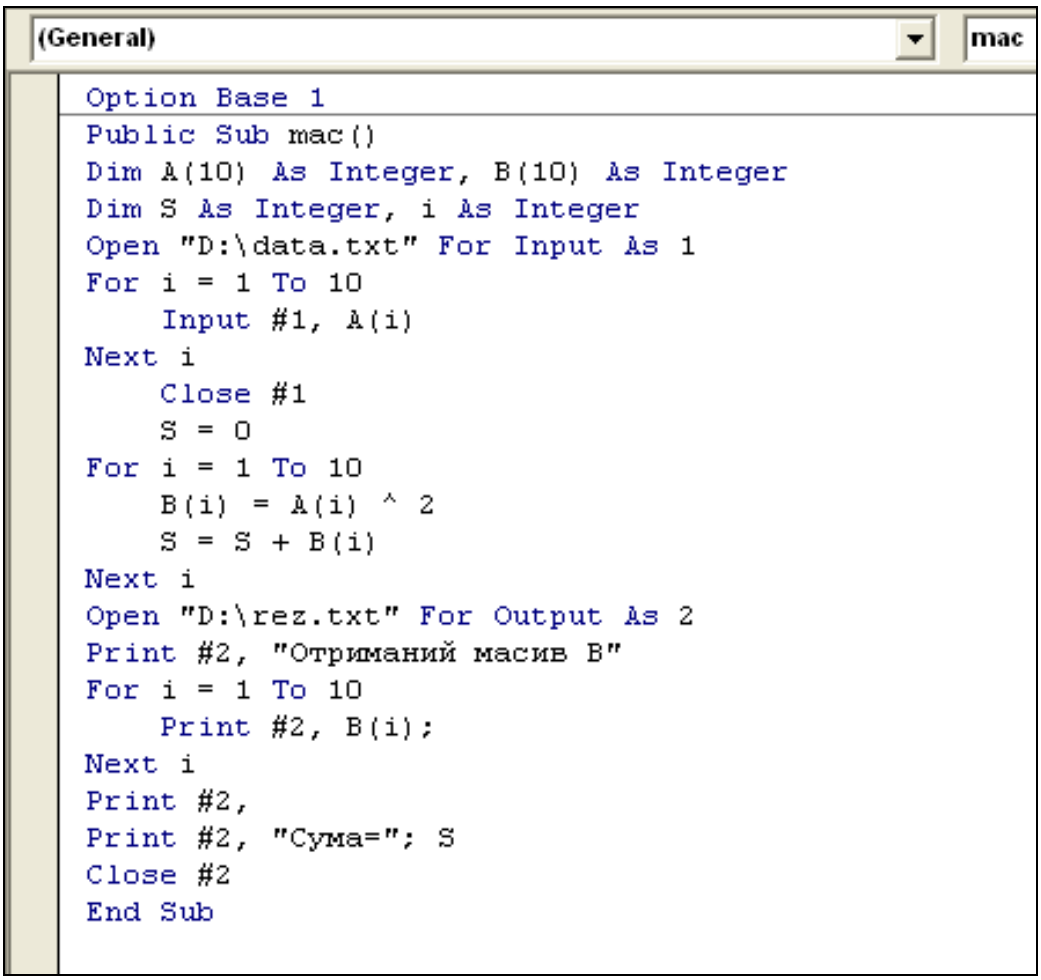


Рис.4.4 – Код програми

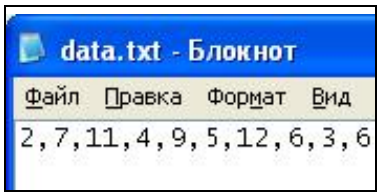


Рис.4.5 – Початкові дані, що записані в файлі data.txt

Рис.4.6 – Результати, що записані в файлі rez.txt

Якщо для запису масиву B в новий файл використати оператор Write:

```
For i = 1 To 10
```

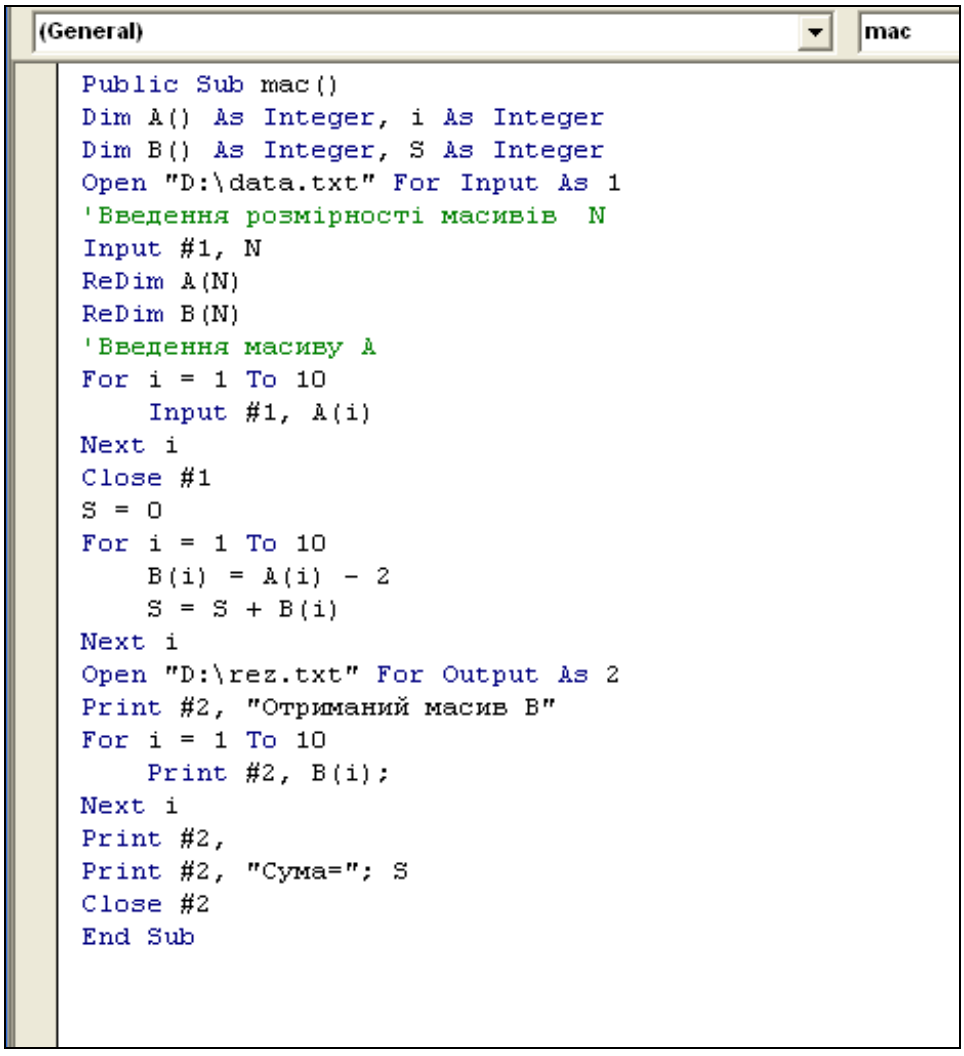
```
    Write #2, B(i),
Next i
```

Отримаємо:

```
Отриманий масив B
4,49,121,16,81,25,144,36,9,36,
```

Приклад 3:

Задано довільний цілочисельний масив А. В новий масив В записати елементи масиву А зменшені на 2. Підрахувати суму елементів масиву В. Початковий масив і його розмірність ввести з файлу, а отриманий масив і суму його елементів записати в новий файл.




```
Public Sub mac()
Dim A() As Integer, i As Integer
Dim B() As Integer, S As Integer
Open "D:\data.txt" For Input As 1
'Введення розмірності масивів  N
Input #1, N
ReDim A(N)
ReDim B(N)
'Введення масиву A
For i = 1 To 10
    Input #1, A(i)
Next i
Close #1
S = 0
For i = 1 To 10
    B(i) = A(i) - 2
    S = S + B(i)
Next i
Open "D:\rez.txt" For Output As 2
Print #2, "Отриманий масив B"
For i = 1 To 10
    Print #2, B(i);
Next i
Print #2,
Print #2, "Сума="; S
Close #2
End Sub
```

Рис.4.7 – Код програми

Рис.4.8 – Початкові дані, що записані в файлі data.txt

Рис.4.9 – Результати, що записані в файлі rez.txt

Приклад 4:

Задано довільний цілочисельний двовимірний масив A. Масив і його розмірність ввести з файлу, а потім записати його в новий файл.

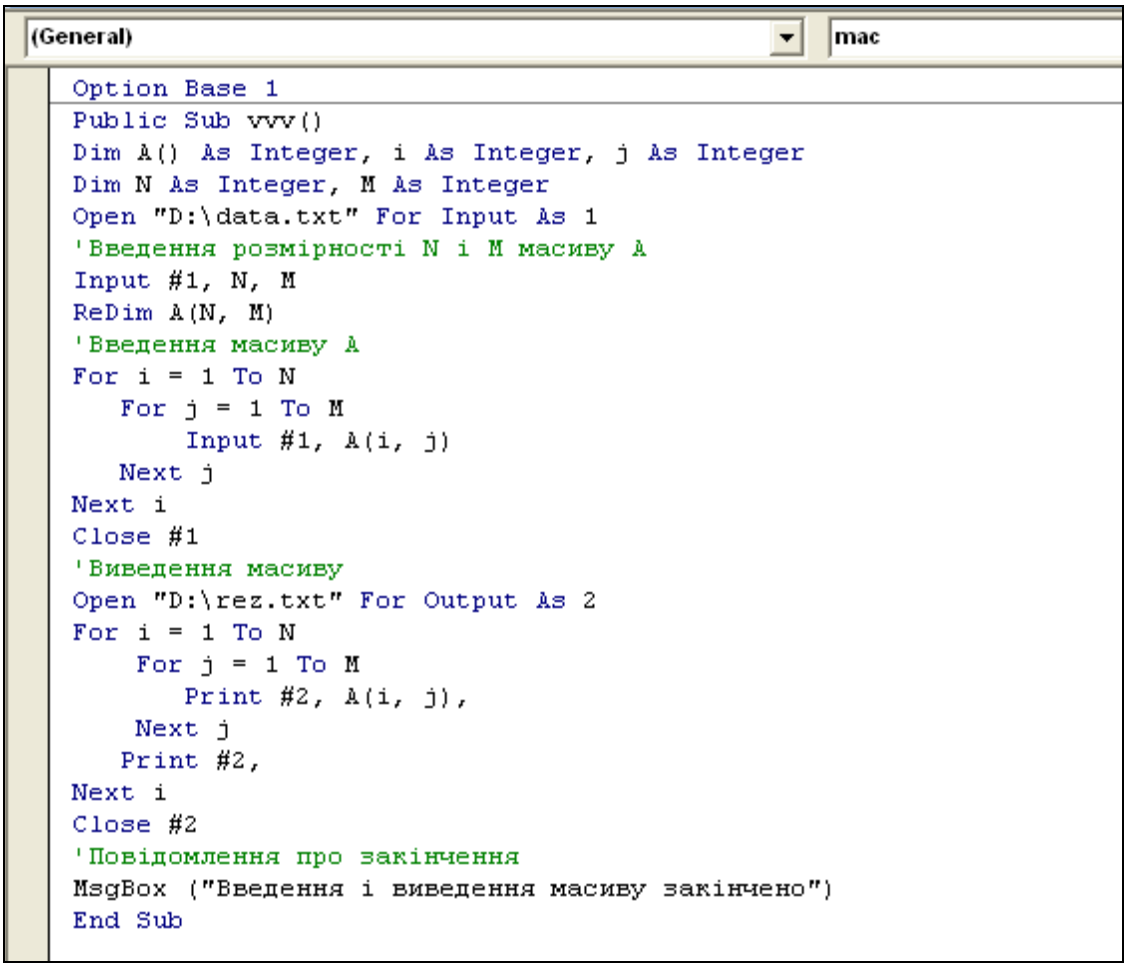


Рис.4.10 – Код програми

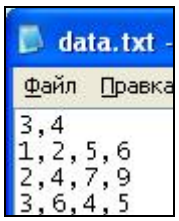


Рис.4.11 – Початкові дані, що записані в файлі data.txt

Рис.4.12 – Результати, що записані в файлі rez.txt

З метою більш поглибленого вивчення теоретичних основ даної лабораторної роботи рекомендується використати конспект лекцій з курсу та список рекомендованої літератури для даних методичних вказівок.

4.2 Опис лабораторних засобів та обладнання

Лабораторна робота виконується на персональному комп'ютері стандарту IBM PC під керуванням операційної системи MS Windows зі стандартним пакетом MS Office.

4.3 Заходи безпеки під час виконання лабораторної роботи

Заходи безпеки, яких треба дотримуватись при виконанні лабораторної роботи «Робота з файлами даних » наведені у Додатку А.

4.4 Послідовність виконання роботи

Згідно з отриманим завданням виконати наступне:

1. Вивчити організацію роботи з файлами даних: отримання дескриптора файлу; відкриття файлу; читання і запис даних; закриття файлу.
2. Розглянути створення і запис файлу для введення вихідних даних.

3. У відповідності до свого варіанту завдання (Додаток Е) розробити програми для оброблення одновимірного і двовимірного масивів. Введення значень одновимірного і двовимірного масивів проводити із заздалегідь сформованих файлів даних. Організувати виведення інформації у файл. Продублювати виведення інформації у файл виведенням на екран.
4. Створити, відредагувати та налаштувати програми в середовищі VBA.
5. Програму зберегти на диску в папці своєї групи. Створити документ Microsoft Word, в який скопіювати текст програми і додати отримані результати роботи програми.
6. Продемонструвати роботу програм та отримані файли даних викладачу.

4.5 Оброблення та аналізування результатів. Оформлення звіту

При оформленні звіту з лабораторної роботи до заздалегідь підготованого протоколу (див. завдання до самостійної роботи) додаються роздруковані аркуші з результатами виконаної роботи.

Контрольні запитання

1. Що таке файл послідовного доступу?
2. Як відкрити дисковий файл у VBA?
3. Які дозволені режими доступу до даних у VBA?
4. Як позначається номер каналу обміну у VBA і як можна визначити вільний канал?
5. Який формат запису зчитування даних із файлу у VBA?
6. Які правила створення файлу з даними?
7. Поясніть особливості зчитування рядкових даних із файлу у VBA.
8. Який формат запису даних у файл в VBA?
9. Поясніть особливості застосування операторів Print# та Write# у VBA?

10. Для чого використовуються у списку значень оператора виведення даних у файл функції Spc(n) та Tab(n)?
11. Як впливають на запис даних в текстовий рядок файлу роздільники крапкою з комою або кома?

Список рекомендованої літератури

1. Харрис М. Освой самостоятельно программирование MS Excel 2000 для за 21 день. – М.: Вильямс (SAMS), 2000. – 880 с.
2. Орвис В. Excel для ученых, инженеров и студентов. – К.: Юниор, 1999. – 528 с.
3. Гарнаев А. Самоучитель VBA. Технология создания пользовательских приложений. – СПб.: BHV, 1999. – 512 с.
4. Microsoft Visual Basic 6.0 для профессионалов. Шаг за шагом: Практ. пособие./Пер. с англ. – М.: Издательство ЭКОМ, 2001. – 720 с.
5. Уокенбах, Джон Профессиональное программирование на VBA в Excel 2002.: Пер.сангл. – М.:Издательский дом «Вильямс» , 2003. –784с.
6. Уокенбах, Джон Профессиональное программирование на VBA в Excel 2003.: Пер.сангл. – М.:Издательский дом «Вильямс» , 2005. –800с.
7. Культин Н. Б. Visual Basic. Освой на примерах. – СПб.: БХВ-Петербург, 2004. – 288 с.
8. VisualBasic.NET: учебный курс / В.Долженков, М.Мозговой. – СПб.: Питер, 2003. –464 с.
9. John Walkenbach, Excel 2010 Power Programming with VBA.–Published by Wiley Publishing, Inc., Indianapolis, Indiana Published simultaneously in Canada, 2010. – p. 1007.

Додаток А. Заходи безпеки під час виконання лабораторних робіт

Цикл лабораторних робіт з дисципліни «Обчислювальна математика та програмування» виконуються в комп'ютерному класі кафедри кібернетики хіміко-технологічних процесів хіміко-технологічного факультету, де розміщені персональні комп'ютери. 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 хіміко-технологічного факультету

1. 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.
2. Всі роботи в учбових лабораторіях кафедри кібернетики ХТП проводяться лише з дозволу викладача або співробітника кафедри.
3. 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.
4. При роботі на ПЕОМ треба пам'ятати, що в них використовується напруга, що небезпечна для життя.

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

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

7. 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.

8. 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.

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

10. 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 в 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 заходи пожежної безпеки у лабораторіях, учбових та робочих приміщеннях кафедри кібернетики хіміко-технологічних процесів хіміко-технологічного факультету

1. Всі студенти повинні знати та ретельно виконувати «Загальні правила пожежної безпеки в КПІ ім. Ігоря Сікорського».
2. Завідуючий кафедрою та завідуючий лабораторією відповідають за забезпечення пожежної безпеки всіх приміщень кафедри та за справність протипожежного обладнання та сигналізації.
3. Все електричне обладнання, яке знаходиться в лабораторіях та приміщеннях кафедри, повинно мати заземлення.
4. В усіх приміщеннях повинно дотримуватись чистоти, не займати приміщення непотрібними меблями, обладнанням та матеріалами.
5. Всі двері основних та додаткових виходів утримувати у стані швидкого відкривання.
6. Зберігання та використання горючих та легкозаймистих рідин у приміщеннях кафедри забороняється.
7. Ремонт електричного обладнання проводити у строгій відповідності з правилами пожежної безпеки.
8. Всі електрозахисти повинні знаходитися у закритому положенні, не займаними сторонніми предметами.
9. Коридори, проходи, тамбури, евакуаційні виходи та підходи до першочергових засобів пожежогасіння, а також комунікаційні ніші повинні бути постійно вільними, чистими та нічим не зайнятими.
10. 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.

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

12. Студенти повинні знати та ретельно виконувати «Загальні правила техніки безпеки в КПІ ім. Ігоря Сікорського», 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. Варіанти завдання до лабораторної роботи №1

1. $S \approx 1 + 2x + \frac{4x^2}{2!} + \frac{8x^3}{3!} + \ldots = e^{2x} \qquad |x| < \infty$

2. $S \approx 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \ldots = e^{-x} \qquad |x| \le \infty$

3. $S \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots = \sin(x) \qquad |x| \le \infty$

4. $S \approx 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots = \cos(x) \qquad |x| \le \infty$

5. $S \approx x + \frac{x^3}{3!} + \frac{x^5}{5!} + \frac{x^7}{7!} + \mathbf{K} \quad = sh(x) \qquad |x| \le \infty$

6. $S \approx 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \mathbf{K} \quad = ch(x) \qquad |x| \le \infty$

7. $S \approx x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots = arctg(x) \qquad |x| \le 1$

8. $S \approx x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots = \ln(1+x) \qquad -1 < x \le 1$

9. $S \approx -x - \frac{x^2}{2} - \frac{x^3}{3} - \frac{x^4}{4} - \cdots \quad = \ln(1-x) \quad -1 \le x < 1$

10. $S \approx 1 - x + x^2 - x^3 + \mathbf{K} \quad = \frac{1}{1+x} \qquad |x| < 1$

11. $S \approx 1 + x + x^2 + x^3 + \ldots = \frac{1}{1-x} \qquad |x| < 1$

12. $S \approx 2\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} + \mathbf{K} \right) = \ln\frac{1+x}{1-x} \qquad |x| < 1$

13. $S \approx 1 - 2x + 3x^2 - 4x^3 + 5x^4 - \mathbf{K} = \frac{1}{(1+x)^2} \quad |x| < 1$

14. $S \approx 1 - x^2 + x^4 - x^6 + x^8 - \mathbf{K} = \frac{1}{1+x^2} \qquad |x| < 1$

15. $S \approx 1 - \frac{2 \cdot 3}{2}x + \frac{3 \cdot 4}{2}x^2 - \frac{4 \cdot 5}{2}x^3 + \cdots = \frac{1}{(1+x)^3} \qquad |x| < 1$

16. $S \approx 1 + \frac{1}{x} + \frac{1}{2!x^2} + \frac{1}{3!x^3} + \ldots = e^{1/x} \qquad |x| \le 1$

17. $S \approx \frac{1}{x} + 1 + \frac{x}{2!} + \frac{x^2}{3!} + \mathbf{K} = \frac{e^x}{x} \qquad |x| < 1$

18. $S \approx 1-\frac{1}{4}\cdot x+\frac{1}{4}\cdot\frac{5}{8}\cdot x^2-\frac{1}{4}\cdot\frac{5}{8}\cdot\frac{9}{12}\cdot x^3+\mathbf{K}=(1+x)^{-1/4} \qquad |x|<1$

19. $S \approx 1+\frac{1}{4}x+\frac{1}{4}\cdot\frac{5}{8}\cdot x^2+\frac{1}{4}\cdot\frac{5}{8}\cdot\frac{9}{12}x^3+\ldots=(1-x)^{-1/4} \quad |x|<1$

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

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

22. $S \approx 1-\frac{1}{2}\cdot x+\frac{1\cdot 3}{2\cdot 4}\cdot x^2-\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\cdot x^3+\ldots=(1+x)^{-\frac{1}{2}} \qquad |x|<1$

23. $S \approx 1+\frac{1}{2}\cdot x+\frac{1\cdot 3}{2\cdot 4}\cdot x^2+\frac{1\cdot 3\cdot 5}{2\cdot 4\cdot 6}\cdot x^3\,\mathbf{K}=(1-x)^{-\frac{1}{2}} \quad |x<1|$

24. $S \approx 1-\frac{3}{2}\cdot x+\frac{3\cdot 5}{2\cdot 4}\cdot x^2-\frac{3\cdot 5\cdot 7}{2\cdot 4\cdot 6}\cdot x^3+\mathbf{K}=(1+x)^{-3/2} \qquad |x|<1$

25. $S \approx 1+\frac{3}{2}\cdot x+\frac{3\cdot 5}{2\cdot 4}\cdot x^2+\frac{3\cdot 5\cdot 7}{2\cdot 4\cdot 6}\cdot x^3+\mathbf{K}=(1-x)^{-3/2} \qquad |x|<1$

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

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

28. $S \approx \frac{1}{x}-\left(\frac{x}{3}+\frac{x^3}{45}+\frac{2x^5}{945}+\frac{x^7}{4725}+\ldots\right)=ctg(x) \qquad (0<|x|<p)$

29. $S \approx 2\left(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+\frac{1}{7x^7}+\mathbf{K}\right)=\ln\frac{x+1}{x-1} \qquad (|x|>1)$

30. $S \approx -\frac{x^2}{2}-\frac{x^4}{12}-\frac{x^6}{45}-\frac{17x^8}{2520}-\ldots=\ln|\cos(x)| \qquad (|x|<\frac{p}{2})$

31. $S \approx 1+3x+\frac{9x^2}{2!}+\frac{27x^3}{3!}+\ldots=e^{3x} \qquad |x|<\infty$

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

33. $S \approx 1+2x+3x^2+4x^3+5x^4+\mathbf{K}=\frac{1}{(1-x)^2} \quad |x|<1$

34. $S \approx 1+2x+4x^2+8x^3+\ldots=\frac{1}{1-2x} \qquad |x|<1/2$

35. $S \approx 1+3x+9x^2+27x^3+\ldots=\dfrac{1}{1-3x} \qquad |x|<1/3$

36. $S \approx 1+\dfrac{x}{2}+\dfrac{x^2}{4}+\dfrac{x^3}{8}+\ldots=\dfrac{1}{1-\frac{1}{2}x} \qquad |x|<1$

37. $S \approx 1+\dfrac{x}{5}+\dfrac{x^2}{25}+\dfrac{x^3}{125}+\ldots=\dfrac{1}{1-\frac{1}{5}x} \qquad |x|<1$

38. $S \approx 1+\dfrac{x}{3}+\dfrac{x^2}{9}+\dfrac{x^3}{27}+\ldots=\dfrac{1}{1-\frac{1}{3}x} \qquad |x|<1$

Додаток В. Варіанти завдання до лабораторної роботи №2

1. В довільному масиві ***A*** з ***n*** елементів визначити окремо суми та кількості додатних та від'ємних елементів.
2. В довільному масиві ***Z*** з ***n*** елементів знайти найменший з парних елементів.
3. В довільному масиві ***T*** з ***n*** елементів знайти найбільший з елементів, що кратні трьом.
4. В довільному масиві ***S*** з ***n*** елементів знайти суму та кількість від'ємних елементів, кратних чотирьом.
5. В довільному масиві ***E*** з ***n*** елементів знайти суму та кількість елементів, значення яких не менше заданої константи ($E_i \geq a$).
6. В довільному масиві ***T*** з ***n*** елементів знайти найбільший з непарних елементів.
7. В довільному масиві ***T*** з ***n*** елементів знайти найменший елемент, що не кратний трьом.
8. В довільному масиві ***A*** з ***n*** елементів визначити окремо добутки та кількості додатних та від'ємних елементів.
9. В довільному масиві ***V*** з ***n*** елементів знайти найбільший та найменший елементи і визначити їх позиції в масиві.
10. В довільному масиві ***V*** з ***n*** елементів знайти кількість елементів, що дорівнюють нулю та добуток інших.
11. В довільному масиві ***B*** з ***n*** елементів знайти суму непарних елементів та добуток парних елементів.
12. В довільному масиві ***E*** з ***n*** елементів знайти добуток та кількість елементів, значення яких не менше заданої константи ($E_i \geq a$).
13. В довільному масиві ***B*** з ***n*** елементів знайти середнє значення B_{cp}, надрукувати елементи, значення яких більше середнього: $B_i > B_{cp}$ та знайти їх суму.

14. В довільному масиві $\boldsymbol{G}$ з $\boldsymbol{n}$ елементів знайти суму та кількість додатних елементів, що кратні п’яти.
15. В довільному масиві $\boldsymbol{Q}$ з $\boldsymbol{n}$ елементів знайти суму та кількість елементів, значення яких знаходяться в межах заданого інтервалу: $a \le Q_i \le b$.
16. В довільному масиві $\boldsymbol{B}$ з $\boldsymbol{n}$ елементів ($\boldsymbol{n}$ – парне число) знайти суму елементів, що знаходяться в першій половині масиву, і найбільший з елементів другої половини масиву.
17. В довільному масиві $\boldsymbol{W}$ з $\boldsymbol{n}$ елементів знайти суму та кількість від’ємних елементів, що не кратні чотирьом.
18. В двох заданих масивах $\boldsymbol{A}$ з $\boldsymbol{n}$ елементів і $\boldsymbol{B}$ з $\boldsymbol{m}$ елементів знайти додатні елементи і записати їх в новий масив $\boldsymbol{W}$, в якому знайти суму елементів.
19. В довільному масиві $\boldsymbol{H}$ з $\boldsymbol{n}$ елементів ($\boldsymbol{n}$ – парне число) знайти найменший елемент в першій половині масиву, і найбільший елемент в другій половині масиву.
20. В довільному масиві $\boldsymbol{U}$ з $\boldsymbol{n}$ елементів знайти найбільший та найменший елементи серед парних елементів масиву.
21. В довільному масиві $\boldsymbol{U}$ з $\boldsymbol{n}$ елементів знайти окремо кількість додатних, від’ємних елементів і елементів, значення яких дорівнює нулю.
22. В довільному масиві $\boldsymbol{Q}$ з $\boldsymbol{n}$ елементів знайти добуток елементів, значення яких знаходяться в межах заданого інтервалу: $a \le Q_i \le b$, а інші елементи масиву замінити на одиниці.
23. В довільному масиві $\boldsymbol{W}$ з $\boldsymbol{n}$ елементів знайти найбільший та найменший елементи серед від’ємних елементів масиву.
24. В довільному масиві $\boldsymbol{L}$ з $\boldsymbol{n}$ елементів знайти суму та кількість додатних елементів, що кратні п’яти .
25. В довільному масиві $\boldsymbol{F}$ з $\boldsymbol{n}$ елементів знайти суму та добуток елементів, значення яких не перевищують заданої константи: $F_i \le a$.
26. В двох заданих масивах $\boldsymbol{A}$ з $\boldsymbol{n}$ елементів і $\boldsymbol{B}$ з $\boldsymbol{m}$ елементів знайти додатні елементи і записати їх в новий масив $\boldsymbol{W}$, в якому знайти найменший елемент.

27. В довільному масиві $\boldsymbol{R}$ з $\boldsymbol{n}$ елементів знайти найменший елемент з елементів, що кратні чотирьом, та найбільший елемент з елементів, що кратні п’яти.
28. В довільному масиві $\boldsymbol{B}$ з $\boldsymbol{n}$ елементів знайти суму від’ємних елементів та добуток додатних елементів.
29. В довільному масиві $\boldsymbol{Q}$ з $\boldsymbol{n}$ елементів знайти суму елементів, значення яких знаходяться в межах заданого інтервалу: $a \le Q_i \le b$, а інші елементи масиву замінити на нулі.
30. В довільному масиві $\boldsymbol{W}$ з $\boldsymbol{n}$ елементів знайти найбільший та найменший елементи та додати до них одиницю.
31. В двох заданих масивах $\boldsymbol{A}$ з $\boldsymbol{n}$ елементів і $\boldsymbol{B}$ з $\boldsymbol{m}$ елементів знайти додатні елементи і записати їх в новий масив $\boldsymbol{W}$, в якому знайти найбільший елемент.
32. В довільному масиві $\boldsymbol{S}$ з $\boldsymbol{n}$ елементів знайти парні елементи та замінити їх значення двійками, а потім в перетвореному масиві знайти добуток елементів.
33. В довільному масиві $\boldsymbol{Q}$ з $\boldsymbol{n}$ елементів знайти окремо добутки елементів, що кратні трьом, чотирьом і п’яти.
34. В довільному масиві $\boldsymbol{G}$ з $\boldsymbol{n}$ елементів знайти найменший елемент серед елементів, значення яких знаходяться в межах заданого інтервалу: $a \le Q_i \le b$, і знайти суму інших елементів масиву.
35. В довільному масиві $\boldsymbol{A}$ з $\boldsymbol{n}$ елементів знайти суму та кількість додатних елементів, значення яких не перевищують заданої константи: $F_i \le a$.
36. В двох заданих масивах $\boldsymbol{A}$ з $\boldsymbol{n}$ елементів і $\boldsymbol{B}$ з $\boldsymbol{m}$ елементів знайти додатні елементи і записати їх в новий масив $\boldsymbol{W}$, в якому знайти добуток елементів.
37. В довільному масиві $\boldsymbol{R}$ з $\boldsymbol{n}$ елементів знайти окремо суми елементів, що кратні трьом, чотирьом і п’яти.

38.В довільному масиві ***U*** з ***n*** елементів знайти кількість елементів значення яких дорівнює нулю і окремо знайти суми додатних і від’ємних елементів.

39..В довільному масиві ***S*** з ***n*** елементів знайти від’ємні елементи, замінити їх одиницями та підрахувати кількість замін.

40.В довільному масиві ***W*** з ***n*** елементів знайти від’ємні елементи і піднести їх до квадрату, а потім у перетвореному масиві знайти суму елементів.

Додаток Д. Варіанти завдання до лабораторної роботи №3

1. Задано довільний цілочисельний двовимірний масив A(n, n). Знайти суму елементів масиву, а також суму елементів другого рядка масиву.
2. Задано довільний двовимірний масив Y(m, n). Знайти суму, кількість та середнє значення всіх елементів значення яких не менше заданої константи ($Yij \geq \alpha$).
3. Задано довільний двовимірний масив A(n, n). Знайти добуток елементів масиву, а також суму елементів третього стовпця масиву.
4. Задано довільний цілочисельний двовимірний масив D(m, n). Знайти найбільший і найменший елементи масиву та їх позиції.
5. Задано довільний двовимірний масив U(m, n). Знайти суму та кількість від'ємних елементів масиву.
6. Задано довільний двовимірний масив A(n, n). Знайти суму елементів масиву, а також суму елементів головної діагоналі масиву.
7. Задано довільний двовимірний масив P(m, n). Знайти добутки елементів всіх рядків масиву.
8. Задано довільний двовимірний масив H(m, n). Знайти суми елементів всіх стовпців масиву.
9. Задано довільний двовимірний масив U(m, n). Знайти найбільші елементи всіх рядків масиву.
10. Задано довільний двовимірний масив A(n, n). Знайти найменші елементи всіх стовпців масиву.
11. Задано довільний цілочисельний двовимірний масив D(m, n). Знайти парні елементи масиву та їх суму, а також − парні елементи третього рядка масиву, які надрукувати в окремому рядку.
12. Задано довільний цілочисельний двовимірний масив P(m, n). Знайти непарні елементи масиву та їх добуток, а також − непарні елементи другого стовпця масиву, які надрукувати в окремому рядку.

13. Задано довільний двовимірний масив U(m, n). Знайти добутки елементів непарних рядків масиву.
14. Задано довільний двовимірний масив A(n, n). Знайти суму елементів масиву, а також добуток елементів побічної діагоналі масиву.
15. Задано довільний двовимірний масив U(m, n). Знайти середні арифметичні значення елементів кожного стовпця.
16. Задано довільний цілочисельний двовимірний масив A(n, n). Знайти суми елементів, що кратні п’яти, в кожному рядку масиву.
17. Задано довільний двовимірний масив U(m, n). Знайти найбільші елементи непарних рядків масиву.
18. Задано довільний цілочисельний двовимірний масив U(n, n). Знайти добутки елементів, що кратні трьом, в кожному стовпці масиву.
19. Задано довільний двовимірний масив H(m, n). Знайти окремо суми елементів другого і третього стовпців масиву.
20. Задано довільний цілочисельний двовимірний масив D(m, n). Знайти суму і добуток елементів, що знаходяться на перетині парних рядків і непарних стовпців масиву.
21. Задано довільний двовимірний масив H(m, n). Знайти окремо найменші елементи другого і третього стовпців масиву.
22. Задано довільний двовимірний масив U(m, n). Знайти суми елементів непарних стовпців масиву.
23. Задано довільний двовимірний масив U(m, n). Знайти окремо найбільші елементи другого і четвертого рядків масиву.
24. Задано довільний двовимірний масив A(n, n). Знайти суму та добуток елементів головної діагоналі масиву.
25. Задано довільний двовимірний масив U(m, n). Знайти найменші елементи парних стовпців масиву.
26. Задано довільний двовимірний масив A(n, n). Знайти суму та добуток елементів побічної діагоналі масиву.

27. Задано довільний двовимірний масив D(m, n). Знайти суму і добуток елементів, що знаходяться на перетині непарних рядків і парних стовпців масиву.
28. Задано довільний двовимірний масив U(m, n). Знайти середні арифметичні значення елементів кожного рядка.
29. Задано довільний двовимірний масив H(m, n). Знайти в кожному рядку кількість елементів, що кратні п’яти.
30. Задано довільний двовимірний масив P(m, n). Знайти позиції найбільшого і найменшого елементів масиву.
31. Задано довільний двовимірний масив U(m, n). Знайти суми парних елементів непарних рядків масиву.
32. Задано довільний двовимірний масив Y(m, n). Знайти окремо кількість додатних, від’ємних елементів і елементів, значення яких дорівнює нулю.
33. Задано довільний двовимірний масив P(m, n). Знайти окремо суми додатних і від’ємних елементів масиву.
34. Задано довільний двовимірний масив P(m, n). Знайти окремо суми парних і непарних елементів масиву.
35. Задано довільний двовимірний масив H(m, n). В кожному парному стовпці знайти суми елементів масиву, а в непарному – добутки елементів масиву.
36. Задано довільний двовимірний масив H(m, n). В кожному парному рядку знайти добутки елементів масиву, а в непарному – суми елементів масиву.
37. Задано довільний двовимірний масив Y(m, n). Знайти суму від’ємних елементів масиву і записати ці елементи в новий одновимірний масив.
38. Задано довільний двовимірний масив D(m, n). Знайти добуток елементів масиву, значення яких знаходяться в межах заданого інтервалу: $\alpha \le Dij \le \beta$, і записати ці елементи в новий одновимірний масив.
39. Задано довільний двовимірний масив D(m, n). Знайти добуток елементів масиву, значення яких не менше заданої константи ($Dij \ge \alpha$), і записати ці елементи в новий одновимірний масив.

40. Задано довільний двовимірний масив D(m, n). Знайти добуток елементів масиву, значення яких не перевищують заданої константи (Dij ≤ α), і записати ці елементи в новий одновимірний масив.

Додаток Е. Варіанти завдання до лабораторної роботи № 4

№	Завдання№1	Завдання№2
1	В довільному масиві **A** з **8** елементів визначити суму додатних елементів.	Задано довільний цілочисельний двовимірний масив **A(2, 4)**. Знайти максимальний елемент з елементів масиву.
2	В довільному масиві **A** з **10** елементів визначити добуток від'ємних елементів.	Задано довільний цілочисельний двовимірний масив **A(3, 2)**. Знайти мінімальний елемент з елементів масиву.
3	В довільному масиві **A** з **11** елементів визначити найбільший елемент масиву.	Задано довільний цілочисельний двовимірний масив **A(5, 3)**. Визначити суму додатних елементів масиву.
4	В довільному масиві **A** з **12** елементів визначити найменший елемент масиву.	Задано довільний цілочисельний двовимірний масив **A(4, 2)**. Визначити добуток від'ємних елементів масиву.
5	В довільному масиві **A** з **13** елементів визначити суму від'ємних елементів.	Задано довільний цілочисельний двовимірний масив **A(5, 4)**. Знайти максимальний елемент з елементів масиву.
6	В довільному масиві **A** з **14** елементів визначити суму додатних елементів.	Задано довільний цілочисельний двовимірний масив **A(6, 4)**. Знайти мінімальний елемент з елементів масиву.
7	В довільному масиві **A** з **12** елементів визначити найбільший елемент масиву.	Задано довільний цілочисельний двовимірний масив **A(4, 4)**. Визначити суму додатних елементів масиву.

№	Завдання№1	Завдання№2
8	В довільному масиві **A** з **10** елементів визначити добуток додатних елементів масиву.	Задано довільний цілочисельний двовимірний масив **A(3, 4)**. Визначити добуток від'ємних елементів масиву.
9	В довільному масиві **A** з **15** елементів визначити максимальний додатній елемент.	Задано довільний цілочисельний двовимірний масив **A(3, 3)**. Знайти максимальний елемент з елементів масиву.
10	В довільному масиві **A** з **8** елементів визначити мінімальний додатній елемент.	Задано довільний цілочисельний двовимірний масив **A(3, 4)**. Знайти мінімальний елемент з елементів масиву.
11	В довільному масиві **A** з **6** елементів визначити максимальний від'ємний елемент масиву.	Задано довільний цілочисельний двовимірний масив **A(5, 4)**. Визначити суму додатних елементів масиву.
12	В довільному масиві **A** з **10** елементів визначити мінімальний від'ємний елемент масиву.	Задано довільний цілочисельний двовимірний масив **A(4, 2)**. Визначити добуток додатних елементів масиву.
13	В довільному масиві **A** з **13** елементів визначити суму додатних елементів.	Задано довільний цілочисельний двовимірний масив **A(6, 4)**. Знайти максимальний від'ємний елемент з елементів масиву.
14	В довільному масиві **A** з **18** елементів визначити добуток від'ємних елементів.	Задано довільний цілочисельний двовимірний масив **A(4, 5)**. Знайти мінімальний додатний елемент з елементів масиву.

№	Завдання№1	Завдання№2
15	В довільному масиві **A** з **12** елементів визначити найбільший елемент масиву.	Задано довільний цілочисельний двовимірний масив **A(6, 2)**. Визначити добуток додатних елементів масиву.
16	В довільному масиві **A** з **16** елементів визначити найменший елемент масиву.	Задано довільний цілочисельний двовимірний масив **A(5, 3)**. Визначити суму від'ємних елементів масиву.
17	В довільному масиві **A** з **18** елементів визначити добуток додатних елементів.	Задано довільний цілочисельний двовимірний масив **A(4, 3)**. Знайти максимальний елемент з елементів масиву.
18	В довільному масиві **A** з **13** елементів визначити суму від'ємних елементів.	Задано довільний цілочисельний двовимірний масив **A(4, 6)**. Знайти мінімальний елемент з елементів масиву.
19	В довільному масиві **A** з **15** елементів визначити найбільший елемент масиву.	Задано довільний цілочисельний двовимірний масив **A(4, 6)**. Визначити суму додатних елементів масиву.
20	В довільному масиві **A** з **14** елементів визначити найменший елемент масиву.	Задано довільний цілочисельний двовимірний масив **A(4, 5)**. Визначити добуток від'ємних елементів масиву.
21	В довільному масиві A з 17 елементів визначити найменший елемент масиву.	Задано довільний цілочисельний двовимірний масив A(5, 2). Визначити суму додатних елементів масиву.
22	В довільному масиві A з 16 елементів визначити добуток додатних елементів.	Задано довільний цілочисельний двовимірний масив A(5, 3). Знайти мінімальний елемент з елементів масиву.

№	Завдання№1	Завдання№2
23	В довільному масиві А з 14 елементів визначити суму додатних елементів.	Задано довільний цілочисельний двовимірний масив А(3, 6). Знайти максимальний елемент з елементів масиву.
24	В довільному масиві А з 13 елементів визначити найбільший елемент масиву.	Задано довільний цілочисельний двовимірний масив А(6, 6). Визначити добуток додатних елементів масиву.
25	В довільному масиві А з 12 елементів визначити найменший елемент масиву.	Задано довільний цілочисельний двовимірний масив А(5, 5). Визначити суму від’ємних елементів масиву.