МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
АВТОМОБІЛЬНО-ДОРОЖНІЙ ІНСТИТУТ

«ЗАТВЕРДЖУЮ»
Директор АДІ ДВНЗ «ДонНТУ»
М. М. Чальцев

Кафедра «Прикладна математика та інформатика»

МЕТОДИЧНІ ВКАЗІВКИ ДО ПРАКТИКУМУ З ДИСЦИПЛІНИ «ІНФОРМАТИКА», «ІНФОРМАТИКА ТА СИСТЕМОЛОГІЯ». ЧАСТИНА 1

22/23-2014

«РЕКОМЕНДОВАНО»
Навчально-методична комісія факультету «Автомобільні дороги»
Протокол № 6 від 12.12.13 р.

«РЕКОМЕНДОВАНО»
Кафедра «Прикладна математика та інформатика»
Протокол № 4 від 18.12.13 р.

Горлівка – 2014

УДК 681.3.06(07)

Методичні вказівки до практикуму з дисципліни«Інформатика», «Інформатика та системологія». Частина 1 [Електронний ресурс] / укладачі М. Є. Корольов, Р. С. Кравченко. – Електрон. дані. – Горлівка: ДВНЗ «ДонНТУ» АДІ, 2014. – 1 електрон. опт. диск (CD-R); 12 см. – Систем. вимоги: Pentium; 32 MB RAM; WINDOWS XP/Vista/7; MS Word 2000-2010. – Назва з титул. екрану.

Вказівки відповідають структурі дисципліні «Інформатика та системологія».

Вказівки є частиною мультимедійного методичного учбового пакета, до якого також входить інсталяційна збірка демонстраційної версії учбового програмного проекту.

Укладачі: Корольов М. Є., к.ф.-м.н., доц.
Кравченко Р. С.

Відповідальний за випуск: В. Г. Хребет, к.ф.-м.н., доц.

Рецензент: В. Л Ніколаєнко, к.т.н., доц.

ЗМІСТ

ВСТУП

Дані методичні рекомендації є частиною мультимедійного комплекту з основ візуального програмування та створення Windows-додатків для платформи .NET Framework у середовищі Microsoft Visual Studio 2010. Розглядаються оптимальні прийоми розробки програм та реалізації основних типів алгоритмів, а також особливості створення графічних додатків. Розглядаються особливості використання основних компонентів бібліотеки Windows Forms.

Мета методичних рекомендацій полягає у виробленні й закріпленні у студентів прийомів і навичок, пов'язаних зі створенням сучасного програмного забезпечення, уміння користуватися стандартними об'єктними бібліотеками й інтегрованим середовищем розробки. Для досягнення поставлених цілей ідеально підходить платформа .NET Framework і мови, розроблені для цієї платформи, зокрема, мова програмування Visual Basic.

Платформа .NET у цей час активно використовується при розробці програмних комплексів у різних предметних галузях. Серед її переваг можна відзначити:

– єдину програмну модель;
– можливість використання на різних апаратних платформах;
– інтеграцію мов програмування;
– спрощене повторне використання коду;
– автоматичне керування пам'яттю;
– перевірку безпеки типів;
– розвинену підтримку налагодження;
– єдиний принцип обробки помилок;
– можливість взаємодії з існуючим кодом;
– багату бібліотеку класів, доступну для будь-якої мови.

Переваги платформи .NET й її широке поширення роблять досить актуальним вивчення її можливостей у рамках обраного курсу.

1 ВІДОМОСТІ ПРО СЕРЕДОВИЩЕ ПРОГРАМУВАННЯ VISUAL STUDIO 2010

Для запуску програми з головного меню Windows, виконайте наступні дії.

1. Натисніть кнопку Пуск, розташовану в нижній частині екрану.
2. У головному меню Windows оберіть команду Програми. З'явиться меню даної команди.
3. Виберіть у меню пункт Microsoft Visual Studio.
4. У вікні, що з'явилося оберіть Microsoft Visual Studio. На екрані з'явиться головне вікно Visual Studio (рис. 1.1).

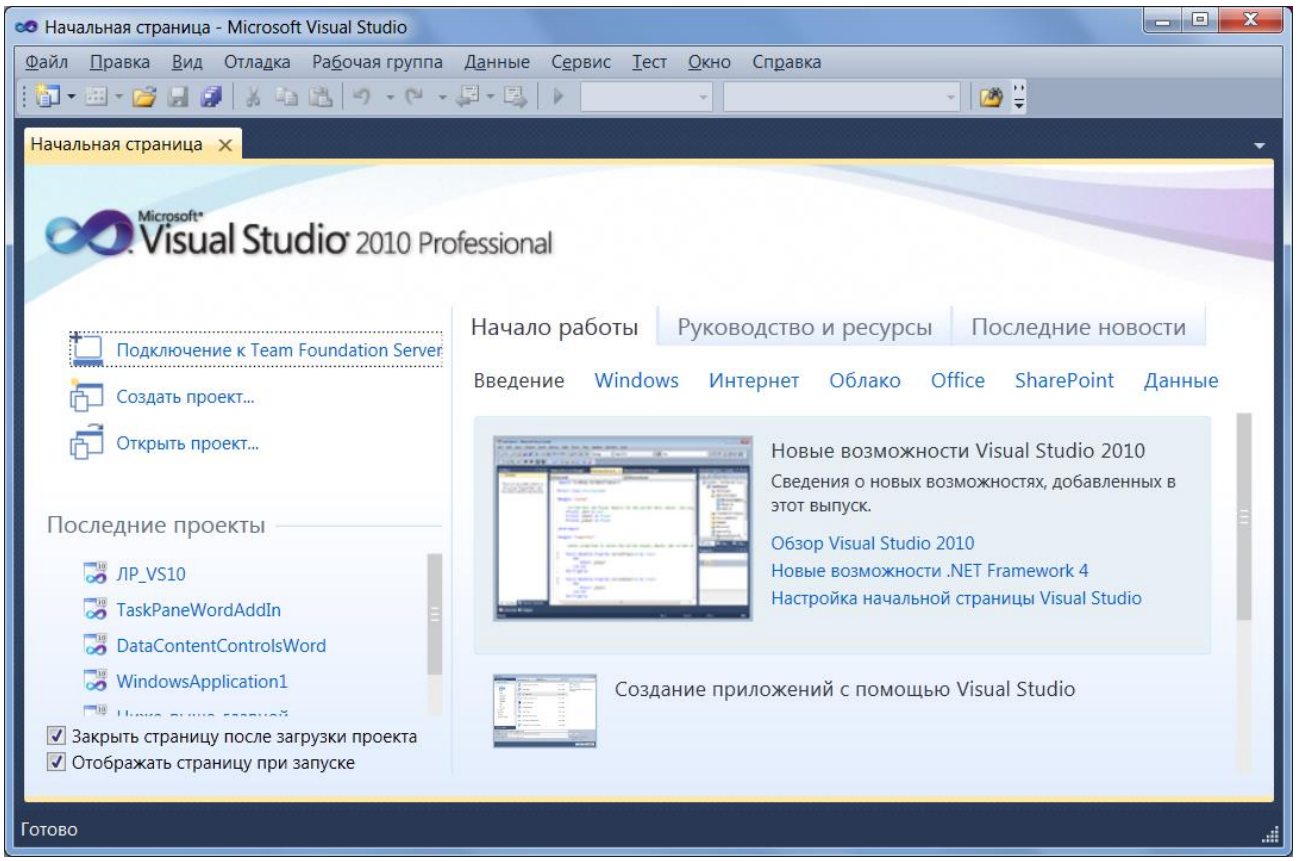


Рисунок 1.1 – Початкова сторінка середовища програмування Visual Studio 2010

Програми, які розробляються у Visual Studio, звичайно називаються проектами або рішеннями, так як вони містять не один-єдиний файл, а багато окремих компонентів. Для програм на Visual Basic .NET завжди є файл проекту (.vbproj) і файл рішення (.sln). Файл проекту містить інформацію, що відноситься до однієї програмної задачі. Файл рішення містить інформацію про один або декілька проектів. Файли рішень

використовуються для управління декількома взаємопов'язаними проектами.

Після створення проекту зазвичай відкривається вся сукупність вікон та панелей середовища програмування (рис. 1.2).

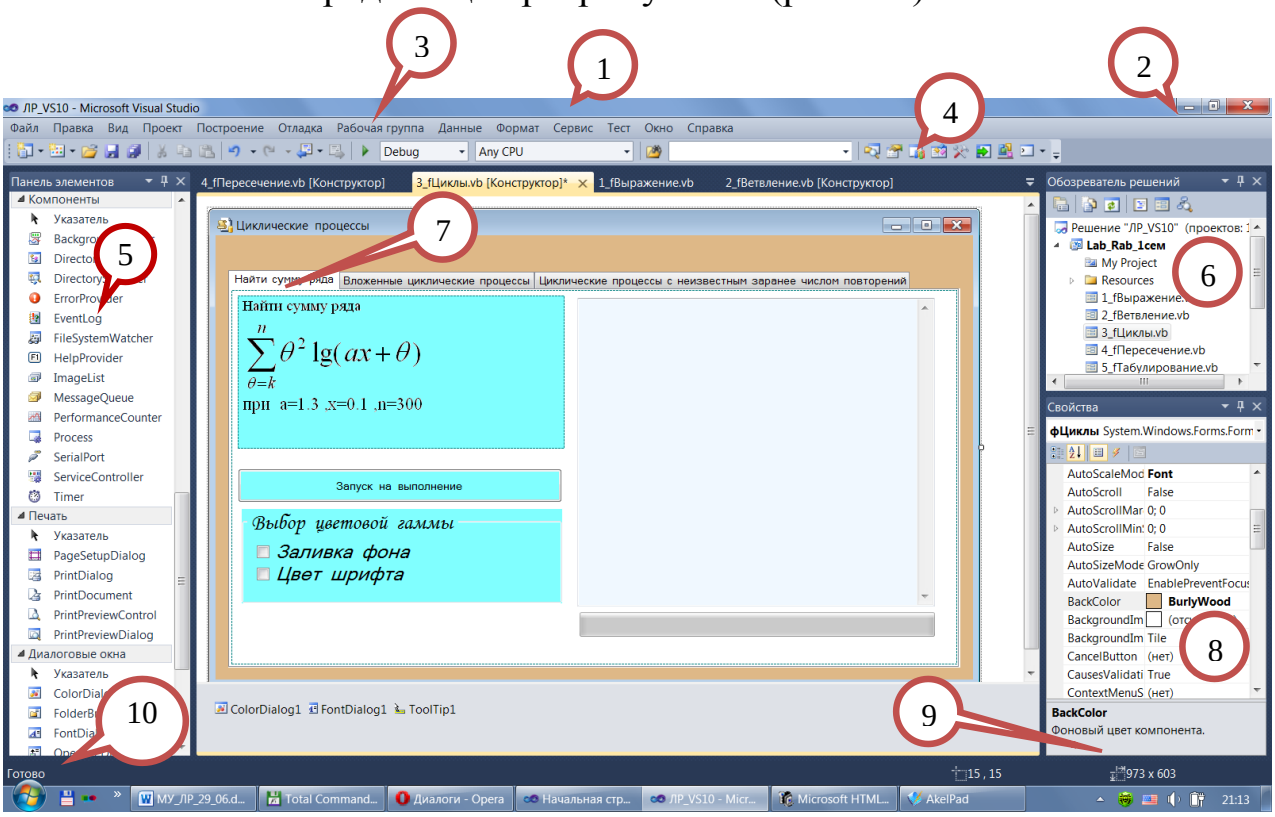


Рисунок 1.2 – Середовище програмування Visual Studio 2010

1. Заголовок головного вікна. Містить іконку Visual Studio 2010, назву поточного проекту, назву робочого середовища (Microsoft Visual Studio).

2. Кнопки головного вікна  згорнути, згорнути / розгорнути у вікно й закрити.

3. Головне меню. Містить пункти, кожен з яких має свої підпункти. Забезпечує швидкий доступ до необхідних функцій.

4. Панель інструментів (ПІ). Дозволяє отримати швидкий доступ до найбільш часто використовуваних функцій, які є в меню. Для зручності вони реалізовані у вигляді значків, кожен з яких відповідає загальноприйнятим стандартам. Панель в Visual Studio 2010 плаваючого типу й складається з декількох частин, які за бажанням можна перетягувати, міняти місцями, додавати нові. Visual Studio при виборі виду проекту розміщує на панелі необхідні частини для зручної роботи з даним проектом.

5. Панель елементів. Викликається кнопкою  на ПІ. Аналогічно ПІ і меню, панель елементів може містити в собі різний набір елементів, що

відповідають типу відкритого проекту. Але загалом дане плаваюче вікно, яке можна переміщати, згортати і закріплювати на місці, містить у собі ряд об'єктів з стандартним кодом. Користувач може використовувати їх як шаблон, просто переміщуючи елемент на потрібне місце на формі; Visual Studio автоматично згенерує код, що скорочує витрати часу на створення інтерфейсу користувача.

6. Оглядач рішень. Викликається кнопкою  на ПІ. Надає швидкий доступ до всіх файлів проекту в деревоподібному оформленні (як у провіднику). Тут можна безпосередньо відкривати їх, перейменовувати, видаляти. Містить і сам файл проекту, форми й каталоги проекту. Містить власну ПІ  , у якій містяться основні команди для роботи з об’єктами проекту.

7. Вікно модуля. У Visual Studio може бути відкрито багато вікон модулів, в такому випадку вони відображаються у вигляді закладок. У даному випадку відкрито 1 вікно модуля. Воно може містити програмний код модуля (викликається кнопкою  у вікні оглядача рішень (6) для відповідного модуля), що представлений в зручному вигляді, з можливістю приховування розділів та функцією редагування й читання. Може бути у вигляді конструктора (викликається кнопкою  у вікні оглядача рішень (6) для відповідного модуля), як у нашому випадку. Конструктор надає можливість бачити реалізацію коду наочно, тобто у вигляді вже готового візуального об’єкта (форми).

8. Властивості. Викликається кнопкою  на ПІ. Відображає властивості виділеного в даний момент об'єкта (елемента управління, форми, тощо). Властивості надаються в згрупованому вигляді для зручності. Змінюючи поля у властивостях, Visual Studio автоматично змінить код, згідно з нововведеннями.

9. Довідка та вибраний елемент. Відображає стислу інформацію про вибраний об'єкт або його властивості, про значення слова в коді програми.

10. Рядок стану. Відображає інформацію про поточну подію у середовищі: відкриття файлу, збірка проекту, ініціалізація об'єктів. «Готово» означає, що Visual Studio виконала всі процеси й очікує дій вже з боку користувача.

Visual Studio включає підтримку мов програмування C# 4.0, Visual Basic .NET 10.0, а також мови F#, які забезпечують підтримку платформи .Net. Така різноманітність програмних засобів дозволяє обирати мову на основі особистих переваг, оскільки всі вони мають однакові можливості. Усі приклади, наведені в наступних главах, реалізовані на мові програмування Visual Basic 2010 .Net (далі VB).

VB призначений для ефективного створення типобезпечних і об'єктно-орієнтованих додатків (ООП). Програми, написані мовою VB, як

і на інших мовах, призначених для Microsoft .NET Framework, відрізняються безпекою й підтримкою взаємодії. Це забезпечує можливість швидкого й ефективного створення додатків на основі .NET Framework.

1.1 Створення завантажувальної форми у VB

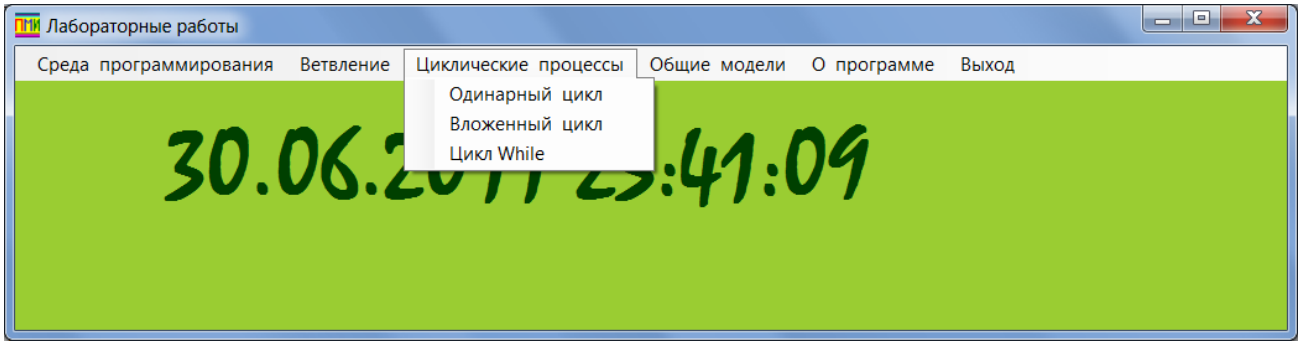
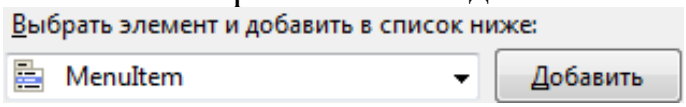


Рисунок 1.3 – Завантажувальна форма

Таблиця 1.1 – Ієрархія структури елементів головного меню

Ім’я (Name)	Напис (Text)
мнВыражение	Среда программирования
мнЭлВыражение	Выражение
мнВетвление	Ветвление
мн2ветви	IF-строка
мн3ветви	IF-блок
мн4ветви	Select Case
мнЦиклы	Циклические процессы
мнОдЦикл	Одинарный цикл
мнВлЦикл	Вложенный цикл
мнЛейбниц	Цикл While
мнОбщМодели	Общие модели
мнКриваяОбл	Пересечение кривой и области
мнТабулирование	Табулирование с проколотой точкой
mnuAbout	
mnuFlash	О программе
мнВыход	Выход

Для того, щоб створити головне меню згідно з шаблоном (рис. 1.3) та структурою (табл. 1.1), обираємо на панелі інструментів елемент MenuStrip та розміщуємо його на формі. Потім правою кнопкою миші

викликаємо контекстне меню → обираємо *Правка элементов* у спадаючому меню → обираємо необхідний тип елементу управління (MenuItem) 

→ натискаємо *Добавить*. Викликаємо вікно властивостей (контекстне меню → *Свойства*) → вказуємо ім'я (Name) **мнВыражение** та надпис Text **Среда программирования**.

Щоб створити наступний за ієрархією елемент згідно структурі встановлюємо курсор на вже створеному елементі → контекстне меню → Правка DropDownItems (рис. 1.4). Встановлюємо необхідні параметри, а насамперед Name (Ім'я), Text (Напис).

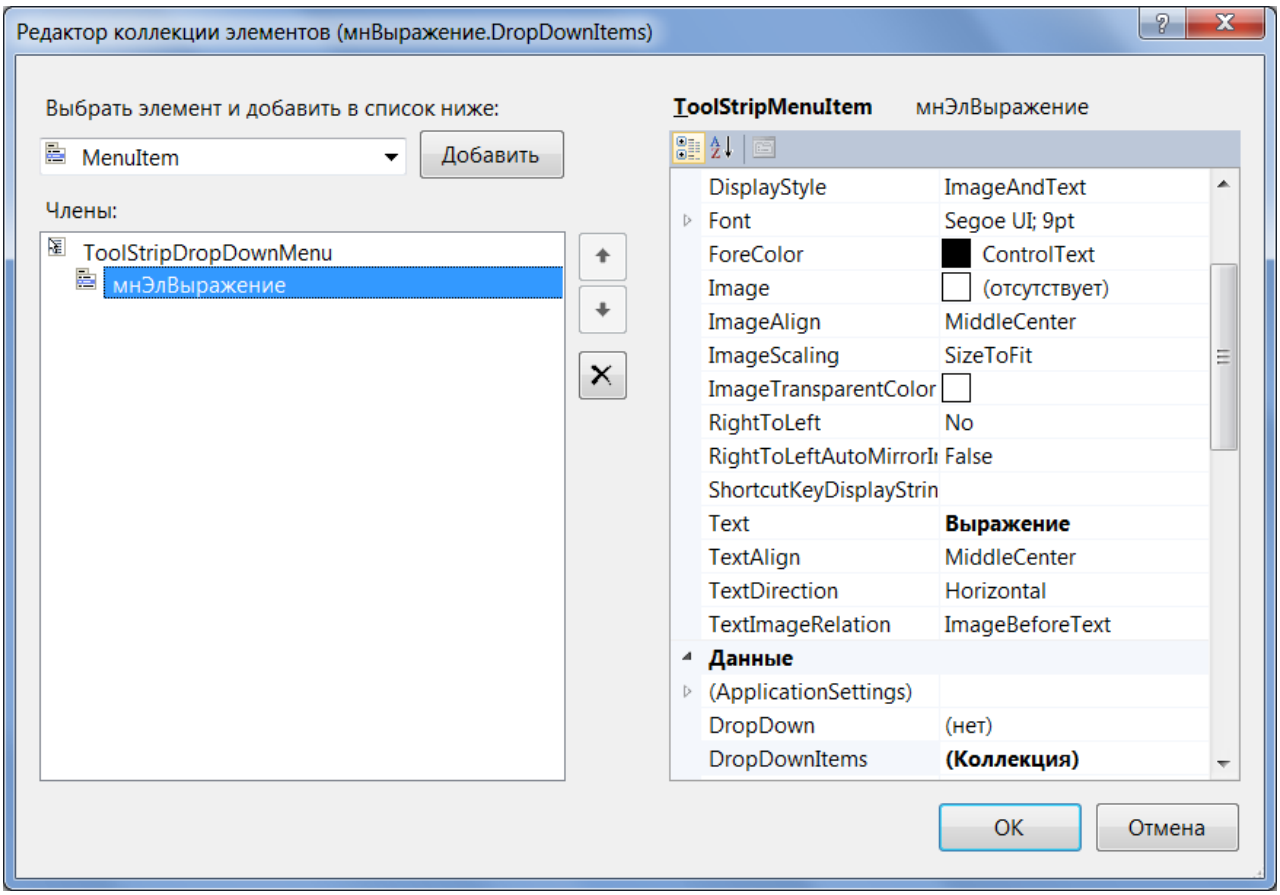


Рисунок 1.4 – Редактор об'єкта DropDownItems

Також можна створити меню головної форми за допомогою майстра візуалізації, вписуючи потрібну назву у відповідну область Выражение. А потім задати ім'я у вікні *Свойства* у правому нижньому кутку.

Аналогічно створюємо інші елементи меню.

За реалізацію функціональних можливостей меню відповідає наступний алгоритм:

```
Private Sub фМеню_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        ' установка необходимых параметров для вывода текущих
даты-времени в Label1, и обновление информации с использованием
таймера Timer1
        Me.BackColor = Color.YellowGreen
        Label1.Text = DateTime.Now
        Timer1.Interval = 1000
        Timer1.Enabled = True
    End Sub
    Private Sub Timer1_Tick(ByVal Sender As Object, ByVal e As
EventArgs) Handles Timer1.Tick
'Событие генерируется по истечении заданного интервала времени
        Label1.Text = DateTime.Now
    End Sub
    Private Sub мнЭлВыражение_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles мнЭлВыражение.Click
        фВыражение.Show() ' вызов формы задачи "Выражение"
    End Sub
Private Sub мн2ветви_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles мн2ветви.Click
   фВетвление.Show() : фВетвление.TabControl1.SelectTab(0) ' вызов
формы задачи "Ветвление", открытие 1-й вкладки (индексация с 0)
    End Sub
    Private Sub мн3ветви_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles мн3ветви.Click
        фВетвление.Show() : фВетвление.TabControl1.SelectTab(1) '
вызов формы задачи "Ветвление", открытие 2-й вкладки (индекс с 0)
    End Sub
    Private Sub мн4ветви_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles мн4ветви.Click
        фВетвление.Show() : фВетвление.TabControl1.SelectTab(2) '
вызов формы задачи "Ветвление", открытие 3-й вкладки (индекс с 0)
    End Sub
    Private Sub мнОдЦикл_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles мнОдЦикл.Click
        фЦиклы.Show() : фЦиклы.TabControl1.SelectTab(0) ' вызов
формы задачи "Циклы", открытие 1-й вкладки (индексация с 0)
    End Sub
    Private Sub мнВлЦикл_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles мнВлЦикл.Click
        фЦиклы.Show() : фЦиклы.Show() :
фЦиклы.TabControl1.SelectTab(1) ' вызов формы задачи "Циклы",
открытие 2-й вкладки (индексация с 0)
    End Sub
    Private Sub мнЛейбниц_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles мнЛейбниц.Click
        фЦиклы.Show() : фЦиклы.Show() :
фЦиклы.TabControl1.SelectTab(2) ' вызов формы задачи "Циклы",
открытие 3-й вкладки (индексация с 0)
    End Sub
    Private Sub мнКриваяОбл_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles мнКриваяОбл.Click
```

```
        фКриваяОбласть.Show() ' вызов формы задачи "Пересечение
кривой и области"
    End Sub
    Private Sub мнТабулирование_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
мнТабулирование.Click
    фТабулирование.Show() ' вызов формы "Табулирование функции"
    End Sub
    Private Sub мнВыход_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles мнВыход.Click
        Me.Close() ' закрытие главной формы
    End Sub
```

2 МАТЕМАТИЧНІ ОБЧИСЛЕННЯ У VB

2.1 Змінні

Змінна – це тимчасове місце в пам’яті для зберігання даних програми. У коді може використовуватися одна чи декілька змінних, і вони можуть містити текст, числа, дати або властивості. Змінні корисні тим, що вони дозволяють присвоїти кожній частині даних, з якими ви працюєте, короткі імена. Змінні можуть зберігати інформацію, введену користувачем при виконанні програми, результат деякого обчислення або блок даних, який потрібно вивести у формі.

Якщо коротко, то змінні є зручними контейнерами, які ви можете використовувати для зберігання інформації довільного типу (табл. 2.1).

Таблиця 2.1 – Типи даних

Тип даних	Розмір	Діапазон значень
Integer (Ціле) Призначений для обробки цілочислових значень	4 байта	Від –2 147 483 648 до 2 147 483 647
Long (Довге ціле) Призначений для обробки цілочислових значень зі збільшеного діапазону	4 байта	Від –9,2E+18 до 9,2E+18
Single (Одинарної точності з плаваючою десятковою крапкою) Призначений для зберігання чисел в експонентній формі одинарної точності	4 байта	Від –3,402824E+38 до –1,401298E–45 для негативних значень; від 1,401298E–45 до 3,4028235E+38 для позитивних значень
Double (Подвійної точності з плаваючою десятковою крапкою) Призначений для зберігання чисел в експонентній формі подвійної точності	8 байт	Від –1,798E+308 до –4.9407E–324 для негативних значень; Від 4,9407E–324 до 1,7977E+308 для позитивних значень
String (Строковий) Призначений для зберігання строк різної довжини	залежить від платформи	До 2 мільярдів знаків в кодуванні Юнікод
Boolean (Логічний)	залежить від платформи	True / False
Date (Дата)	8 байт	Від 0:00:00 (опівночі) 1 січня 0001 року до 11:59:59 вечора 31 грудня 9999 року

Оголошення змінних

Використання змінних у програмах Visual Basic .NET вимагає деякого планування. Перш ніж ви зможете використовувати змінну, ви повинні виділити пам'ять, яку ця змінна буде використовувати.

Для оголошення змінної необхідно ввести її ім'я після оператора Dim (від англ. dimension – величина). Це оголошення при виконанні програми виділяє для змінної місце в пам'яті й дає можливість компілятору визначити тип даних, який він може зустріти надалі. Хоча це оголошення можна зробити в будь-якому місці коду програми (за умови, що воно з'являється до використання змінної), більшість програмістів оголошують змінні в одному місці на початку їх процедур обробки подій або модулів коду.

Наприклад, наступний оператор створює в програмі місце для змінної з ім'ям Змінна, яка буде зберігати текстове або рядкове значення.

```
Dim Переменная As String
Dim Переменная1, Переменная2,…, ПеременнаяN As Integer
```

Таблиця 2.2 – Приклади оголошення змінних

Тип даних	За допомогою типа даних
Ціле	Dim x As Integer
Дійсне одинарної точності	Dim x As Single
Дійсне подвійної точності з плаваючою десятковою комою	Dim x As Double

Область видимості змінних

При виконанні програми принципове значення має область видимості використовуваних змінних. Спроба використання змінних, які не діють в даному місці програми, призводить до помилки програми або до неоднозначності результатів. У Visual Basic можуть застосовуватися глобальні та локальні змінні. Глобальні змінні доступні з будь-якої частини програми. Для локальних змінних можна задавати область видимості в рамках усього модуля або окремої процедури.

Для оголошення змінної, локальної всередині модуля або форми, використовуються оператори Private або Dim в розділі [Общие / Объявления] (General / Declaration) модуля або форми. У цьому випадку оголошена змінна буде доступна для всіх процедур, які входять у форму або модуль , але в той же час виявиться недоступною в процедурах інших модулів і форм.

Змінні, локальні на рівні процедури, створюються операторами Dim або Static всередині процедури.

```
Sub Процедура
     Dim локПеременная As Integer
End Sub
```

Для створення глобальної змінної, в розділі [Общие / Объявления] (General / Declaration) головного модуля програми необхідно включити оператор Public.

```
Public глПеременная As Date
```

Присвоєння значення змінним

Після того, як змінна створена, можна задати її значення, використовуючи оператор присвоєння (=).

Наприклад, наступний оператор програми присвоює змінній < Имя > значення "Джон":

```
Имя = "Джон"
```

Присвоїти значення змінної можливо при її оголошенні:

```
Dim X As Integer = 4
```

Константи

Якщо в програмі використовується змінна, значення якої не змінюється, то краще використовувати замість змінної константу. Константа являє собою умовне ім'я, що використовується замість числа, або текстового рядка, що не підлягає зміні. Дія константи схожа з дією змінної, але її значення не може бути змінено в процесі використання програми. Константи оголошуються за допомогою ключового слова `Const`.

```
Public/ Private Const < ім'я > [ AS < ім'я типу > ] = < значення >
Const Pi=3.14159
```

Коментарі

Крім команд і виразів можна включити в методи й процедури довільний текст або коментарі. Коментарі, що пояснюють текст програми, зроблять її більш читабельною й допомагають краще орієнтуватися в програмному коді.

Для включення в текст програми коментаря необхідно ввести символ ['], який може бути першим символом рядку або перебувати в будь-якому її місці. Цей символ означає початок коментаря. Будь-який текст,

розташований у рядку слідом за цим символом, буде сприйматися як коментар, а не як програмний код для трансляції.

```
Imports System.Math 'импорт пространства имен System.Math
```

Розміщення оператора на декількох рядках

У випадку, якщо оператор має велику довжину, його можна розбити на кілька рядків, використовуючи символи продовження рядка, що представляють собою пробіл, за яким слідує символ підкреслення [_].

Розміщення декількох операторів на одному рядку

Як правило, при написанні програм оператори розміщують на окремому рядку. Якщо оператори мають невелику довжину, можна їх помістити на одному рядку, розділивши двокрапкою [:].

Основні варіанти розміщення операторів подано у табл. 2.3.

Таблиця 2.3 – Приклади розміщення операторів

`a = 2`
`b = 5`
`c = 10`
	Оператори на окремому рядку

`a = 2 : b = 5 : c = 10`	Оператори в один рядок
`MsgBox("x =" & Str(x) & Chr(13) & "cos(x) =" & Str(Cos(x)), _ vbInformation + vbOKCancel, "Результат")`	Конструкція розділена знаком переносу строки [_]

2.2 Математичні оператори. Функції

У табл. 2.4 наведені умовні позначення шести базових математичних операцій.

Результат стандартного ділення (/) завжди відноситься до типу Double, навіть у разі ділення без залишку. Результат цілочисельного ділення (\ ) завжди відноситься до типу Integer.

Таблиця 2.4 – Арифметичні операції

Оператор	Операція
+	Додавання
-	Віднімання та позначення негативних чисел
/	Ділення
\	Цілочисельне ділення
*	Множення
^	Піднесення до степені

Стандартні типи функцій

Функція – це оператор, що виконує певні дії та повертає результат роботи у програму. Функція може мати один або кілька аргументів, які беруться в дужки й відокремлюються між собою комами.

Круглі дужки та пріоритет операцій

При обробці складних виразів послідовність виконання операцій задається двома способами. При використанні круглих дужок вам не доведеться запам'ятовувати пріоритети різних операцій (табл. 2.5). У VB. NET, як і в багатьох мовах програмування, операції мають пріоритет, який визначає послідовність їх виконання. Множення володіє більш високим пріоритетом, ніж складання; отже, *вираз 3+4*5 дорівнює 23, оскільки множення (4*5) виконується раніше, ніж складання.*

Нижче перераховані математичні операції в порядку зменшення пріоритету.

Таблиця 2.5 – Арифметичні операції за пріоритетом

Оператори	Порядок обчислення
( )	Першими завжди обчислюються значення в круглих дужках
^	Піднесення числа до степеня обчислюється в другу чергу
-	Створення негативного числа (зміна знака) йде третім
* /	Четвертими йдуть множення і ділення
\	П'яте – цілочисельне ділення
Mod	Шосте – залишок від ділення
+ -	Останні – додавання і віднімання

Якщо дві операції мають однаковий пріоритет, порядок виконання визначається порядком їх слідування у виразі (зліва направо).

Математичні функції та математичні константи

Математичні функції та математичні константи в VB. NET реалізовані у вигляді методу класу Math, що входить в .NET Framework.

Всі перераховані методи є загальними методами класу Math, тому вони повинні викликатися з префіксом Math – наприклад: `Math.Log10 (10)`, або імпортувати клас Math у модуль: `Imports System.Math`.

Таблиця 2.6 – Основні математичні функції

Математична функція	Опис
Abs(арг)	Повертає абсолютне значення (модуль) числа (арг)
Acos(арг)	Повертає кут, косинус якого дорівнює заданому числу (арг)
Asin(арг)	Повертає кут, синус якого дорівнює заданому числу (арг)
Atan(арг)	Повертає кут, тангенс якого дорівнює заданому числу (арг)
Cos(арг)	Повертає косинус заданого кута (арг)
Exp(арг)	Повертає число e (≈2,71828182845905), зведене в задану ступінь (арг)
Log(арг)	Повертає натуральний логарифм числа (арг)
Log10(арг)	Повертає десятковий логарифм числа (арг)
Log(арг, осн)	Повертає логарифм числа (арг) за заданою основою (осн)
Round(арг[, точн])	Округляє число (арг) до найближчого цілого або зазначеної кількості десяткових знаків після коми (точн)
Sign(арг)	Повертає величину, що визначає знак числа (арг)
Sin(арг)	Повертає синус заданого кута (арг)
Sqrt(арг)	Повертає квадратний корінь числа (арг)
Tan(арг)	Повертає тангенс заданого кута (арг)

2.3 Функції конвертації

Таблиця 2.7 – Основні функції конвертації

Ім'я функції	Тип, що повертається	Діапазон для аргументу
CDbl	Double	Від –1.79769313486231570E+308 до –4.94065645841246544E–324 для негативних значень; від 4.94065645841246544E–324 до 1.79769313486231570E+308 для позитивних значень
CInt	Integer	Від –2147483648 до 2147483647; дробова частина округляється
CLng	Long	Від –9,223,372,036,854,775,808 до 9,223,372,036,854,775,807; дробова частина округляється
CShort	Short	Від –32768 до 32767; дробова частина округляється
CSng	Single	Від –3,402823E+38 до –1,401298E–45 для негативних значень Від 1,401298E–45 до 3,402823E+38 для позитивних значений
CStr	String	Значення що повертаються функцією CStr залежать від аргументу
Val	Числовий	Повертаються числа, що містяться в строковій змінній в якості числових значень відповідного типу

2.4 Функції вводу/виводу

Для реалізації більш гнучкого інтерфейсу користувача доцільно замість жорсткого присвоювання значень змінним або об’єктам використовувати функцію вводу InputBox. Для виводу результатів або повідомлень програми, можливе використання функції виводу MsgBox.

Оператор вводу InputBox

Синтаксис:

```
InputBox (<повідомлення> ,[<заголовок>], [<значення>])
```

<повідомлення> / [<заголовок>] — Обов'язковий / необов'язковий параметр типу String. довільна послідовність символів в подвійних лапках ("); текст, який міститься в згенерованому вікні InputBox в якості пояснення та до заголовку відповідно. Якщо текст складається з декількох рядків, то можна розділити рядки за допомогою знаку повернення каретки (Chr(13)), знаку переходу на новий рядок (Chr(10)) або їх комбінації між кожним рядком;

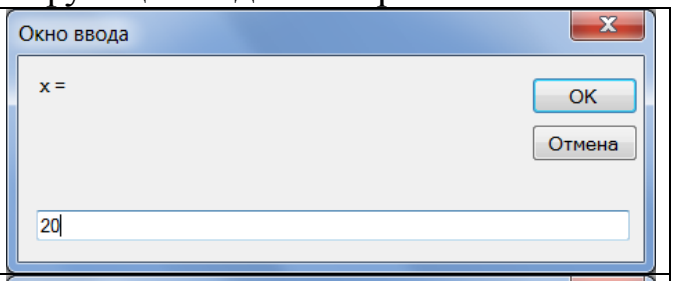
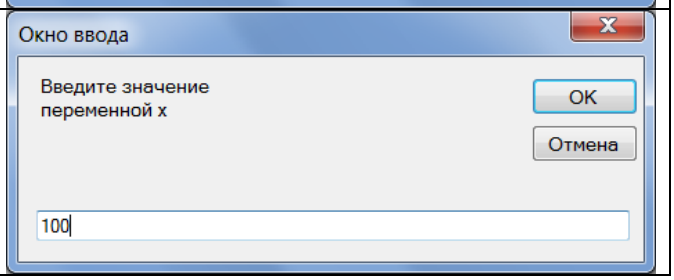
<значення> — значення за замовчуванням змінної відповідного типу.

Якщо будь-який необов'язковий параметр оператора замовчується, ставляться коми.

Приклад (результат виконання подано на рис. 2.1):

```
x = InputBox("x = ", "Окно ввода", 20, 100, 500)
```

Таблиця 2.1 – Приклади використання функції введення InputBox

`x = InputBox("x = ", "Окно ввода", 20)`	
`x = InputBox("Введите значение" & Chr(13) & "переменной x", "Окно ввода", 100)`	

Оператор виведення даних MsgBox

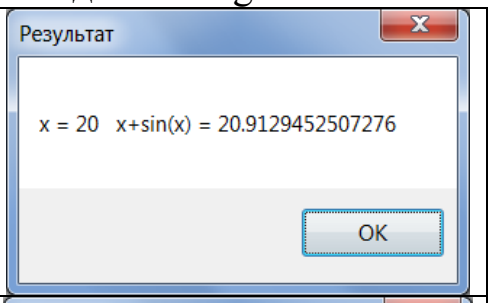
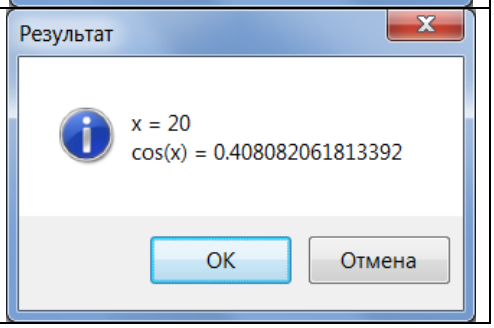
Синтаксис:

```
MsgBox("<список виводу>" [,<кнопки>] [, "<заголовок>"])
```

<список виводу> — Обов'язковий параметр типу String. Текст, що відображається в діалоговому вікні у вигляді повідомлення. Якщо текст складається з декількох рядків, то можна розділити рядки за допомогою знаку повернення каретки (Chr(13)), знаку переходу на новий рядок (Chr(10)) або їх комбінації між кожним рядком. В якості списку виводу також

	зручно використовувати функцію Str (аргумент). Елементи списку виводу з'єднуються між собою знаком &;
[<кнопки>]	Необов'язковий параметр. Значення, що задають номер і тип відображуваних кнопок, стиль використовуваного значка, тип кнопки за замовчуванням і ознаку модальності вікна повідомлення. Якщо даний параметр опущений, по-замовчуванню використовується нульове значення;
[<заголовок>]	Необов'язковий параметр типу String. Текст, який відображається в рядку заголовка діалогового вікна. Якщо параметр опущений, рядок заголовка містить ім'я додатку.

Таблиця 2.2 – Приклади використання функції виведення MsgBox

`MsgBox("x =" & Str(x) & " " & "x+sin(x) =" & Str(x + Sin(x)), , "Результат")`	
`MsgBox("x =" & Str(x) & Chr(13) & "cos(x) =" & Str(Cos(x)), vbInformation + vbOKCancel, "Результат")`	

2.5 Розв'язок задачі «Обчислення значення математичного виразу»

Умова. Обчислити значення виразу, з розробкою інтерфейсу користувача:

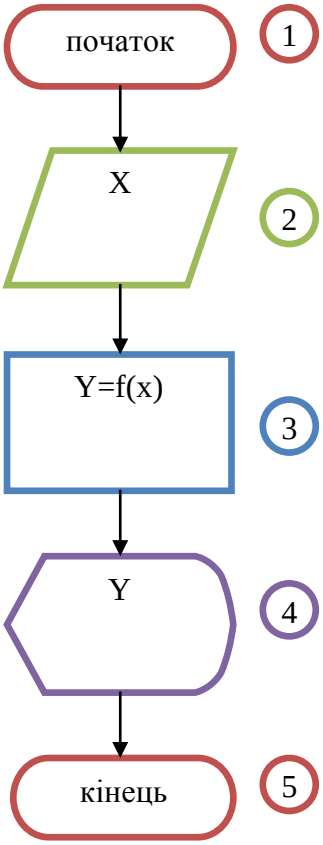
$$\left(Sin(x)-1+\frac{1}{Sin^2(3x^5+67x^2-6x)}\right)*\left(\frac{x^2-x}{(2-x)^2}\right) \qquad \text{при } x = 0{,}5$$

Використовуючи викладки гл. 2.2 отримаємо еквівалентну математичну конструкцію згідно з даною умовою:

```
y = (Sin(x) - 1 + 1 / ((Sin(3 * x ^ 5 + 67 * x ^ 2 - 6 * x)) ^ 2)) * ((x ^ 2 - x) / ((2 - x) ^ 2))
```

Стислі відомості про елементи управління, які призначенні для створення інтерфейсу користувача наведені у додатку Б.

Принципова блок-схема алгоритму



1) початок алгоритму
2) формування даних
3) процес: обчислення виразу
4) виведення результату на дисплей
5) кінець алгоритму

```
f(x) = (Sin(x) - 1 + 1 / ((Sin(3 * x ^ 5 + 67 * x ^ 2 - 6 *
x)) ^ 2)) * ((x ^ 2 - x) / ((2 - x) ^ 2))
```

Програмний код алгоритму

Імпортуємо системний клас Math, який служить для використання математичних функцій і математичних констант:

```
Imports System.Math ' импорт пространства имен System.Math
Public Class Form1
```

Оголошуємо змінні:

```
    Dim y As Double, x As Double
    Private Sub Form3_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
        Label1.Text = "Вычислить, с разработкой
соответствующего интерфейса пользователя:   "
        Label1.BackColor = Color.Gray
        Label1.ForeColor = Color.WhiteSmoke
        Label1.Font = New System.Drawing.Font("Arial", 10)
    End Sub
    Private Sub NumericUpDown1_Click(ByVal sender As Object,
ByVal e As System.EventArgs) Handles NumericUpDown1.Click
```

У разі помилки програма одразу перейде до позначки 999

```
        On Error GoTo 999
        x = TextBox1.Text
```

В наступному блоці обчислимо вираз, згідно з умовою:

```
            y = (Sin(x) - 1 + 1 / ((Sin(3 * x ^ 5 + 67 * x ^ 2 -
6 * x)) ^ 2)) * ((x ^ 2 - x) / ((2 - x) ^ 2))
            TextBox2.Text = y
```

Використання математичної функції Round, яка округлює значення змінної *y* до встановленої точності:

```
            TextBox2.Text = Round(y, CInt(NumericUpDown1.Value))
    999:
        End Sub
    End Class
```

Тестування алгоритму

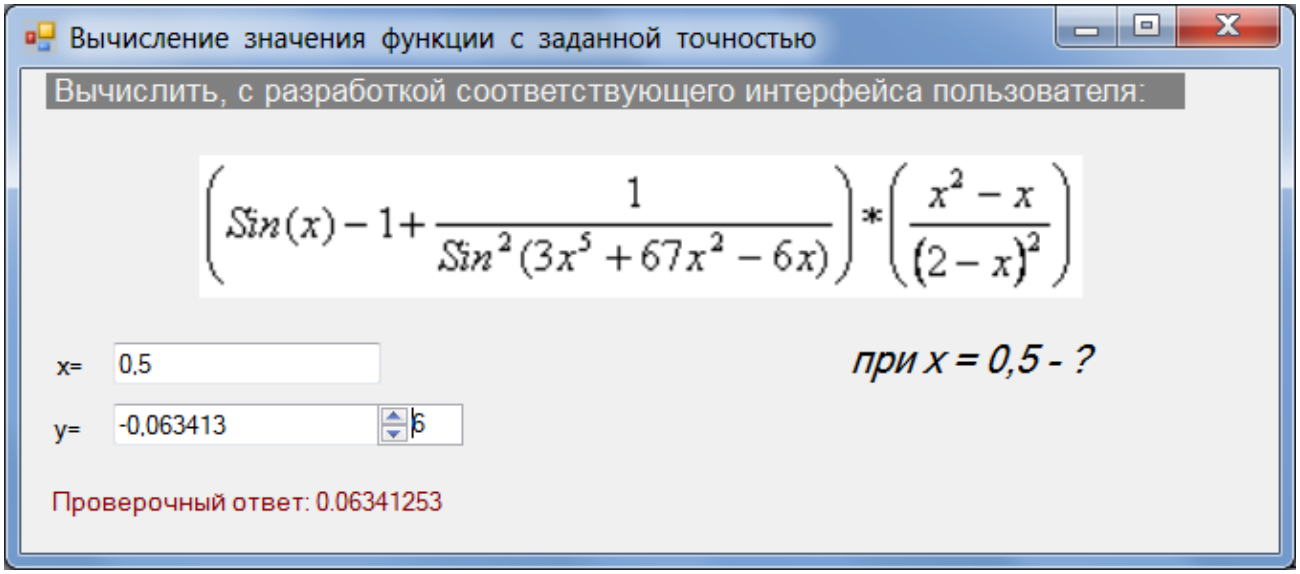


Рисунок 2.1 – Форма задачі «Обчислення виразу»

3. АЛГОРИТМИ РОЗГАЛУЖЕННЯ У VB

3.1 Умовний оператор If…Then…Else

Умовний оператор використовується для подання розгалуженого обчислювального процесу у коді програми мовою VB.Net. Детальна інформація про алгоритми розгалуження подана у додатку А (гл. А.1).

Умовний оператор If…Then…Else має два типи структур: рядкову та блочну.

Синтаксис рядкової структури умовного оператора:

```
If <логічний вираз> Then <оператори1> [Else <оператори2>]
```

логічний вираз	результатом може бути одне з двох значень (істина) або (хибність); логічний вираз складається з операндів логічного типу, між якими містяться знаки логічних операцій і може мати одне з двох значень: True (Істина) або False (Хибність).
оператори 1	виконуються при значенні логічного виразу «істина».
оператори 2	виконуються при значенні логічного виразу «хибність».

У табл. 3.1 наведено список операторів порівняння й умов.

Таблиця 3.1 – Логічні оператори та результати

Оператор	Істина, якщо	Хибність, якщо
< (менше)	expression1 < expression2	expression1 >= expression2
<= (менше або рівно)	expression1 <= expression2	expression1 > expression2
> (більше)	expression1 > expression2	expression1 <= expression2
>= (більше або рівно)	expression1 >= expression2	expression1 < expression2
= (рівно)	expression1 = expression2	expression1 <> expression2
<> (не рівно)	expression1 <> expression2	expression1 = expression2

Синтаксис блочної форми умовного оператора:

```
If <логічний вираз 1> Then
     оператори 1
[Else if <логічний вираз 2> Then
     оператори 2]
[Else
     оператори n]
End If
```

оператори 1	довільна кількість операторів, що виконуються за умовою:
логічний вираз 1	є «істина»;

логічний вираз 2	повертає ненульове значення («істина») або нуль («хибність»);
оператори 2	довільна кількість операторів, що виконуються за умовою:
логічний вираз 2	є «істина»
оператори n	довільна кількість операторів, що виконуються при інших умовах.

В наступному прикладі реалізований алгоритм визначення кількості цифр числа. Критерієм визначення є належність числа до одиниць, десяток, сотень (1 цифра, 2 цифри, 3 цифри відповідно).

```
Dim число, цифры As Integer
        Dim строка As String
        число = 53 ' задаем число
        'в логической конструкции проверяем принадлежность числа к
одному из интервалов
        If число < 10 Then
            цифры = 1 'если число меньше 10 - оно содержит 1 цифру
        ElseIf число < 100 Then
            цифры = 2 'если число меньше 100 (и при этом больше
10, исходя из предыдущего условия) - оно содержит 2 цифры
        Else
            цифры = 3 'если число больше, или равно 100 (следует
из предыдущих 2 условий) - оно содержит 3 цифры
        End If
        If цифры = 1 Then строка = "Одна цифра" Else строка =
"Несколько цифр"
        MsgBox(строка, MsgBoxStyle.Information, "Сообщение")
```

3.2 Логічна конструкція Select Case

Конструкція Select Case дозволяє обробляти в програмі декілька умов і аналогічна блоку конструкцій If ... Then ... Else. Ця конструкція складається з аналізованого виразу й набору операторів Case на кожне можливе значення виразу.

Синтаксис конструкції Select Case наступний:

```
Select Case <змінна>
     CASE <значення 1>
<конструкція 1>
     CASE <значення 2>
<конструкція 2>
     ...
End Select
```

змінна	змінна для порівняння;
значення 1, значення 2,...	обов'язковий в операторі Case. Список можливих значень для параметра <змінна>, що розділяються

	комами, який може мати вигляд (табл. 3.2);
<конструкція 1> / <конструкція 2>	Один або група операторів, що реалізуються, якщо значенню <змінна> відповідає значення 1 / значення 2.

Таблиця 3.2 – Конструкції для оператора Case

значення1 To значення2	вказує діапазон значень для <змінна>
Is <оператор порівняння> <значення>	Використовується з оператором порівняння (=, <>, <, <=, > або >=) для обмеження можливих значень <змінна>
значення	Вид, у якому надається тільки значення, розглядається як особливий випадок використання Is, де оператор порівняння – знак рівності (=). Дана форма обчислюється як змінна = значення

Логіка виконання конструкції наступна: обчислюється значення заданого в конструкції виразу. Потім отримане значення порівнюється зі значеннями, що задаються в операторах CASE конструкції. Якщо знайдено шукане значення, виконуються команди, приписані даному оператору CASE. Після завершення виконання конструкцій управління буде передано конструкції, наступної за ключовим словом End Select.

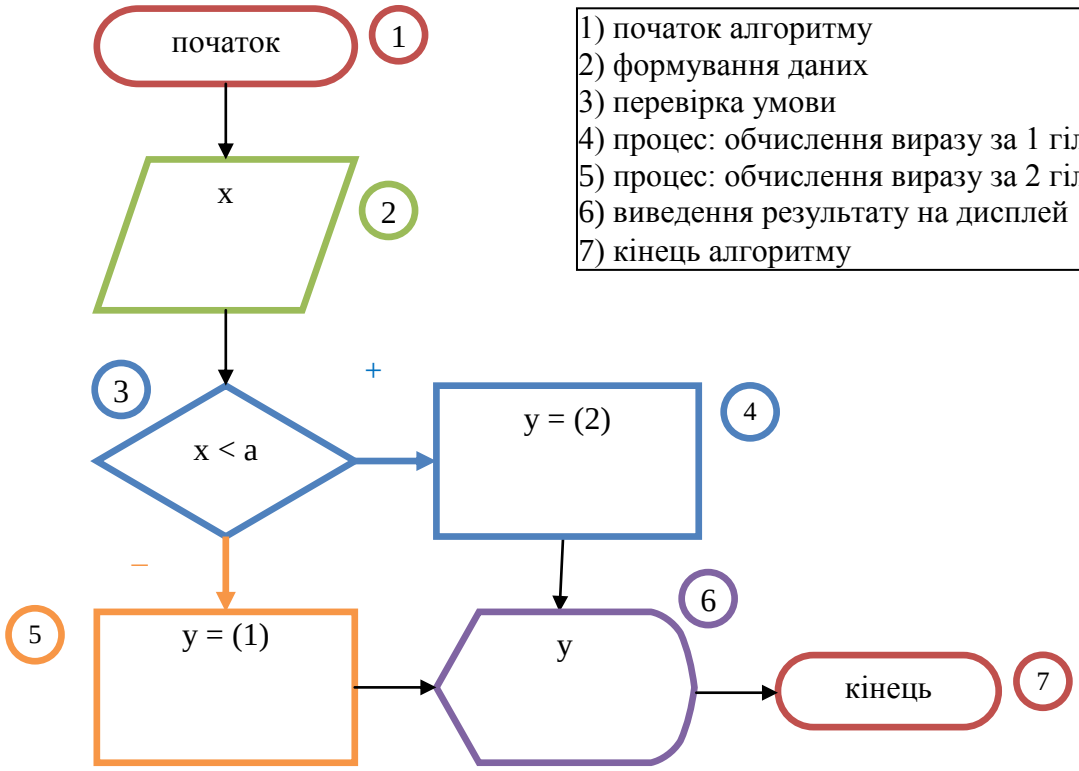
3.3 Розв'язок задачі «Алгоритм розгалуження 2 гілки» (If-рядок)

Умова. Обчислити значення кусково-аналітичної функції 2 гілки з використанням конструкції (If-рядок).

$$y = \begin{cases} e, x < a \\ \pi, x \geq a \end{cases}$$

Стислі відомості про елементи управління, які призначені для створення інтерфейсу користувача наведені у додатку Б.

Принципова блок-схема алгоритму



1) початок алгоритму
2) формування даних
3) перевірка умови
4) процес: обчислення виразу за 1 гілкою
5) процес: обчислення виразу за 2 гілкою
6) виведення результату на дисплей
7) кінець алгоритму

$$y = \begin{cases} (1) = \text{Math.E} \\ (2) = \text{Math.PI} \end{cases}$$

Програмний код алгоритму

```
Dim x, y, v, a
    Dim qqq As Label
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Button1.Click
```

Якщо буде обраний RadioButton1, то значення аргументів будуть вводитися через текстові поля:

```
        If RadioButton1.Checked = True Then
            a = CDbl(TextBox1.Text)
            x = CDbl(TextBox2.Text)
        Else
```

Якщо буде обраний RadioButton2, то значення аргументів будуть вводитися через діалогові вікна, за допомогою оператора вводу InputBox:

```
            a = InputBox("Введите значение узла a")
            x = InputBox("Введите значение аргумента x")
        End If
```

В наступному блоці реалізується алгоритм вибору гілки згідно з умовою розгалуження, представлений у вигляді строкової конструкції If...Then:

```
        If x < a Then y = Math.E : v = 1 Else y = Math.PI : v = 2
        MsgBox("ОТВЕТ:   x=" & x & ";   y=" & y & Chr(13) &
"Вычисления прошли по " & v & "-й ветви ", , "Ветвление: 2
ветви")
    End Sub
```

Тестування алгоритму

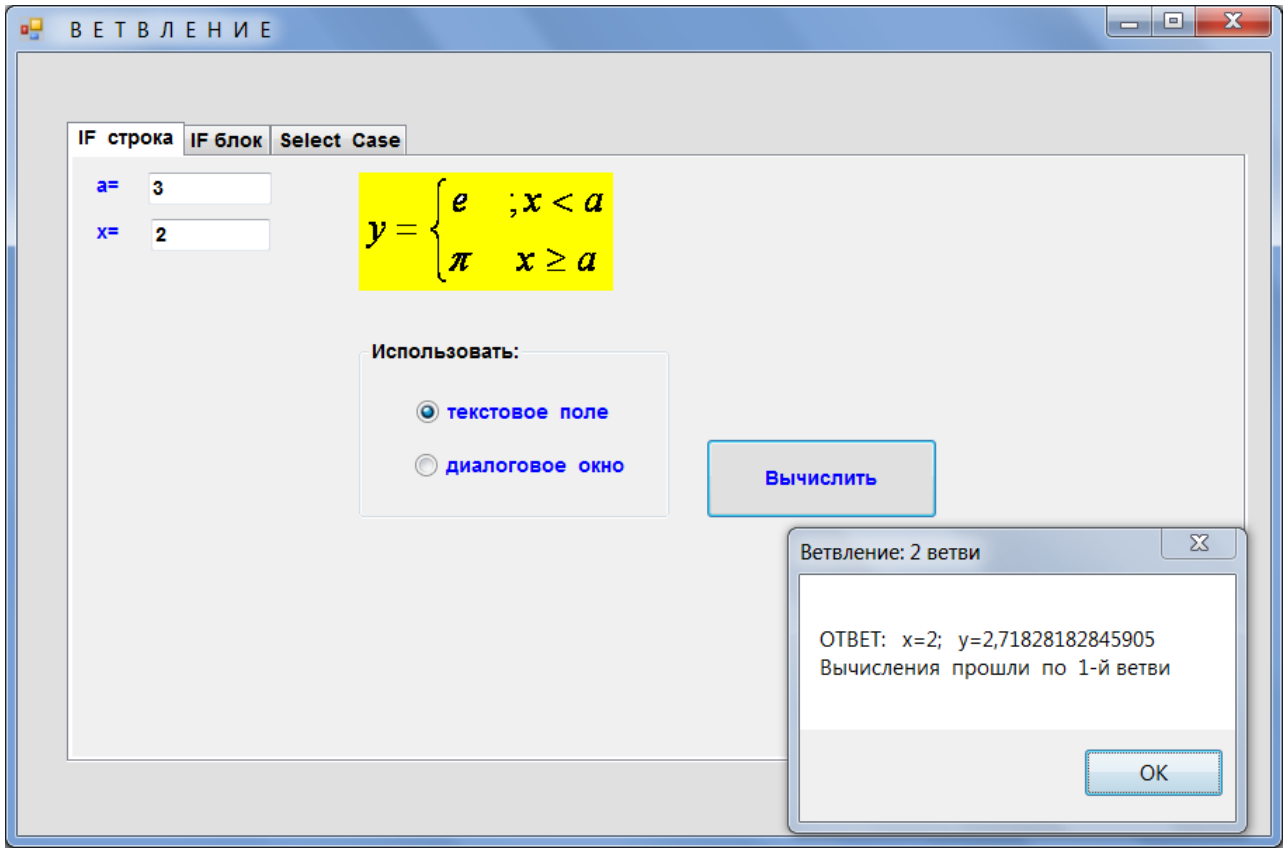


Рисунок 3.1 – Форма задачі «Алгоритм розгалуження 2 гілки»

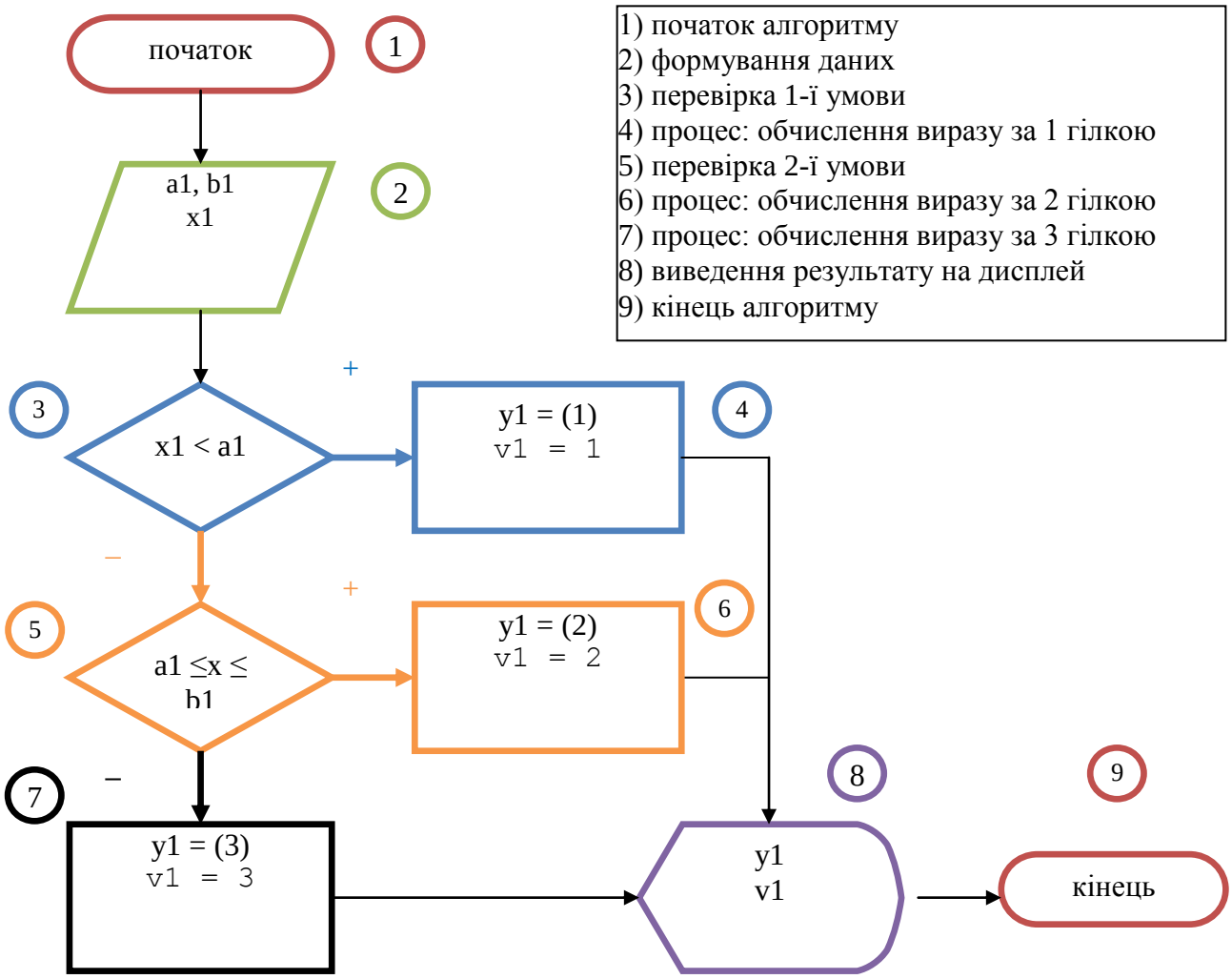
3.4 Розв'язок задачі «Алгоритм розгалуження 3 гілки» (If-блок)

Умова. Обчислити значення кусково-аналітичної функції 3 гілки з використанням конструкції (If-блок)

$$Y = \begin{cases} \ln(x), & x < a \\ \arccos(x), & a \le x \le b \\ \pi, & x \ge b \end{cases}$$

Стислі відомості про елементи управління, які призначенні для створення інтерфейсу користувача наведені у додатку Б.

Принципова блок-схема алгоритму



$$y1 = \begin{cases} (1) = \ln(x) \\ (2) = \arccos(x) \\ (3) = \pi \end{cases}$$

Програмний код алгоритму

```
    Dim a1 As Double = -6
    Dim b1 As Double = 5
    Dim x1 As Double = 22
    Dim Y1 As Double
    Dim v1 As Double
    Private Sub Button3_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Button3.Click
        ListBox1.Items.Clear()
```

```
        On Error GoTo 999 'в случае ошибки управление передается
метке 999
        ComboBox1.Items.Add(ComboBox1.Text) 'Если  надо  добавить
новое  значение
        a1 = ComboBox1.Text 'Задание значения узла A
        ComboBox2.Items.Add(ComboBox2.Text) 'Если  надо  добавить
новое  значение
        b1 = ComboBox2.Text ' Задание значения узла B
        ComboBox3.Items.Add(ComboBox3.Text) 'Если  надо  добавить
новое  значение
        x1 = ComboBox3.Text ' Задание значения узла C
```

В наступному блоці реалізується алгоритм вибору гілки згідно з умовою розгалуження, представлений у вигляді блокової конструкції If...Then:

```
        If x1 < a1 Then
            Y1 = Log(x1) : v1 = 1
        ElseIf x1 >= a1 And x1 <= b1 Then
            Y1 = Acos(x1) : v1 = 2
        Else
            Y1 = PI : v1 = 3
        End If
```

```
        'Вывод результата в ListBox1
        ListBox1.Items.Add("Получили  значение  функции:")
        ListBox1.Items.Add("Y=" & Y1)
        ListBox1.Items.Add("при  значении  аргумента:")
        ListBox1.Items.Add("x=" & x1)
        ListBox1.Items.Add("Вычисления  прошли  по " & v1 & "-й
ветви") : GoTo 1000
999:    ListBox1.Items.Add("Проверь  правильность ")
        ListBox1.Items.Add("   ввода  данных")
1000:
    End Sub
    Private Sub Button4_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Button4.Click
        'Запускающая  кнопка
        If CheckBox1.Checked = True Then
            Button3.Visible = True
            Button4.Visible = False
            'Layout (пер с англ. - схема размещения)
            TableLayoutPanel1.Visible = True
        End If
'***********  IF  3  ветви (Стандартный вариант) **********
```

В наступному блоці реалізується алгоритм вибору гілки згідно з умовою розгалуження, представлений у вигляді блокової конструкції If...Then:

```
        If x1 < a1 Then
            Y1 = Log(x1) : v1 = 1
        ElseIf x1 >= a1 And x1 <= b1 Then
            Y1 = Acos(x1) : v1 = 2
        Else
```

```
            Y1 = PI : v1 = 3
        End If
        Dim Ответ As Object = MsgBox("ОТВЕТ:   x=" & x1 & ";   y="
& Y1 & Chr(13) & _
                "Вычисления прошли по " & v1 & "-й ветви",
MsgBoxStyle.Exclamation + MsgBoxStyle.YesNo)
        If Ответ = MsgBoxResult.No Then
            Dim Ответ1 As Object = MsgBox("ПРОДОЛЖАТЬ РАБОТУ ?",
MsgBoxStyle.Question + MsgBoxStyle.YesNo)
            If Ответ1 = MsgBoxResult.No Then
                TabControl1.SelectTab(2)
            End If
        End If
    End Sub
```

Тестування алгоритму

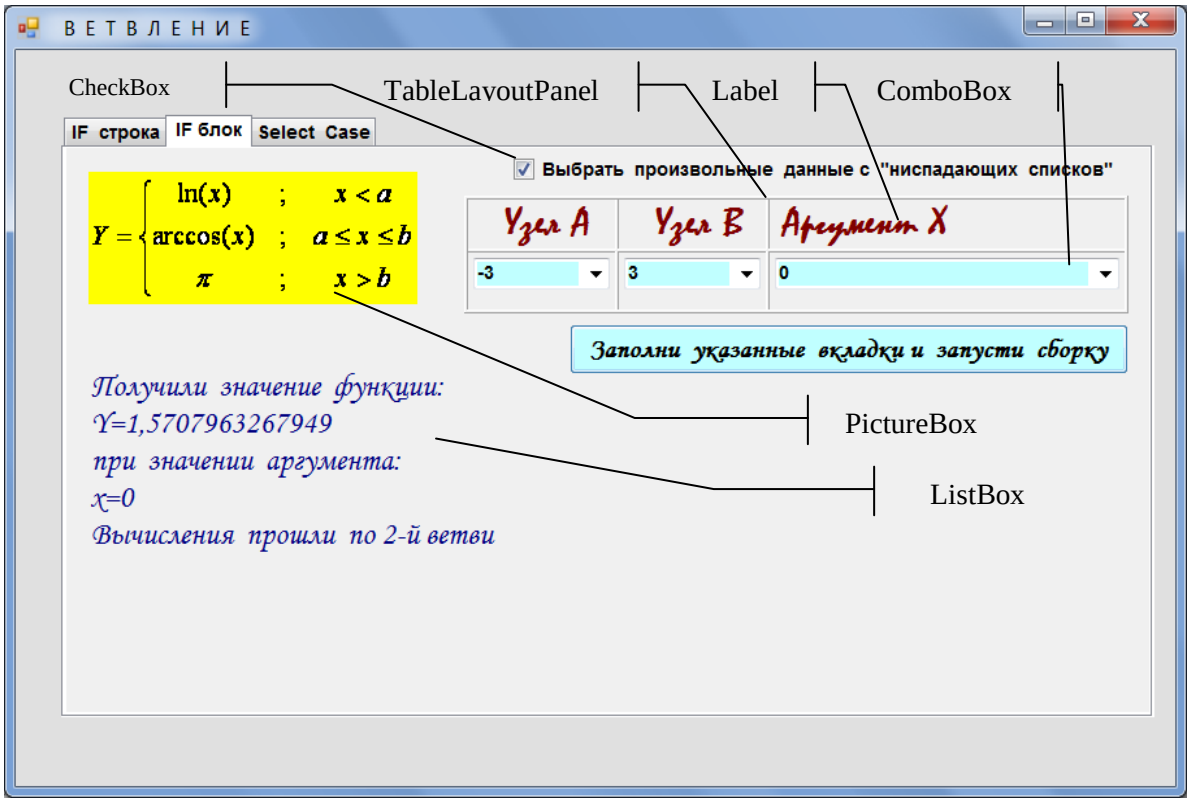


Рисунок 3.2 – Форма задачі «Алгоритм розгалуження 3 гілки»

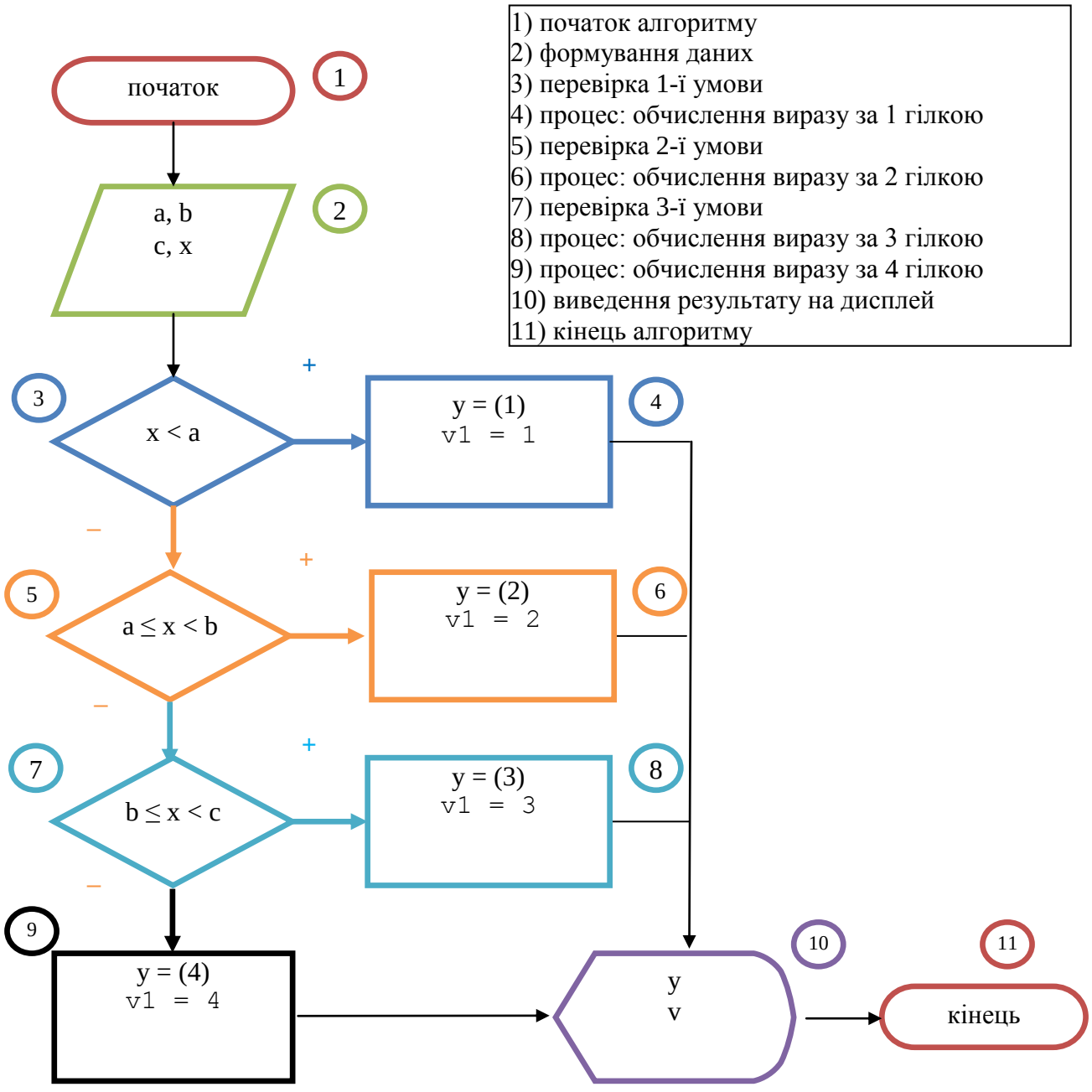
3.5 Розв'язок задачі «Алгоритм розгалуження 4 гілки» (Select Case)

Умова. Обчислити значення кусково-аналітичної функції 4 гілки з використанням конструкції (Select Case).

$$y = \begin{cases} \sin(x),\ x < a \\ \cos(x),\ a \le x < b \\ tg(x),\ b \le x < c \\ ctg(x),\ x \ge c \end{cases}$$

Стислі відомості про елементи управління, які призначені для створення інтерфейсу користувача наведені у додатку Б.

Принципова блок-схема алгоритму



$$y = \begin{cases} (1) = \sin(x) \\ (2) = \cos(x) \\ (3) = tg(x) \\ (4) = ctg(x) \end{cases}$$

Програмний код алгоритму

```
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Button2.Click
        Label10.Visible = False : Label11.Visible = False
        Label12.Visible = False : Label13.Visible = False
        Dim a As Double, b As Double, c As Double, v As Integer, x
As Double
        a = TextBox3.Text : b = TextBox4.Text : c = TextBox5.Text
        x  =  InputBox("Введите   значение   аргумента",  "Окно
ввода", -5)
        Select Case x
            Case Is < a
                y = Sin(x) : v = 1 : Label10.Text = "y=" &
Round(y, 5) : Label10.Visible = True : qqq = Label10
            Case a To b And x <> b
                y = Cos(x) : v = 2 : Label11.Text = "y=" &
Round(y, 3) : Label11.Visible = True : qqq = Label11
            Case b To c And x <> c
                y = Tan(x) : v = 3 : Label12.Text = "y=" &
Round(y, 3) : Label12.Visible = True : qqq = Label12
            Case Else
                y = 1 / Tan(x) : v = 4 : Label13.Text = "y=" &
Round(y, 5) : Label13.Visible = True : qqq = Label13
        End Select
    End Sub
#Region "МЕРЦАЮЩЕЕ ПОЛЕ"
    Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Timer1.Interval = 100
        Timer1.Enabled = True
        Timer2.Interval = 150
        Timer2.Enabled = True
        qqq = Label14
        '---------------
        Button3.Visible = False
        'Layout-ПЕРЕВОД:Схема размещения
        TableLayoutPanel1.Visible = False
    End Sub
    Sub Timer1_Tick(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
        With qqq
            .BackColor = Color.Blue
            .ForeColor = Color.Aqua
            .Font = New System.Drawing.Font("Arial", 8)
```

```
        End With
    End Sub
    Sub Timer2_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer2.Tick
        With qqq
            .BackColor = Color.Aqua
            .ForeColor = Color.Blue
            .Font = New System.Drawing.Font("Arial Black", 8)
        End With
    End Sub
#End Region
```

Тестування алгоритму

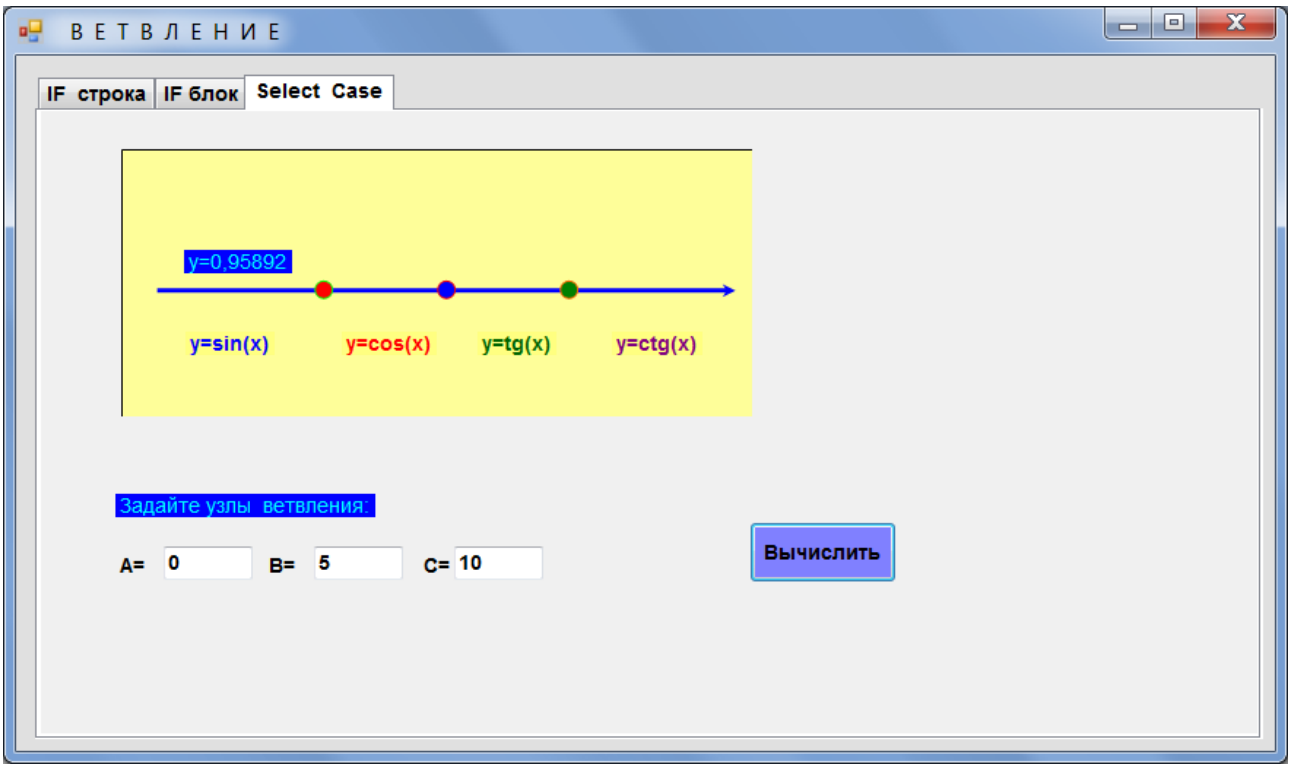


Рисунок 3.3 – Форма задачі «Алгоритм розгалуження 4 гілки»

4 ЦИКЛІЧНІ АЛГОРИТМИ У VB

4.1 Цикли без вкладення зі заздалегідь відомою кількістю повторень (For…Next)

Цикл For…Next використовується, коли наперед відоме початкове та кінцеве значення лічильника. Повторює виконання групи інструкцій зазначене число раз. Синтаксис запису циклу For…Next має вигляд:

```
    For <лічильник> [As <тип даних>] = <початок> To <кінець>
[Step крок>]
          [інструкції]
    [Exit For]
          [інструкції]
    Next [лічильник]
```

лічильник	обов'язковий. Числова змінна, використовувана в якості лічильника циклу;
початок	обов'язковий. Початкове значення змінної лічильника;
кінець	обов'язковий. Кінцеве значення змінної лічильника;
крок	необов'язковий. Значення, на яке змінюється лічильник при кожному виконанні тіла циклу. Якщо це значення не задане, за замовченням крок дорівнює одиниці;
інструкції	необов'язковий. Одна або декілька інструкцій між For і Next, що виконуються зазначене число раз.

Приклад використання циклу For…Next без вкладень:

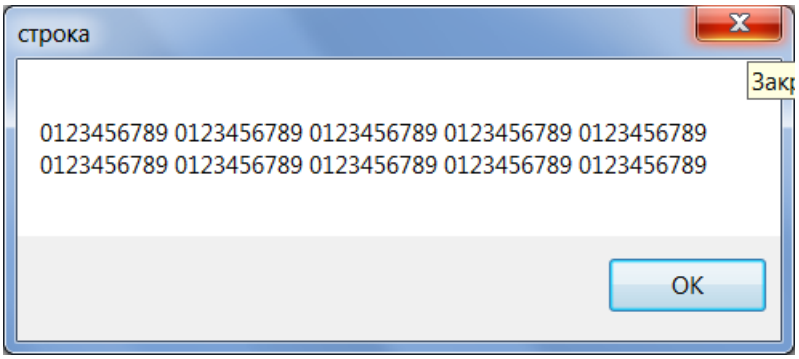
```
Dim число As Integer
Dim сумма As Integer
     For число = 0 To 10
         сумма = число + 1
     Next число
     MsgBox(число, MsgBoxStyle.Information, "Сообщение")
```

4.2 Цикли з вкладенням зі заздалегідь відомою кількістю повторень (For…Next)

Допускається організація вкладених циклів For... Next (один цикл For... Next розташовується в середині іншого). Лічильник кожного циклу повинен мати унікальне ім'я.

В наступному прикладі реалізований алгоритм формування та виведення на екран строк, які містять цифри від 0 до 9. Алгоритм заснований на застосуванні вкладених циклічних конструкцій For... Next.

```
Dim слова, цифра As Integer
Dim строка As String = ""
For слова = 10 To 1 Step -1
    For цифра = 0 To 9
        строка &= CStr(цифра)'
    Next цифра
    строка &= " "
Next слова
MsgBox(строка, , "строка")
```



4.3 Цикли з невідомою заздалегідь кількістю повторень (While … End While)

Оператор While завжди перевіряє умову до виконання циклу. Виконання циклу продовжується доки значення умови залишається True (Істина). Якщо умова дорівнює False (Хибність) при першому вході в цикл, вона не виконається жодного разу. Синтаксис запису має вигляд:

```
While <умова>
     [<інструкції>]
     [ Exit While ]
     [<інструкції>]
End While
```

Приклад використання циклу While … End While. Алгоритм реалізує накопичення значень у змінній <счетчик>, та виведення на екран відповідно цьому інформації про кількість ітерацій.

```
Dim счетчик As Integer = 0
While счетчик < 20
    счетчик = счетчик + 1
End While
MsgBox("Цикл прошел " & CStr(счетчик) & "раз")
```

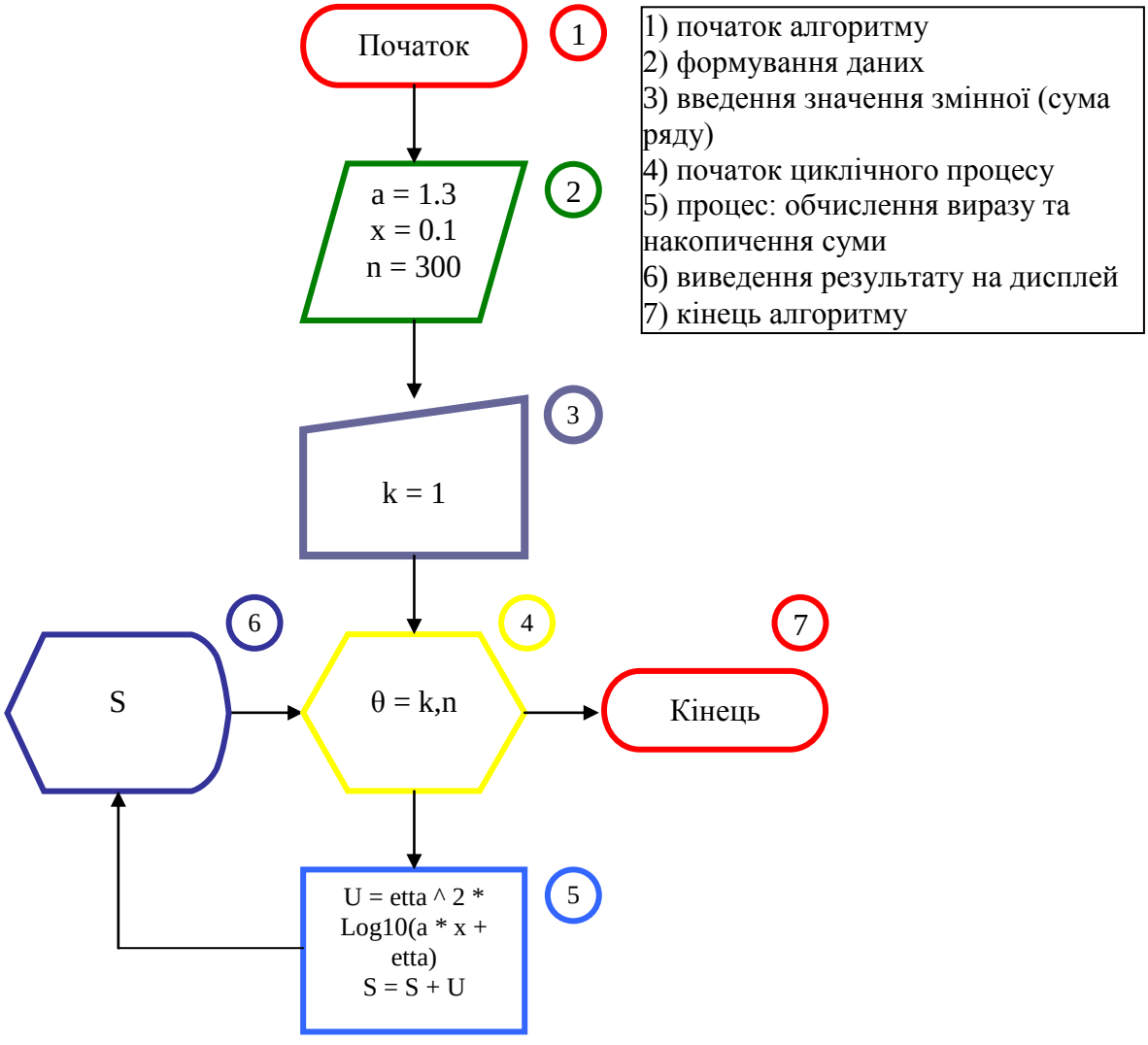
4.4 Розв'язок задачі «Одинарна сума / добуток числового ряду» (конструкція For…Next без вкладення)

Умова

$$\sum_{\theta=k}^{n} \theta^2 \lg(ax+\theta) \quad a=1.3 \quad x=0.1 \quad n=300$$

Стислі відомості про елементи управління, які призначені для створення інтерфейсу користувача наведені у додатку Б.

Принципова блок-схема алгоритму



Програмний код алгоритму

```
Dim etta As Integer
        Dim a As Double = 1.3
        Dim x As Double = 0.1
        Dim n As Integer = 300 ' смотри (.Maximum)
        Dim k As Integer = 1, S As Double, U As Double
        tbxInf0.Text = ""
        tbxInf0.Visible = True
        ProgressBar1.Visible = True
        tbxInf0.ScrollBars = ScrollBars.Horizontal
        tbxInf0.ScrollBars = ScrollBars.Vertical
        tbxInf0.Multiline = True ' многоуровневое  поле
        ' ________ Работаем с  ProgressBar _____________________
        With ProgressBar1
            .Maximum = n ' ВНИМАНИЕ:(.Maximum)=реальному  кол-ву
итераций
            .Minimum = 0
            .Step = 1
            .Value = 0
        End With
        k = InputBox("Введите  начальное  значение  счетчика",
"Определяет  нижний  индекс  суммирования", 1)
```

В наступному блоці реалізується циклічний алгоритм без вкладення, обчислення членів ряду *U*, накопичення їх суми, та виведення результату в текстове поле:

```
        For etta = k To n
            ProgressBar1.PerformStep() 'Step-Приращение, на
которое каждый вызов
            'метода PerformStep увеличивает текущее положение
индикатора выполнения
            tbxInf0.Text = tbxInf0.Text & vbNewLine & "№" & etta &
":  U=" & U & "  S=" & S
            U = etta ^ 2 * Log10(a * x + etta)
            S = S + U
        Next
```

В наступному блоці реалізується функціональність елементу управління ColorDialog, який задає колір заливання та шрифту текстового поля tbxInf0:

```
        If cbxBGround.Checked = True Then
            If ColorDialog1.ShowDialog() = DialogResult.OK Then
                tbxInf0.BackColor = ColorDialog1.Color
            End If
        End If
        ' _______  Работа  с  ColorDialog1 _________________
        If cbxColorFont.Checked = True Then
            If ColorDialog1.ShowDialog() = DialogResult.OK Then
                tbxInf0.ForeColor = ColorDialog1.Color
            End If
        End If
```

Тестування алгоритму

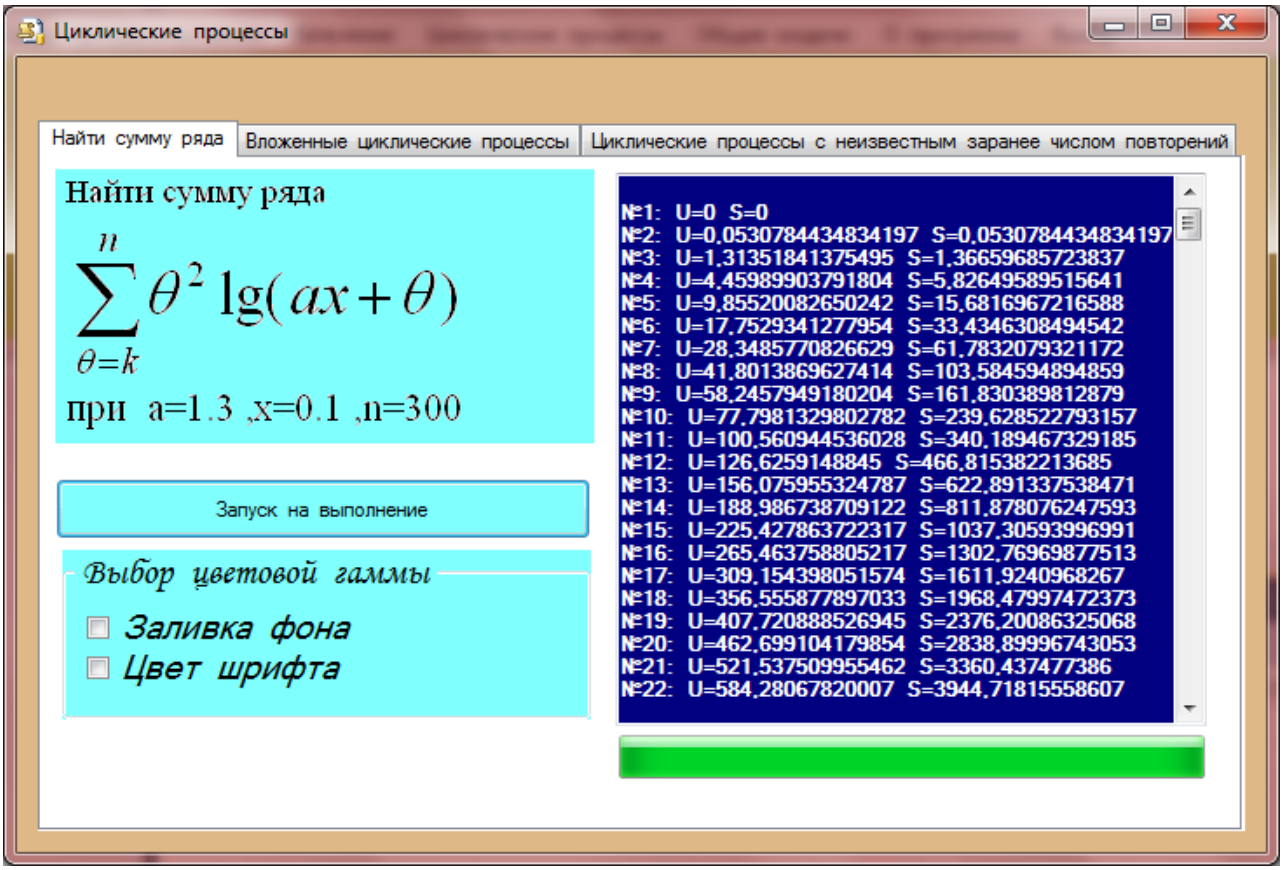


Рисунок 4.1 – Форма задачі «Пошук суми кінцевого ряду»

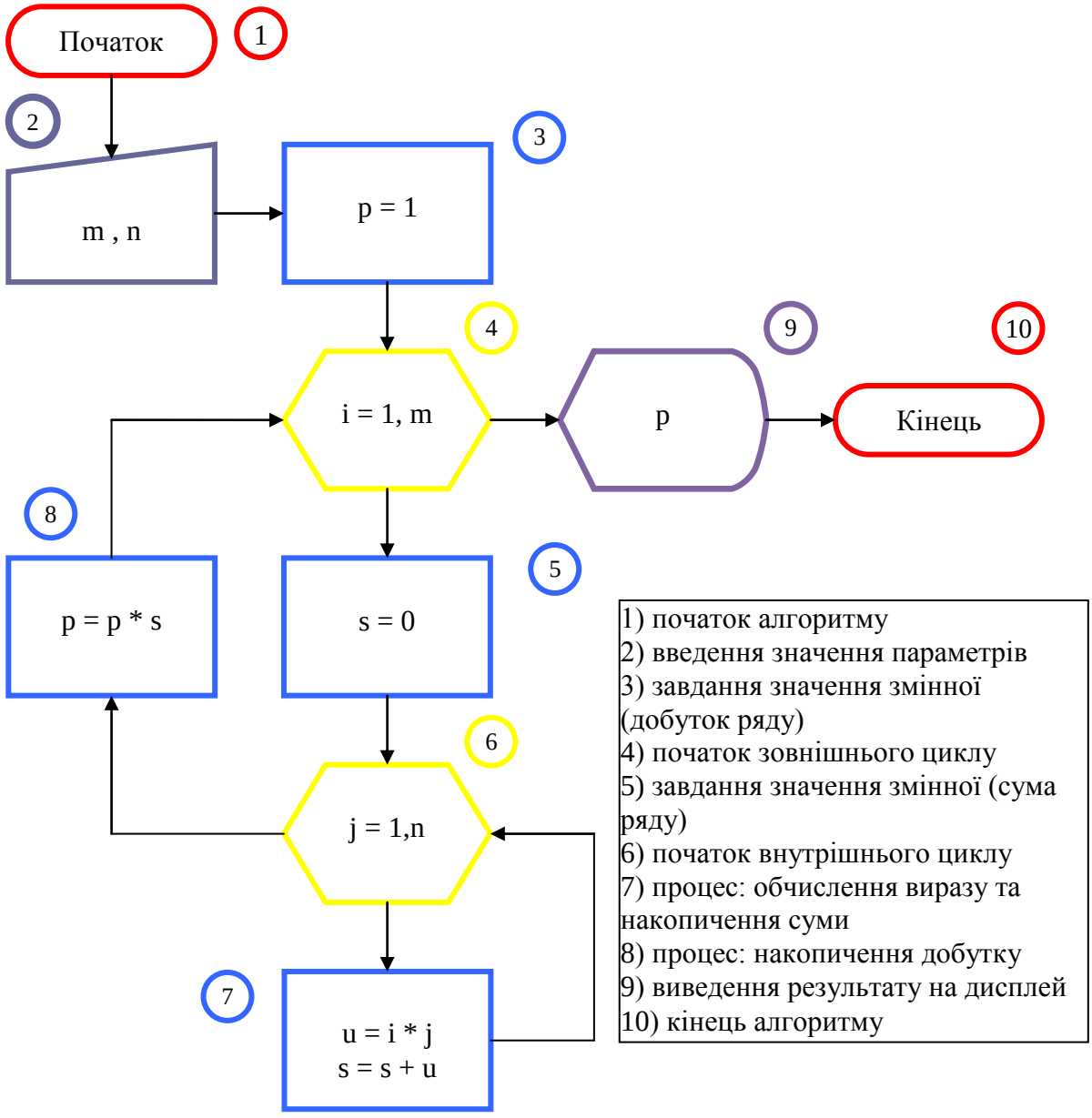
4.5 Розв’язок задачі «Подвійна сума / добуток числового ряду» (конструкція For…Next з вкладенням)

Умова

$$\prod_{i=1}^{2}\sum_{j=1}^{3} ij$$

Стислі відомості про елементи управління, які призначені для створення інтерфейсу користувача наведені у додатку Б.

Принципова блок-схема алгоритму



Програмний код алгоритму

```
Dim i As Integer, m As Integer = tbxIn.Text, n As Integer =
tbxJn.Text
       Dim j As Integer, u As Double, s As Double, p As Double = 1
        With tbxInf1
            .ScrollBars = ScrollBars.Both
            .Multiline = True
            .Font = New System.Drawing.Font("Times New Roman", 14)
            .Text = "Тест  внутреннего  цикла:"
        End With
        With tbxInf2
            .Text = "Тест  внешнего  цикла:"
            .ScrollBars = ScrollBars.Both
            .Multiline = True
```

```
            .Font = New System.Drawing.Font("Times New Roman", 14)
        End With
```

В наступному блоці реалізується циклічний алгоритм із вкладенням. У внутрішньому циклі (за змінної *j*) організовано обчислення членів ряду *U*, накопичення їх суми, та виведення результату в текстове поле. У зовнішньому циклі (за змінної *i*) організовано накопичення добутку суми членів ряду, та виведення результату в текстове поле:

```
        For i = 1 To m : s = 0
            For j = 1 To n
                u = i * j : s = s + u
                tbxInf1.Text = tbxInf1.Text & vbNewLine & "U(" & j
& ")=" & u
            Next
            tbxInf1.Text = tbxInf1.Text & vbNewLine &
"                                "
            tbxInf1.Text = tbxInf1.Text & vbNewLine & "  ЧАСТИЧНАЯ
СУММА: S(" & i & ")=" & s
            p = p * s
            tbxInf2.Text = tbxInf2.Text & vbNewLine & "S(" & i &
")=" & s
        Next
    tbxInf2.Text = tbxInf2.Text & vbNewLine &
"                                "
    tbxInf2.Text = tbxInf2.Text & vbNewLine & "P( ИТОГОВОЕ )=" & p
        If cbxFont.Checked = True Then
            If FontDialog1.ShowDialog =
Windows.Forms.DialogResult.OK Then _
            tbxInf1.Font = FontDialog1.Font
            tbxInf2.Font = tbxInf1.Font
        End If
```

Тестування алгоритму

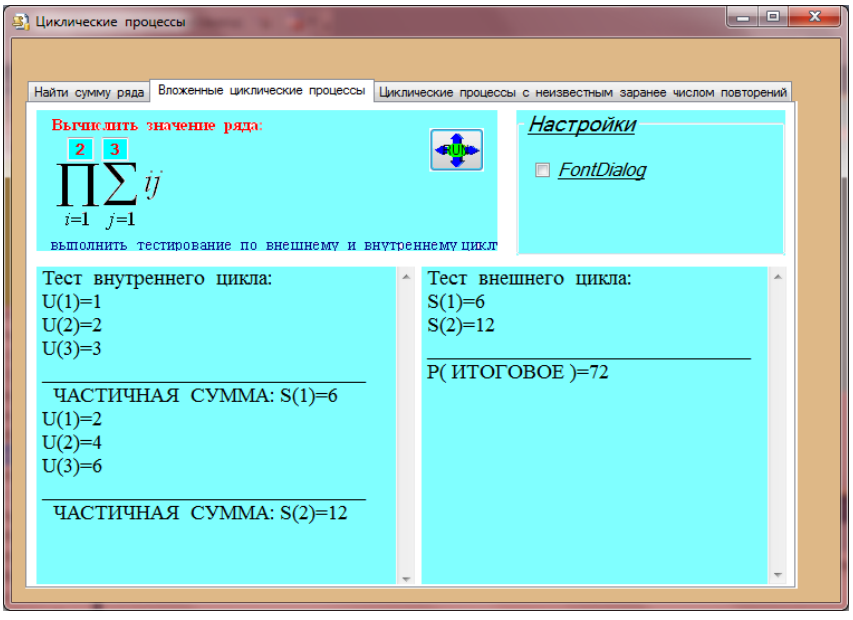


Рисунок 4.2 – Форма задачі «Пошук добутку суми кінцевого ряду»

4.6 Розв'язок задачі «Сума ряду збіжного за Лейбніцем» (конструкція While … End While)

Умова. Знайти суму ряду збіжного за Лейбніцем:

$$S = \sum_{i=1}^{\infty} (-1)^i \frac{x^i}{(2i+1)!}$$

Теорема Лейбніца про збіжність знакозмінних рядів

Якщо:

1. Існує знакозмінний нескінченний числовий ряд:

$$u_1 - u_2 + u_{3...} - u_n + u_{n+1...}$$

2. Члени ряду не зростають за абсолютним значенням:

$$|u_{i+1}| \le |u_i|$$

3. $\lim\limits_{i \to \infty} u_i = 0$,

то ряд сходиться, причому сума ряду не перевищує абсолютного значення першого члена ряду:

$$S < |u_1|$$

Алгоритм знаходження суми ряду з заданою точністю ε. Накопичуємо члени ряду до тих пір, поки черговий член ряду по абсолютному значенню не стане менше ε. На цьому накопичення суми завершуємо. Накопичену суму приймаємо за суму ряду з точністю до ε.

Для реалізації алгоритму необхідно обчислити рекурентний коефіцієнт за формулою:

$$k_{i+1} = \frac{u_{i+1}}{u_i}$$

Таким чином, згідно з умовою:

$$u_{i+1} = (-1)^{i+1} \frac{x^{i+1}}{(2i+3)!}$$

$$u_i = (-1)^i \frac{x^i}{(2i+1)!}$$

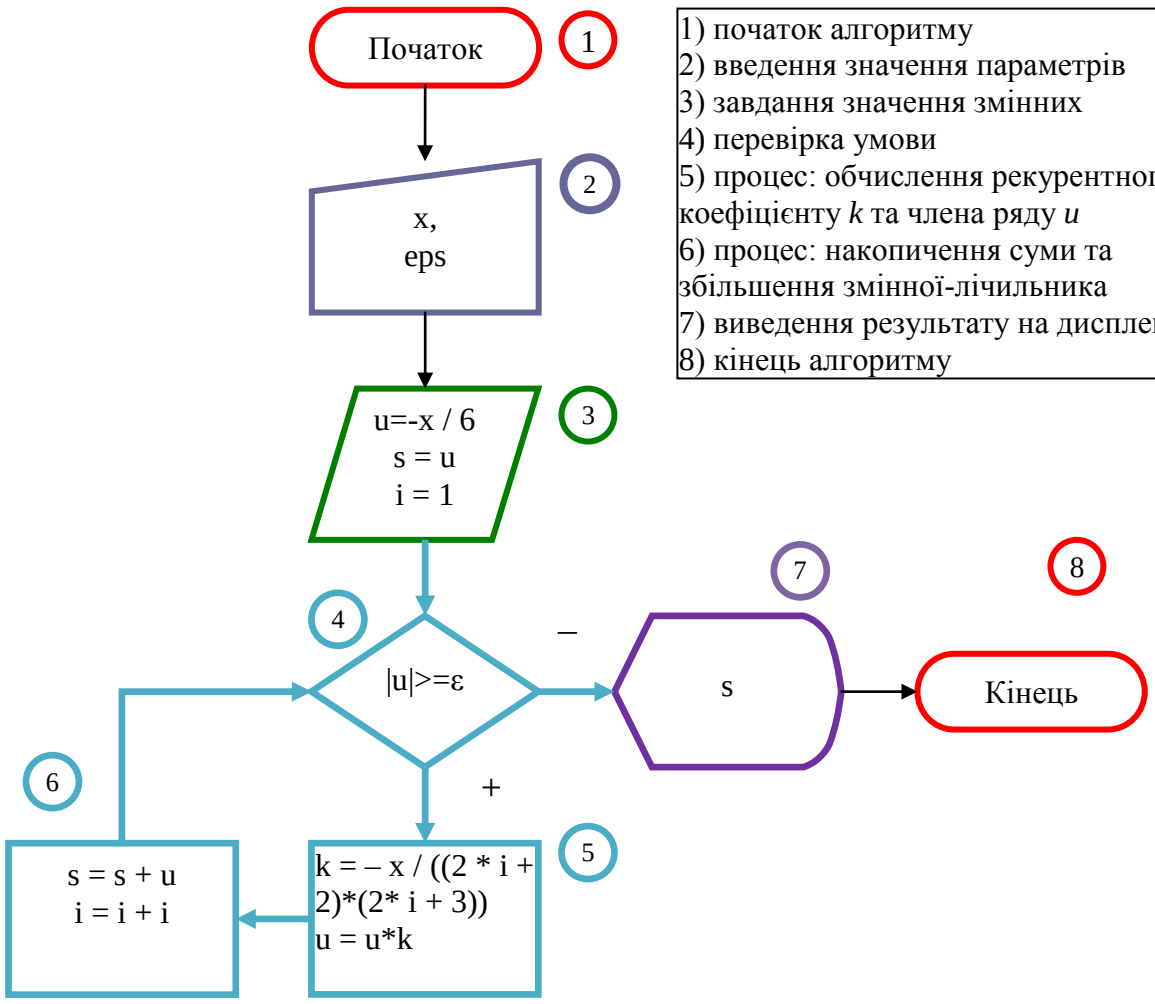
$$k_i = \frac{(-1)^{i+1}\dfrac{x^{i+1}}{(2i+3)!}}{(-1)^{i}\dfrac{x^{i}}{(2i+1)!}} = -\frac{x}{(2i+2)(2i+3)}$$

Також необхідно обчислити перший член ряду, за формулою:

$$u_1 = (-1)^1 \frac{x^1}{(2\cdot 1+1)!} = -\frac{x}{3!} = -\frac{x}{6}$$

Стислі відомості про елементи управління, які призначені для створення інтерфейсу користувача наведені у додатку Б.

Принципова блок-схема алгоритму



Програмний код алгоритму

```
'***********  Л  Е  Й  Б  Н  И  Ц ***************************
    Private Sub Button3_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Button3.Click
        tbxInf3.Text = "РЕЗУЛЬТАТЫ  ТЕСТИРОВАНИЯ  МОДЕЛИ:"
        tbxInf3.Visible = True
        Dim eps As Double, x As Double, u As Double, s As Double,
i As Integer, k As Double
        x = Val(tbxX.Text)
        eps = Val(tbxEps.Text)
```

Обчислимо перший член суми, згідно з умовою:

```
        u = -x / 6
        s = u : i = 1
```

В наступному блоці реалізується циклічний алгоритм без вкладення, обчислення членів ряду *U* (за допомогою значення рекурентного коефіцієнту *k*), накопичення їх добутку, та суми. Цикл буде працювати доти, поки кожен із членів ряду *U* буде більше заданого значення *eps* чи буде дорівнювати йому.

```
        While Abs(u) > eps
            tbxInf3.Text = tbxInf3.Text & vbNewLine & "U(" & i &
")=" & u
            k = -x / ((2 * i + 2) * (2 * i + 3))
            u = u * k
            s = s + u
            i = i + 1
        End While
        tbxInf3.Text = tbxInf3.Text & vbNewLine & "_____________"
        tbxInf3.Text = tbxInf3.Text & vbNewLine & "Итоговая  сумма
с  заданной  точностью:" & s
    End Sub
```

Тестування алгоритму

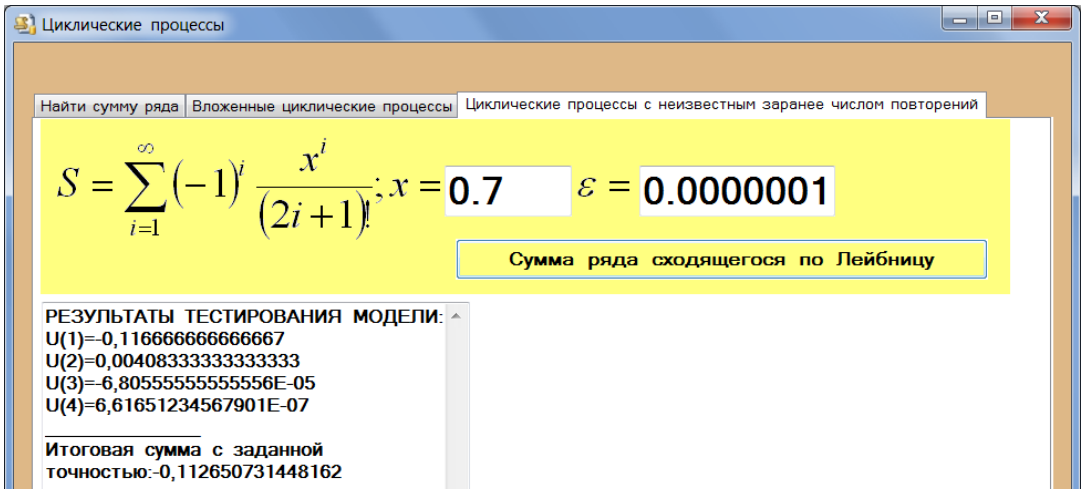


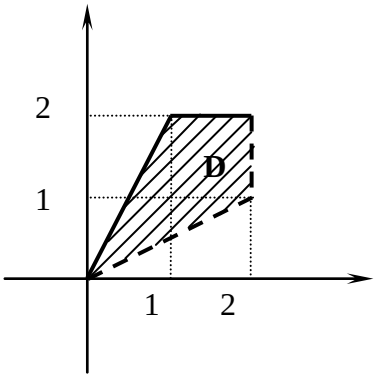
Рисунок 4.3 – Форма задачі «Сума ряду, який сходиться за Лейбніцем»

5 КОМБІНАЦІЯ АЛГОРИТМІВ У VB

5.1 Розв'язок задачі «Перетин кривої та області»

Умова. Перевірити, чи перетинає крива γ задану область *D*. Крива задана параметрично. Вивести в таблицю усі точки кривої. Червоним кольором відмітити точки, в яких крива не перетинає область. Прапорцями відмітити точки перетину.

$$\gamma : \begin{cases} x = \sin t \\ y = \cos t \end{cases} \qquad t \in [0, 2\pi],\ \Delta t = 0{,}1$$



Задаємо область системою алгебраїчних нерівностей, використовуючи формулу рівняння прямої у загальному вигляді:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} \Rightarrow D : \begin{cases} y > \dfrac{x}{2} \\ y \le 2x \\ y \le 2 \\ x < 2 \end{cases}$$

Примітка. Якщо потрібно, використовуйте загальну формулу окружності:

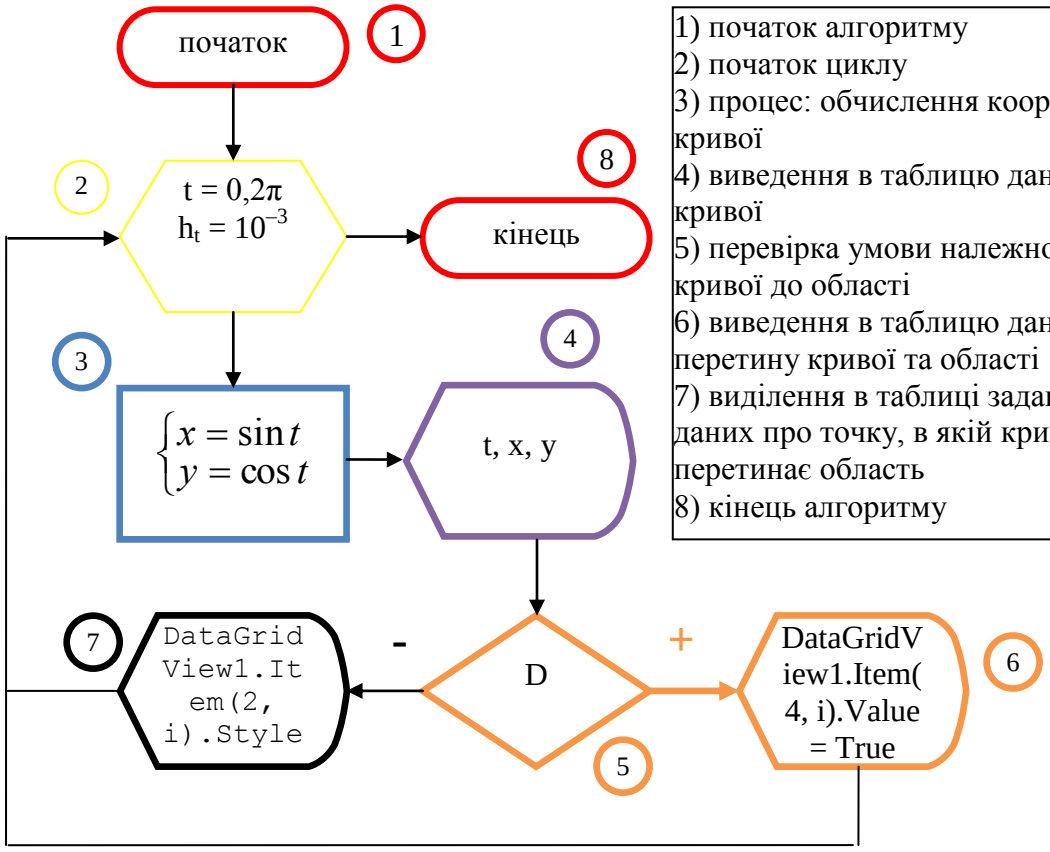
$$(x - x_0)^2 + (y - y_0)^2 = R^2,$$

де (x_0, y_0) – координати центру окружності,

R – радіус окружності.

Стислі відомості про елементи управління, які призначені для створення інтерфейсу користувача наведені у додатку Б.

Принципова блок-схема алгоритму



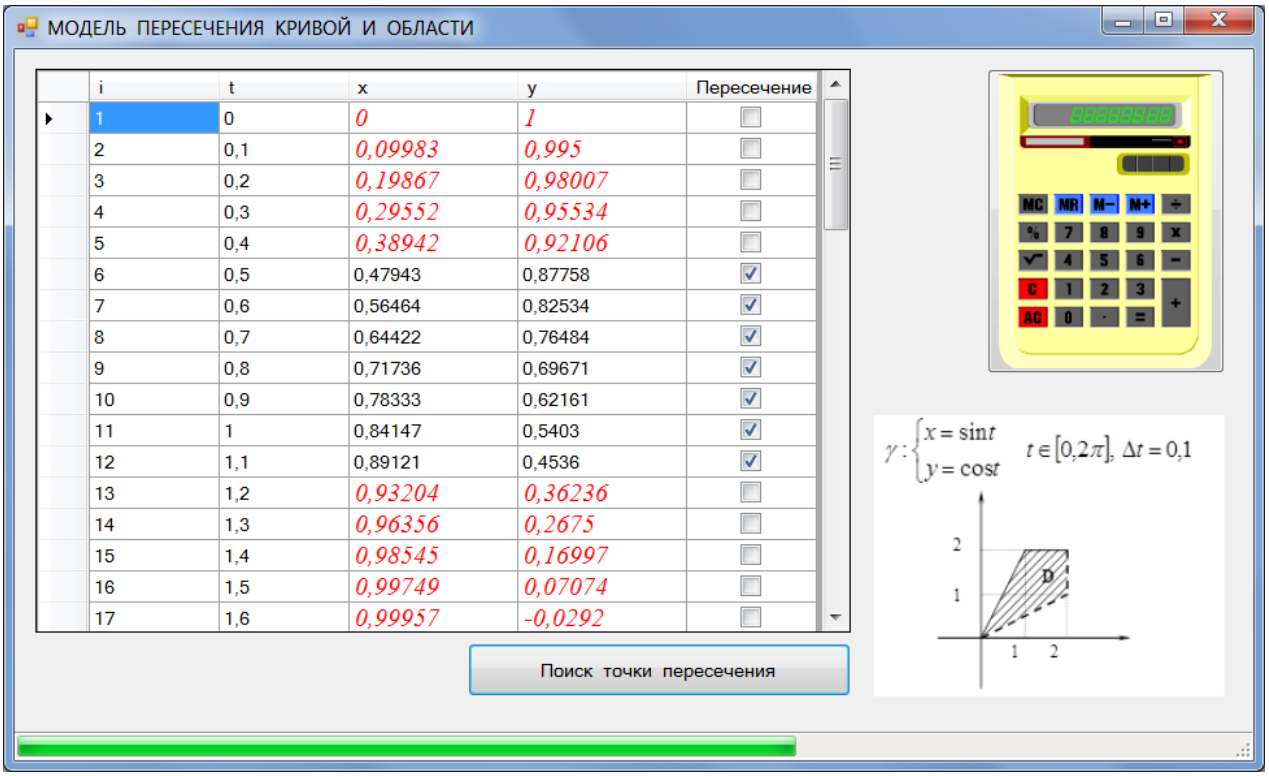
1) початок алгоритму
2) початок циклу
3) процес: обчислення координат точки кривої
4) виведення в таблицю даних про точку кривої
5) перевірка умови належності точки кривої до області
6) виведення в таблицю даних про точку перетину кривої та області
7) виділення в таблиці заданим стилем даних про точку, в якій крива не перетинає область
8) кінець алгоритму

Програмний код алгоритму

```
    Dim objFont As New Font("Times New Roman", 12, FontStyle.Italic)
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim i As Integer = 0, x As Double, y As Double, t As Double
        For t = 0 To 2 * 3.14 Step 0.001
            x = Sin(t)
            y = Cos(t)
            DataGridView1.Rows.Add()
            DataGridView1.Item(0, i).Value = i + 1
            DataGridView1.Item(1, i).Value = Round(t, 3)
            DataGridView1.Item(2, i).Value = Round(x, 5)
            DataGridView1.Item(3, i).Value = Round(y, 5)
            If y > x / 2 And y <= 2 * x And y <= 2 And x < 2 Then
                DataGridView1.Item(4, i).Value = True
            Else
                DataGridView1.Item(2, i).Style.Font = objFont : DataGridView1.Item(3, i).Style.Font = objFont
                DataGridView1.Item(2, i).Style.ForeColor = Color.Red : DataGridView1.Item(3, i).Style.ForeColor = Color.Red
            End If
            i = i + 1
```

```
        Next t
    End Sub
  Private Sub Button2_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Button2.Click
        Dim qqq As Object
        qqq     =     Shell(My.Application.Info.DirectoryPath     &
"\calc.exe",  AppWinStyle.NormalFocus)  'функция  Shell  вызывается
для загрузки приложения calc.exe
    End Sub
```

Тестування алгоритму



i	t	x	y	Пересечение
1	0	0	1	
2	0,1	0,09983	0,995	
3	0,2	0,19867	0,98007	
4	0,3	0,29552	0,95534	
5	0,4	0,38942	0,92106	
6	0,5	0,47943	0,87758	☑
7	0,6	0,56464	0,82534	☑
8	0,7	0,64422	0,76484	☑
9	0,8	0,71736	0,69671	☑
10	0,9	0,78333	0,62161	☑
11	1	0,84147	0,5403	☑
12	1,1	0,89121	0,4536	☑
13	1,2	0,93204	0,36236	
14	1,3	0,96356	0,2675	
15	1,4	0,98545	0,16997	
16	1,5	0,99749	0,07074	
17	1,6	0,99957	-0,0292	

Рисунок 5.1 – Форма задачі «Перетин кривої та області»

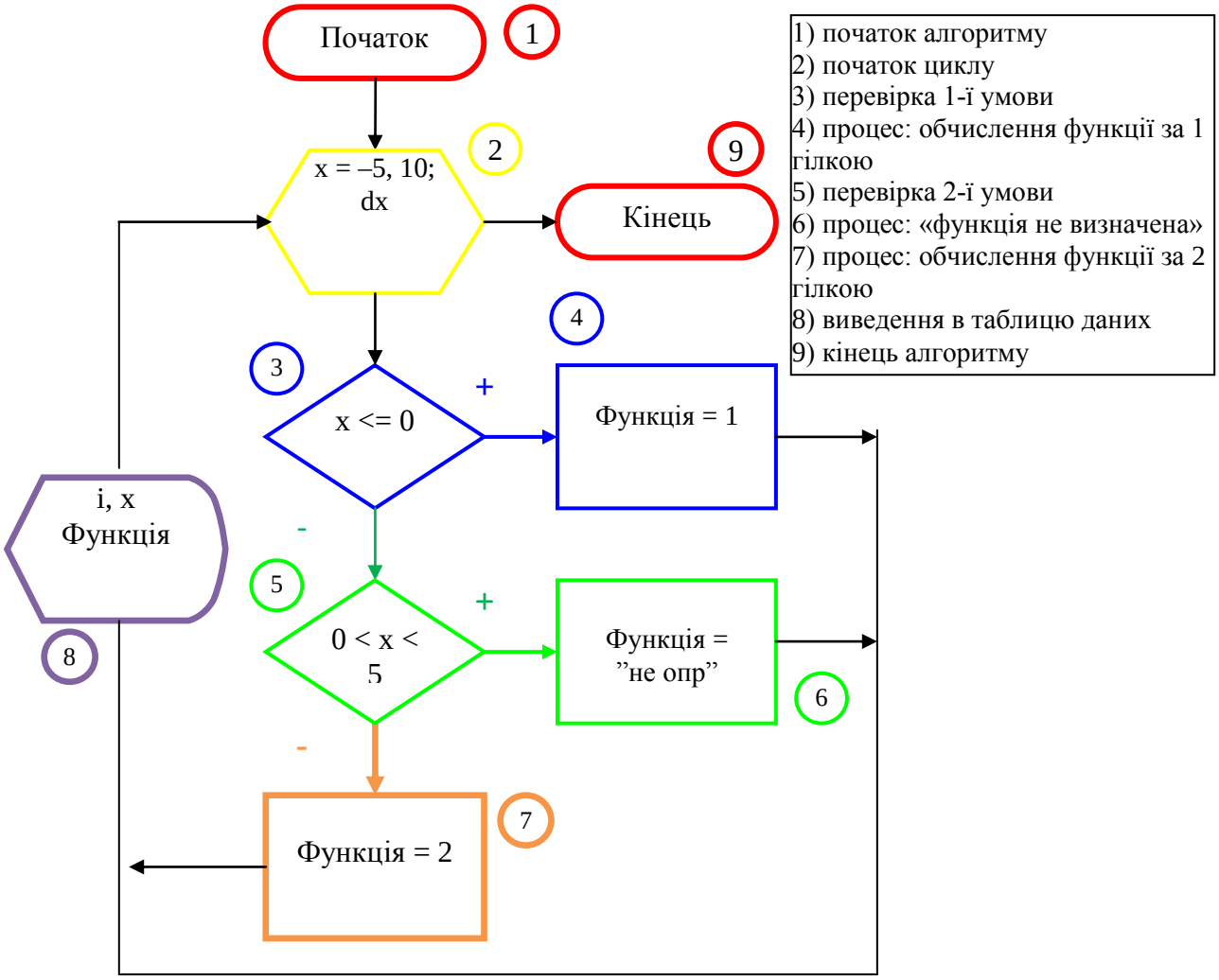
5.2 Розв’язок задачі «Табулювання функції з проколотою областю»

Умова. Протабулювати функцію з проколотою областю

$$y = \begin{cases} 1, & x \le 0 \\ 2, & x \ge 5 \end{cases}$$

Стислі відомості про елементи управління, які призначені для створення інтерфейсу користувача наведені у додатку Б.

Принципова блок-схема алгоритму



Програмний код алгоритму

```
Dim Функція
        Dim шаг As Object, dx As Double, x As Double, i As Integer
        шаг = MsgBox("Хотите ввести шаг ?", vbQuestion + vbYesNo, " Відповідь_ на питання")
        If шаг = vbYes Then
            dx = InputBox("Уведіть крок зміни аргументу", "Введення кроку", 0.51)
        Else
            dx = 0.51
        End If
        DataGridView1.Rows.Add()

        For x = -5 To 10 Step dx
            i = i + 1
            Select Case x
                Case Is <= 0
                    Функція = 1
                Case 0 To 5 And x <> 5
```

```
                Функція = "не определена"
            Case Else
                Функція = 2
        End Select
        DataGridView1.Rows.Add()
        DataGridView1.Item(0, i).Value = i
        DataGridView1.Item(1, i).Value = Math.Round(x, 3)
        DataGridView1.Item(2, i).Value = Функція
    Next x
```

Тестування алгоритму

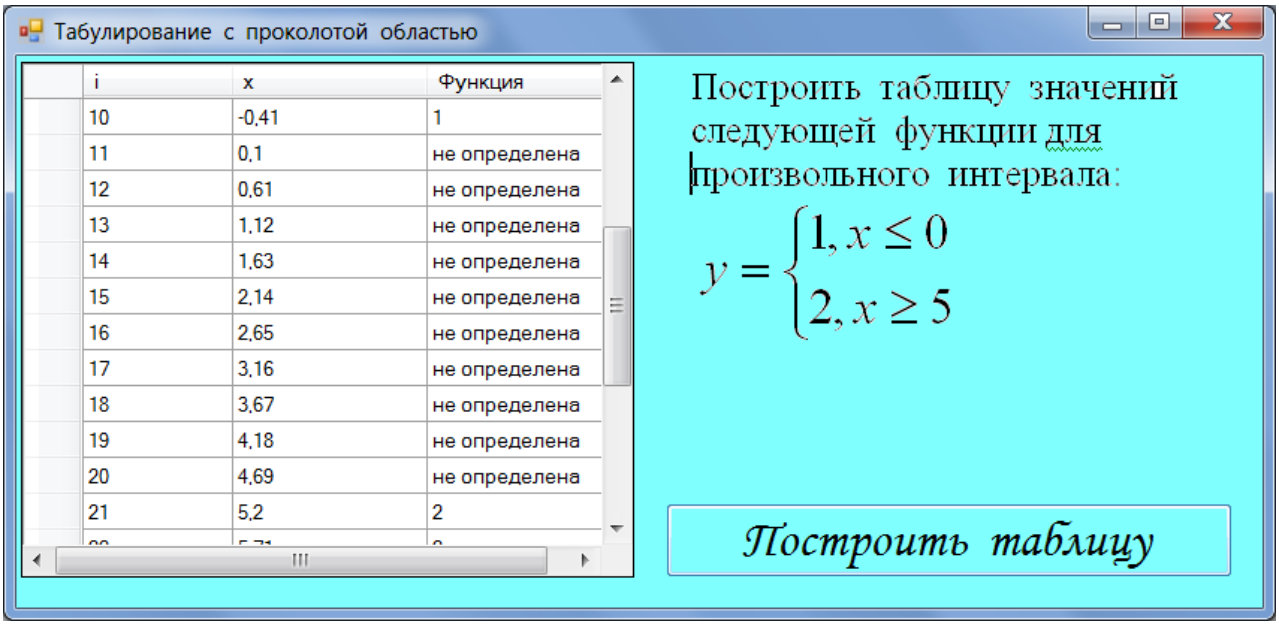


Рисунок 5.2 – Форма задачі «Табулювання функції з проколотою областю»

6 ВИКОРИСТАННЯ МОВИ ПРОГРАМУВАННЯ VBA У MICROSOFT EXCEL

6.1 Введення у Visual Basic for Applications

Visual Basic for Applications (надалі VBA) – це об'єктно-орієнтована мова програмування, спеціально розроблена для запису макросів у додатках Microsoft Office. VBA стала фактично стандартом мови макропрограмування. Вигоди такого підходу очевидні: поява стандартної мови для макропрограмування означає, що незалежно від того, яким додатком ви користуєтеся, досить знати єдиний набір операторів і прийомів програмування. Крім того, це також сприяє більш тісній взаємодії різних додатків, оскільки VBA «знає» команди й об'єкти, які використовуються кожним з додатків. За допомогою VBA можна розробляти комплексні додатки, що одночасно використовують ті або інші компоненти декількох додатків.

Однією із основних переваг VBA є простота використання. Можна взагалі обійтися без усякого програмування: досить включити автоматичний запис дій користувача й в результаті одержати готовий макрос, а потім зіставити йому кнопку на панелі інструментів або нову команду меню, які будуть використовуватися для виклику цього макросу. Або на базі отриманого макросу створити необхідну процедуру, розширивши автоматично створений програмний код власними доповненнями.

VBA є повноцінною мовою програмування, що дозволяє записати не тільки послідовно виконувані користувачем дії, але й реалізувати усі необхідні конструкції мови програмування високого рівня, включаючи різноманітні засоби організації розгалужень, циклів і ведення діалогу користувача. Досить зручний редактор VBA дозволяє не тільки писати й редагувати програми, але й вести їх налагодження.

VBA будується на основі концепції об'єктно-орієнтованого програмування (ООП), таким чином являє собою новий етап розвитку сучасних концепцій побудови мов програмування. Тут одержали подальший розвиток принципи структурного програмування-структуризація програм і даних, модульність і т.д. (короткі відомості про ООП подано у додатку Б.1).

6.2 Об'єктна структура Excel

В VBA для кожного додатку Office визначена множина об'єктів, організованих в ієрархію – об'єктну модель додатку. Тому, працюючи з мовою VBA у середовищі Excel, для досягнення успіху необхідно добре розуміти, що таке об'єкт, і чітко уявляти собі об'єктну модель цього додатку.

Отже, об'єкти в Excel, як і в інших додатках MS Office, утворюють єдину ієрархію об'єктів (рис. 6.1).

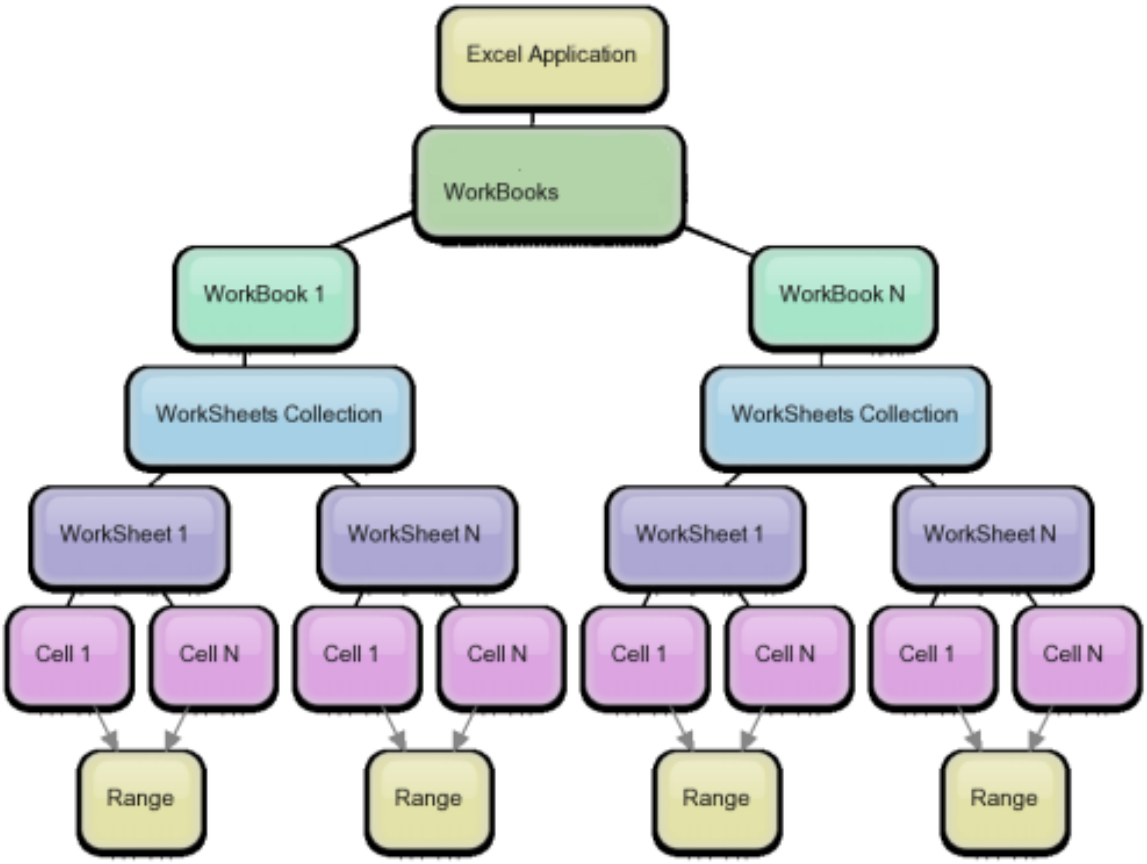


Рисунок 6.1 – Ієрархія об'єктів Excel

На вершині цієї ієрархії перебуває об'єкт Application (Додаток), який містить усі інші об'єкти й відповідає за функціонування всієї програми Excel у цілому. Об'єкт Application виступає сховищем для властивостей і методів, які не підходять для включення в будь-який інший об'єкт, але необхідні для програмного керування Excel. Наприклад, існують властивості об'єкта Application, призначені для керування відновленням екрана й включення попереджень. Існує метод об'єкта Application, що перераховує формули у відкритих книгах.

Оскільки в додатку Excel одночасно можна відкрити кілька робочих книг, існує також і колекція Workbooks, що містить по одному об'єкту Workbook на кожний відкритий файл. Ця колекція належить об'єкту Application.

Найважливіший елемент об'єктної моделі додатка Excel – об'єкт Worksheet, що представляє один робочий аркуш у файлі. Кожний об'єкт Worksheet є частиною колекції Worksheets, яка належить об'єкту Workbook, що представляє файл робочої книги Excel.

Більша частина дій, які користувач (або програма) виконує в Excel, пов'язана з об'єктом Range, що представляють виділений діапазон комірок.

Об'єкт Workbook

В ієрархії Excel відразу після об'єкта Application слідує об'єкт Workbook – робоча книга. Кожний об'єкт Workbook у додатку представляє один файл із розширенням .XLSX (стандартна робоча книга) або .XLSM (стандартна робоча книга з макросами). Коли Excel відкривається не з VBA-Програми, а як самостійний додаток, у ньому за замовчуванням створюється нова робоча книга. При запуску Excel із середовища VBA завдання створення нових робочих книг покладає на програму.

Відкриття й створення робочих книг

Оскільки об'єкти робочих книг Workbook входять у колекцію Workbooks, для створення нової робочої книги слід використовувати метод Add колекції Workbooks.

```
Workbooks.Add(шаблон)
```

Тут аргумент *шаблон* являє собою або ім'я файлу, який буде використаний у якості шаблону при створенні нової робочої книги, або одну з визначених констант xlWBATemplate (повний перелік цих констант можна знайти в Object Browser, помістивши ім'я xlWBATemplate у поле пошуку й клацнувши на кнопці *Найти*). Метод Add повертає посилання на нову робочу книгу, яку можна зберегти в змінній відповідного типу.

```
Dim Нова_книга As Excel.Workbook
Set Нова_книга = Workbooks.Add(шаблон)
```

Відкрити робочу книгу на диску можна за допомогою методу Open колекції Workbooks. Синтаксис виклику цього методу наступний.

```
Workbooks.Open(Ім'я_файлу)
```

Тут обов'язковий аргумент *Ім'я_файлу* задає шлях і ім'я файлу, що відкривається. Запис цього методу в програмному коді може виглядати в такий спосіб.

```
Set Workbookl = Workbooks.Open (Filename:="C:\Data\SaleData.xls")
```

Слід також зазначити, що значення, що повертається, методу *Open* можна безпосередньо застосувати для створення об'єктної змінної з метою звертання до відкритої книги надалі.

Об'єкт Worksheet

Більша частина роботи, яка виконується програмами в середовищі Excel, звичайно пов'язана безпосередньо з робочими аркушами, представленими об'єктами Worksheet у колекції Worksheets. Як уже згадувалося вище, в ієрархії об'єктів Excel клас об'єктів Worksheet слідує безпосередньо після об'єкта Workbook, який є батьківським для колекції Worksheets.

Для доступу до об'єктів Worksheet колекції Worksheets можна скористатися властивістю Item. У якості параметра властивості Item використовується ім'я аркуша або номер, що описує положення аркуша в колекції. Об'єкт Worksheet надає доступ до всіх атрибутів аркуша в Excel. Це стосується як форматування аркуша, так і інших його властивостей. Крім того, об'єкт Worksheet надає події, які можуть оброблятися програмно.

Для додавання нового робочого аркуша в колекцію Worksheets використовується її метод Add.

```
Dim Новий_аркуш As Excel.Worksheet
Set Новий_аркуш = Worksheets.Add(Before, After, Count)
```

Тут усі аргументи є необов'язковими, і якщо вони опущені, то відразу після активного створюється один робочий аркуш. Аргументи *Before* і *After* використовуються для вказівки вже існуючого робочого аркуша, до (або після) якого слід вставити новий робочий аркуш. Для того щоб додати ще кілька робочих аркушів, необхідна їхня кількість задається в аргументі *Count.* Якщо цей аргумент більше одиниці, метод Add повертає посилання на останній з доданих робочих аркушів. Корисно відразу ж після створення призначити новим робочим аркушам імена, щоб згодом по цим іменам можна було звертатися до аркушів у програмі.

```
Новий_аркуш.Name="Обсяги_продажів"
```

До аркуша можна звертатися як по імені, так і за номером в колекції Worksheets. Якщо відоме ім'я аркуша, то його можна використовувати для вибору аркуша з колекції Worksheets. Якщо необхідно відобразити всі елементи колекції Worksheets, наприклад, у циклі For...Next, на кожний аркуш можна посилатися за номером. У межах книги на аркуш Sheet1 можна посилатися за допомогою наступних конструкцій.

```
Activeworkbook.Worksheets("Sheet1")
Activeworkbook.Worksheets(1)
```

Для видалення робочого аркуша використовується метод Delete об'єкта Worksheet.

```
Activeworkbook.Worksheets(Ім'я_аркуша).Delete
```

Тут аргумент *Ім'я_аркуша* – це ім'я робочого аркуша, що видаляється. Звичайно, коли користувач робить спробу видалити робочий аркуш у вікні програми, Excel автоматично виводить вікно запиту для підтвердження операції видалення. Для того, щоб при видаленні аркуша із програми цей запит не виводився, необхідно привласнити властивості Displayalerts об'єкта Application значення False. Однак, після того як непотрібний або тимчасово використовуваний у програмі аркуш буде видалений, обов'язково необхідно змінити значення властивості на True.

```
Application.Displayalerts = False
Activeworkbook.Worksheets("Обсяги_продажів").Delete
Application.Displayalerts = True
```

Об'єкт Range

Об'єкт Range є одним із ключових об'єктів VBA і в ієрархії Excel слідує відразу після об'єкта Worksheet. В Excel об'єкт Range може представляти окрему комірку, рядок або стовпець робочого аркуша й навіть довільний дво- або тривимірний блок комірок робочої книги. VBA-програма може посилатися на будь-яку необхідну їй кількість об'єктів Range, і доступні для програми елементи робочої книги не обмежуються виділенням, виконаним користувачем у цей момент. Інакше кажучи, для впливу програми на яку-небудь область робочої книги її не потрібно вручну виділяти на екрані.

Об'єкт Range є властивістю об'єкта Worksheet і являє собою засіб звертання до деякого діапазону комірок. Діапазон визначається вказівкою адреси першої й останньої вхідних у нього комірок. Використання об'єкта

Range дозволяє створювати більш ефективний код за рахунок спрощеного способу звертання до необхідних елементів робочих аркушів Excel.

Визначення об'єкта Range

В Excel доступно кілька способів ідентифікації діапазону, на який повинен впливати програмний код.

1. Стандартне посилання на комірку. Так званий A1-стиль посилання на комірку є, мабуть, найпростішим способом роботи з об'єктами Range.

У наступному прикладі комірці “B3” активного аркуша привласнюється значення 50.

```
Activesheet.Range("B3").Value = 50
```

Для того, щоб привласнити значення довільному діапазону комірок ("M5:S20"), який належить аркушу "Аркуш2", необхідно виконати наступну команду:

```
Worksheets("Аркуш2").Range("M5:S20").Value = 100
```

2. Властивість Cells об'єкта Worksheets. Цей спосіб особливо зручний при написанні складних VBA-програм, тому що дозволяє визначати діапазон не шляхом явної вказівки фіксованих адрес комірок, а за допомогою використання для його завдання вмісту змінних. Основна ідея полягає в підготовці числових значень координат рядків і стовпців необхідного діапазону, що зберігаються в змінних, імена яких і вказуються при звертанні до даної властивості.

3. Властивість Selection. Коли програма повинна працювати з діапазоном, який у цей момент виділений користувачем у вікні додатка, використовується властивість Selection об'єктів Application або Window.

4. Властивість Activecell. Властивість Activecell (Активна комірка) використовується для доступу до діапазону, що представляє активну комірку зазначеного вікна. При опусканні специфікатора об'єкта вікна (що означає звертання до об'єкта Application) властивість Activecell посилається на поточне активне вікно.

```
Значення_комірки = Activecell.Value 'Читання значення комірки
```

5. Властивості Rows або Columns об'єкта Worksheet. Доступ до діапазону, що включає весь рядок або весь стовпець, здійснюється за допомогою властивостей Rows і Columns об'єкта робочого аркуша відповідно. В операторі вказується номер необхідного стовпця або рядка –

не можна адресувати стовпець, вказавши його літерне позначення, відображуване на екрані в заголовку цього стовпця. Так, у наступному прикладі визначається діапазон, що включає стовпець G, тобто сьомий стовпець, який заповнюється значеннями 20.

```
Workbooks("Звіт.xls").Worksheets("Відомість").Columns(7).Value =20
```

Використання властивості Cells для визначення діапазону

При використанні у виразах без вказівки координат властивість Cells об'єкта Worksheets визначає діапазон, що включає усі комірки даного робочого аркуша. Аналогічна властивість Cells об'єкта Application містить посилання на всі комірки аркуша, активного в цей момент. У цьому випадку властивість Cells може використовуватися окремо, без вказівки в явному вигляді об'єкта Application, тобто Application.Cells еквівалентно Cells.

Якщо обробляється діапазон, що включає лише частину комірок робочого аркуша, при звертанні до властивості Cells буде потрібно вказати числові координати рядків і стовпців (буквена вказівка стовпців не допускається).

Метод Cells має такі типи синтаксису:

– об’єкт.Cells (номерРядка, номерСтовбця)

– об’єкт.Cells (номерКомірки)

У наступному прикладі діапазону, що включає тільки комірку “Е3” привласнюється значення 500.

```
Worksheets("Звіт").Cells(3,5).Value = 500
```

Зверніть увагу на те, що в цьому випадку спочатку вказується координата рядка (3), а потім стовпця (5) – прямо протилежно записи в А1-стилі. У наведеному вище прикладі друге значення в посиланні на комірку дорівнює 5, тобто вказує на п'ятий стовпець – Е. Слід зазначити, що перевага такої така системи запису полягає в тому, що обидві координати є числами, а виходить, їх можна вказувати як значення змінних. Використання змінних для зберігання координат дозволяє динамічно, безпосередньо при виконанні програми, визначати, де перебуває необхідний діапазон, – на основі введених користувачем даних, виходячи з результатів обчислень і т. д. У наведеному нижче прикладі рядок таблиці вибирається залежно від поточного місяця року.

```
Місяць = Month(Now ())
Показник = Worksheets("Річний звіт").Cells(Місяць, 8)
```

Для визначення діапазону на неактивному аркуші необхідно спільно використовувати властивості Range і Cells цього аркуша. Наступний вираз визначає діапазон E3:F4 на робочому аркуші з іменем “Звіт”.

```
Worksheets("Звіт").Range(Worksheets("Звіт").Cells(3,5), _
                         Worksheets("Звіт").Cells(4,6))
```

Наведений вище приклад виглядає досить громіздко, однак його можна спростити за рахунок використання оператора With, що дозволяє уникнути повторних посилань на робочий аркуш. У наступному прикладі до вмісту того ж діапазону застосовується напівжирне форматування.

```
With Worksheets ("Звіт")
     .Range(.Cells(3,5), .Cells (4,6)).Font.Bold = True
End With
```

Зверніть увагу: у цьому випадку наявність крапки перед кожним звертанням до властивості Cells є обов'язковим. Саме крапка вказує на приналежність властивості Cells тому аркушу, який вище в коді був заданий оператором With. При відсутності крапки властивість Cells буде сприйнято як приналежна активному аркушу, що приведе до виникнення конфлікту при визначенні посилання для властивості Range.

6.3 Створення макросів

Макрос – це макрокоманда або набір макрокоманд, за допомогою яких можна автоматизувати різні задачі. Макроси записуються мовою програмування VBA. Макрос завжди можна виконати за допомогою команди *Макросы* на стрічці (група *Код* на вкладці *Разработчик*).

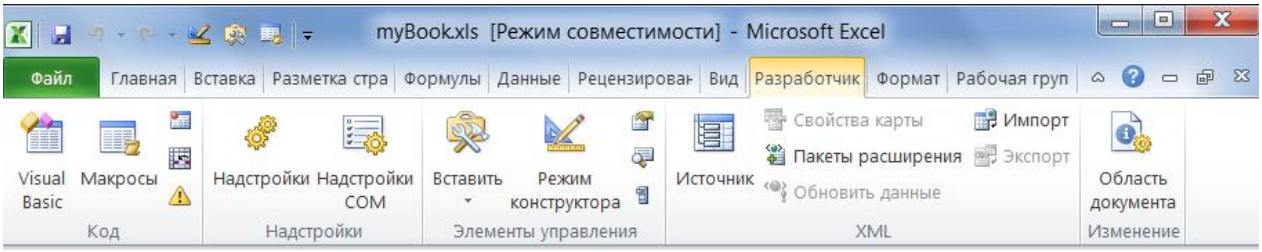


Рисунок 6.2 – Вкладка «Разработчик»

Якщо вкладка *Разработчик* недоступна, виконаєте зазначені нижче дії для її відображення.

1. На вкладці *Файл* виберіть команду *Параметры*, а потім – категорію *Настройка ленты*.

2. У списку *Основные вкладки* встановите прапорець *Разработчик* і натисніть кнопку *ОК*.

Для установки рівня безпеки, що дозволяє виконання всіх макросів, виконаєте наступні дії.

1. На вкладці *Разработчик* у групі *Код* натисніть кнопку *Безопасность макросов*.

2. У категорії *Параметры макросов* у групі *Параметры макросов* встановіть перемикач у положення *Включить все макросы (не рекомендуется, возможен запуск опасной программы)* і натисніть кнопку *ОК*.

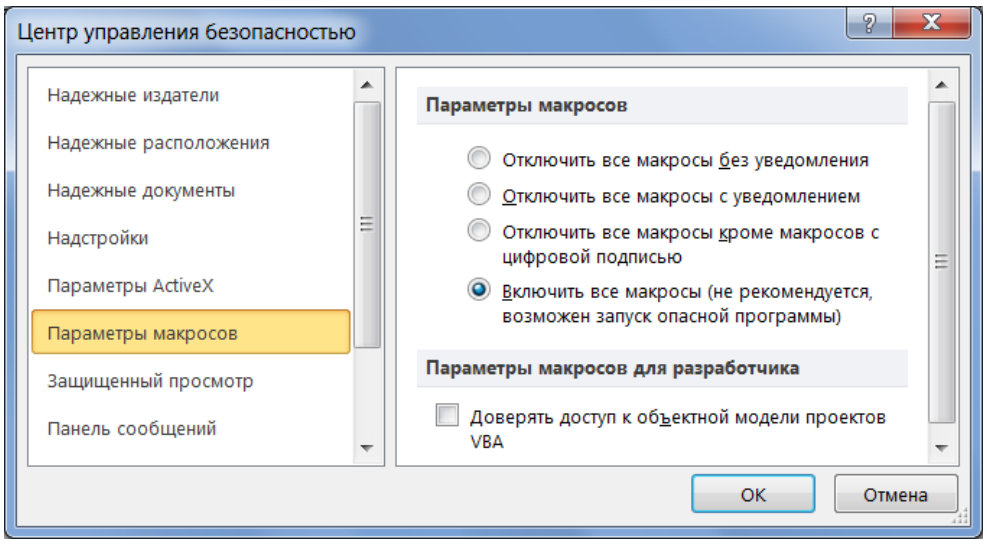


Рисунок 6.3 – Параметри макросів

Запис макросу

Для запису макросу необхідно:

– на вкладці *Разработчик* у групі *Код* натиснути кнопку *Запись макроса* Запись макроса (рис. 6.2);

– встановити необхідні параметри у вікні запису макросів (рис. 6.4) та натиснути кнопку *ОК*;

– виконати необхідні операції в поточній книзі;

– на вкладці *Разработчик* у групі *Код* натиснути кнопку *Остановить запись* Остановить запись ;

– на вкладці *Разработчик* у групі *Код* натиснути кнопку *Макросы* (рис. 6.2) та обрати необхідний макрос у списку для подальших дій (рис. 6.5).

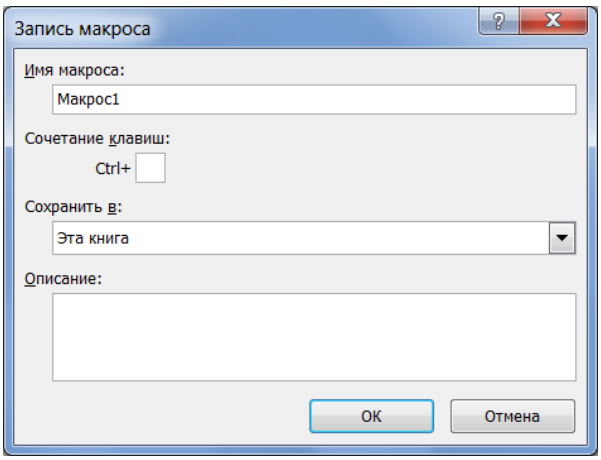


Рисунок 6.4 – Запис макросів

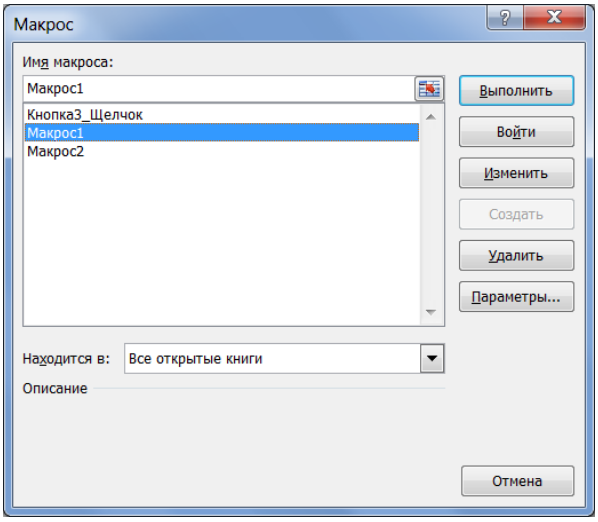


Рисунок 6.5 – Вікно макросів

Макрос та елементи керування

Для створення макросу, що закріплений за деяким елементом керування, необхідно:

– на вкладці *Разработчик* у групі *Элементы управления* натиснути кнопку *Вставить*, обрати необхідний елемент (рис. 6.6) та помістити на аркуш;

– у вікні (рис. 6.7) задати необхідне ім’я макросу та натиснути кнопку *Создать*;

– в створеній процедурі реалізувати необхідний алгоритм (рис. 6.8).

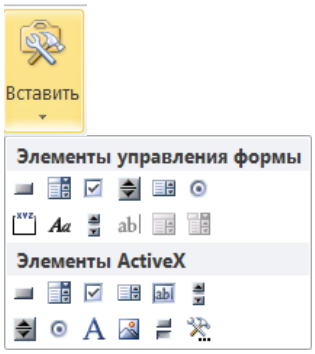


Рисунок 6.6 – Елементи управління

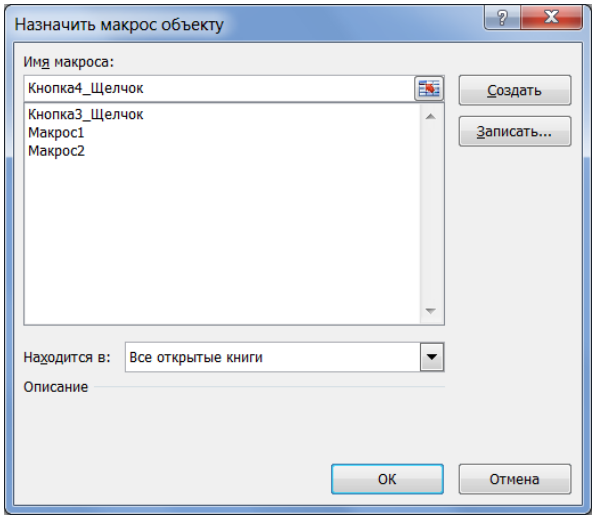


Рисунок 6.7 – Макроси об’єкта

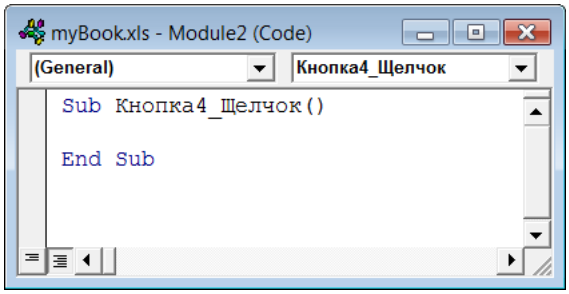


Рисунок 6.8 – Процедура макросу

Створення користувальницької функції

Для створення користувальницької функції у вигляді макрокоманди, необхідно:

– на вкладці *Разработчик* у групі *Код* натиснути кнопку *Visual Basic* (рис. 6.2) та відкрити середовище програмування VBA;

– додати програмний модуль (меню *Insert→Module*);

– створити необхідну функцію у вигляді:

```
Function Ім'яФункції [(Аргумент1, Аргумент2, …)]
    Оператор 1
    Оператор 2
    …
    Ім'яФункції = Значення
End Function
```

яка реалізує деякий алгоритм (рис. 6.9);

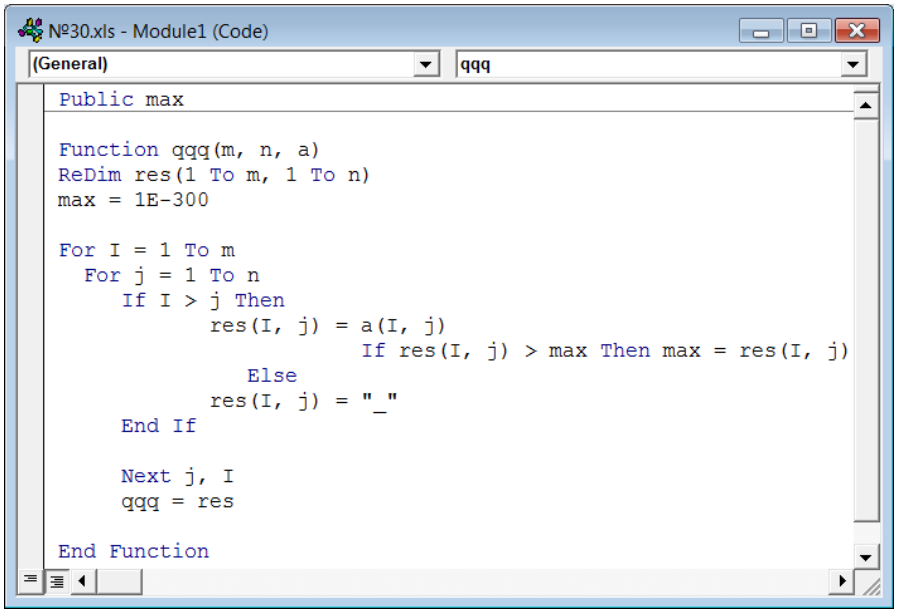


Рисунок 6.9 – Вікно модуля

– за коміркою (діапазоном комірок у разі виконання операцій над масивами) аркуша закріпити функцію, викликавши майстер функцій; перейти у категорію *Определенные пользователем*, обрати необхідну функцію, натиснути кнопку *ОК* (рис. 6.10) та задати необхідні параметри функції.

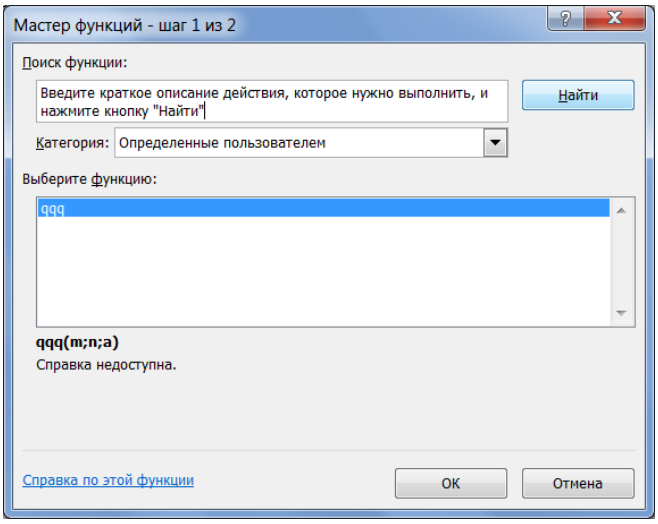


Рисунок 6.10 – Вікно майстра функцій

6.4 Розв’язок типових задач

Задача 1. «Визначник матриці»

Умова. В діапазоні комірок дана довільна матриця (А). За допомогою мови програмування VBA розробити алгоритм перевірки визначника поданої матриці на нерівність 0.

Реалізація розрахунків

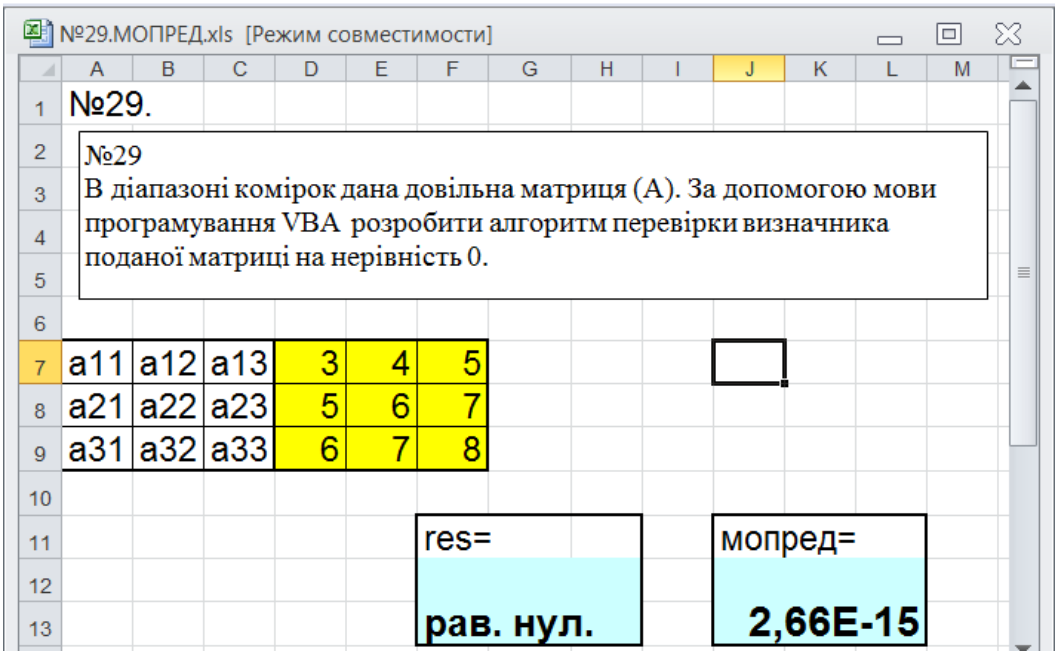


Рисунок 6.11 – Робочий аркуш

Програмний код

```
Public res(1 To 3, 1 To 3)

Function МОПРЕД1(mat)
'Dim res(1 To 3, 1 To 3)
For i = 1 To 3
For j = 1 To 3
res(i, j) = mat(i, j)
Next j, i
determinant = res(1, 1) * res(2, 2) * res(3, 3) + _
          res(3, 1) * res(1, 2) * res(2, 3) + _
          res(2, 1) * res(1, 3) * res(3, 2) - _
          res(3, 1) * res(2, 2) * res(1, 3) - _
          res(2, 1) * res(1, 2) * res(3, 3) - _
          res(1, 1) * res(3, 2) * res(2, 3)
```

```
    If determinant <> 0 Then МОПРЕД1 = "отл.от нул." Else МОПРЕД1
= "рав. нул."
    End Function
```

Задача 2. «Максимальний елемент матриці»

Умова. В діапазоні комірок дано довільний масив (А). За допомогою мови програмування VBA створити користувальницьку функцію, яка дозволяє знайти максимальний елемент, серед елементів масиву, що знаходяться нижче головної діагоналі. Функція повинна обробляти будь-який масив А(M×N)

Реалізація розрахунків

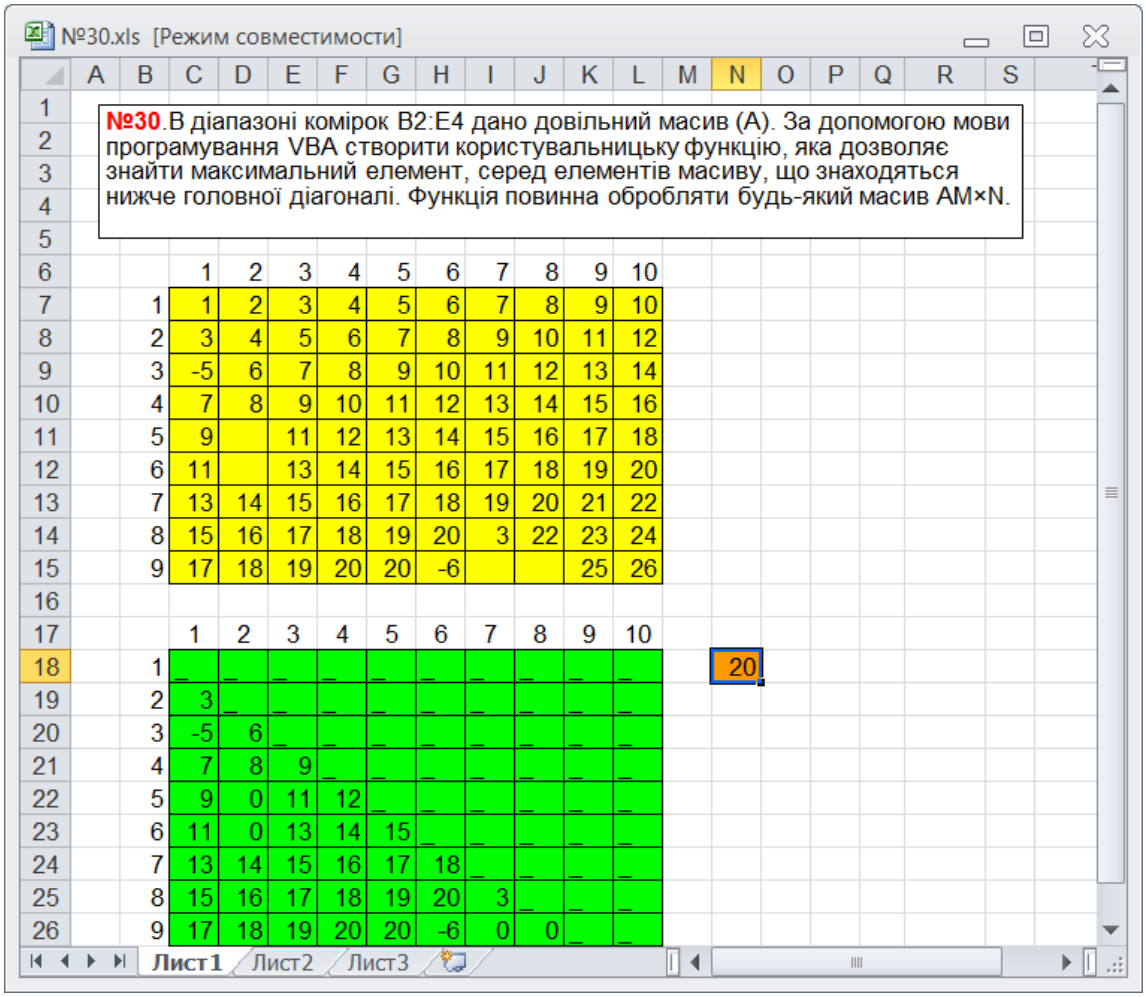


Рисунок 6.12 – Робочий аркуш

Програмний код

```
Public max
Function qqq(m, n, a)
ReDim res(1 To m, 1 To n)
```

```
max = 1E-300

For I = 1 To m
  For j = 1 To n
     If I > j Then
            res(I, j) = a(I, j)
                      If res(I, j) > max Then max = res(I, j)
               Else
            res(I, j) = "_"
     End If

     Next j, I
     qqq = res
End Function
```

Задача 3. «Довжина строки»

Умова. За допомогою мови програмування VBA створити користувальницьку функцію, аналогічну вбудованій функції Excel ДЛСТР(), не використовуючи функцію VBA Len(). Результат перевірити за допомогою вбудованої функції Excel.

Програмний код

```
Sub ДЛСТР1()
  a = Range("E7")
    i = 1
    While Mid(a, i, 1) <> ("")
    MsgBox CStr(Left(a, i))
        i = i + 1
      Wend
      MsgBox "Всего длина исходной строки=" & i
End Sub
```

Задача 4. «Добуток матриць»

Умова. В діапазоні комірок дані довільні матриці. За допомогою мови програмування VBA розробити алгоритм пошуку матричного добутку поданих матриць. Результат перевірити за допомогою вбудованої функції Excel.

Реалізація розрахунків

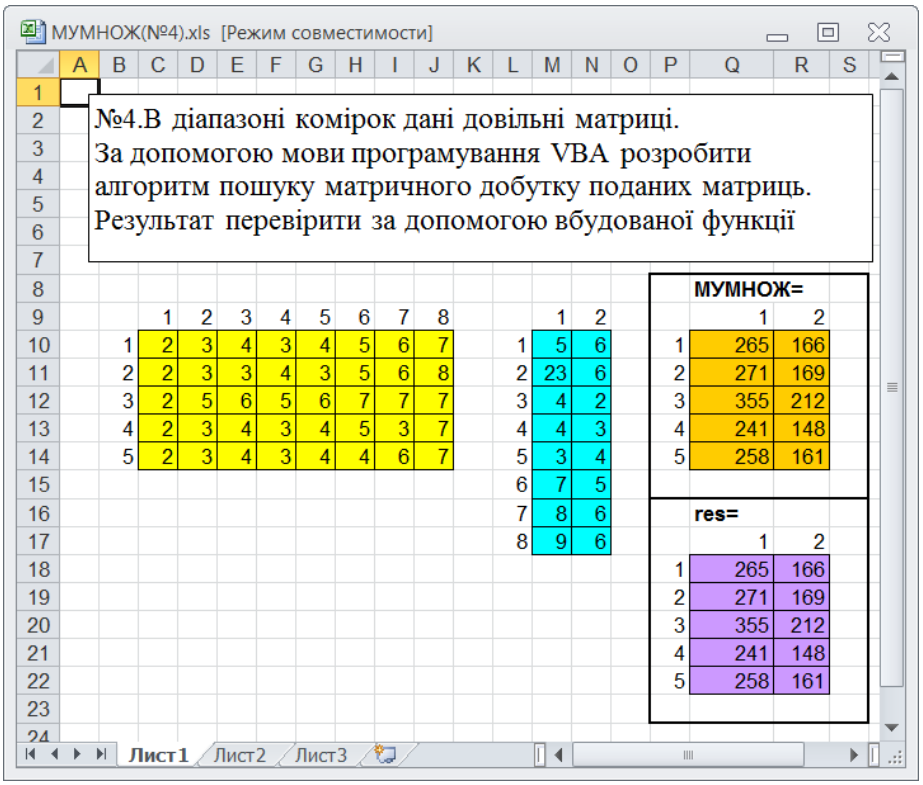


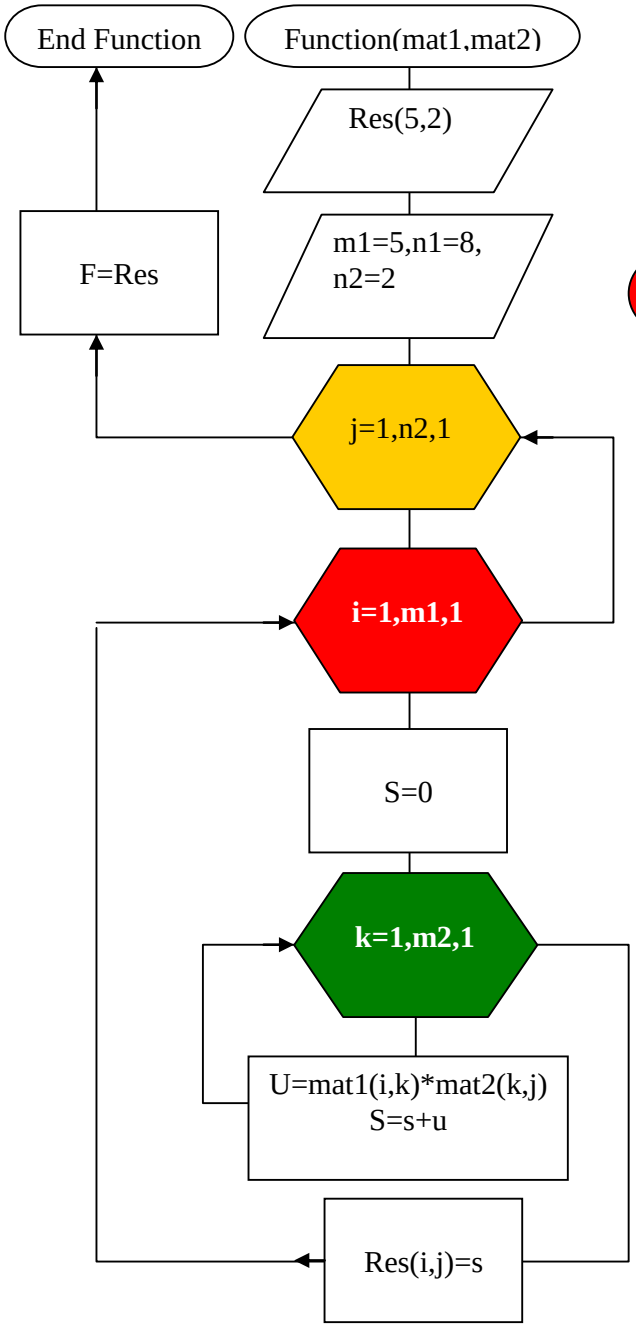
Рисунок 6.13 – Робочий аркуш

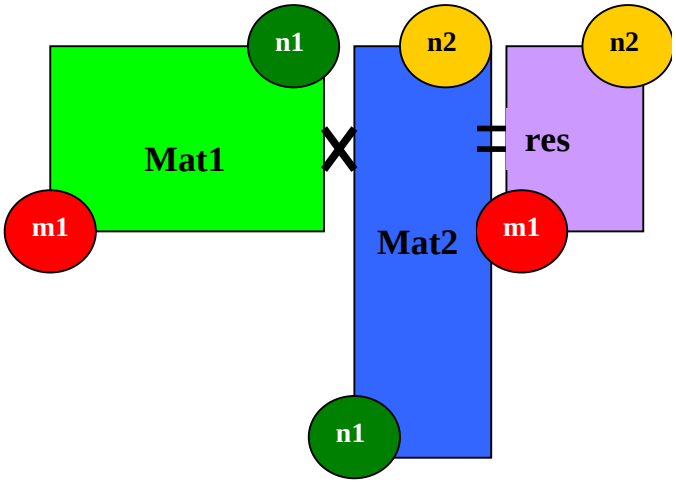
Програмний код

```
Public res(1 To 5, 1 To 2)
Function МУМНОЖ1(mat1, mat2)

m1 = 5
n1 = 8
m2 = n1
n2 = 2
For j = 1 To n2
      For i = 1 To m1
     s = 0
                 For k = 1 To n1
                     u = mat1(i, k) * mat2(k, j)
                     s = s + u
                 Next k
           res(i, j) = s
      Next i
  Next j
МУМНОЖ1 = res
End Function
```

Схема алгоритму





Задача 5. «Сума сум квадратів матриць»

Умова. В діапазоні комірок дані довільні матриці. За допомогою мови програмування VBA розробити алгоритм пошуку суми сум квадратів поданих матриць. Результат перевірити за допомогою вбудованої функції Excel.

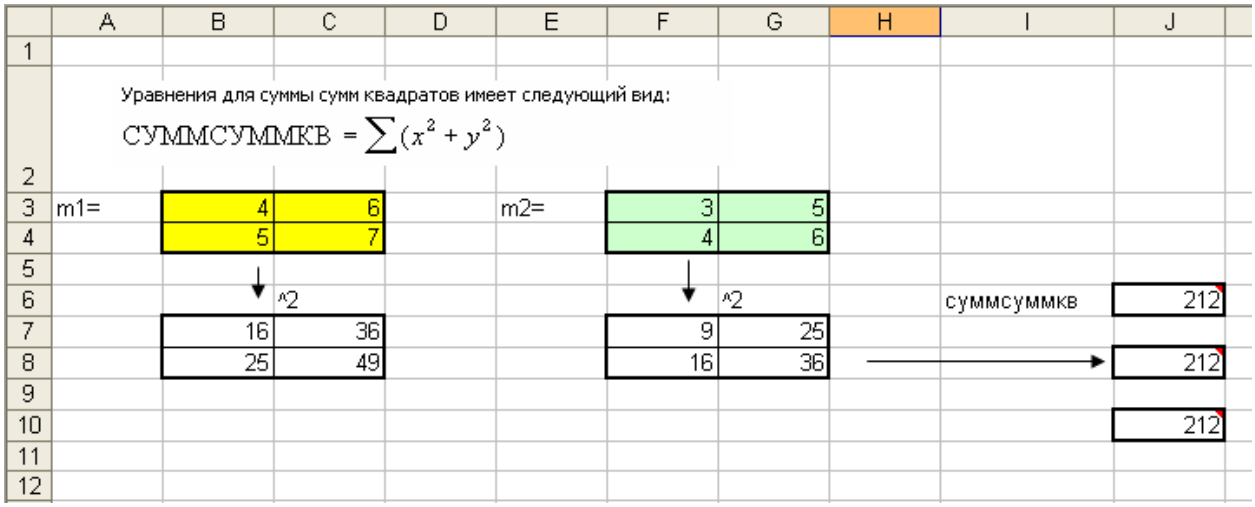


Рисунок 6.14 – Робочий аркуш

Програмний код

```
Public m1(1 To 10, 1 To 10)
Public m1кв(1 To 10, 1 To 10)
Public m2(1 To 10, 1 To 10)
Public m2кв(1 To 10, 1 To 10)

Function Сумма_сумм_квадратов(m1, m2)
  For i = 1 To 2
    For j = 1 To 2

m1кв(i, j) = m1(i, j) ^ 2
m2кв(i, j) = m2(i, j) ^ 2
s = s + m1кв(i, j) + m2кв(i, j)
Next j, i

 Сумма_сумм_квадратов = s

End Function
```

Індивідуальні завдання подано у додатку В.

ДОДАТОК А

А.1 Алгоритм. Види алгоритмів. Блок-схеми

Алгоритм – це послідовність дій над початковими даними та проміжними результатами необхідних для одержання кінцевого результату.

Алгоритм повинен мати наступні властивості:

– дискретність – процес рішення подається як послідовність кроків (етапів) і відбувається в часі дискретно;

– детермінованість (визначеність) – кожен крок повинен бути чітко визначеним і не повинен допускати довільного тлумачення;

– результативність – алгоритм повинен приводити до рішення задачі за скінчене число кроків;

– масовість – алгоритм рішення задачі розробляється в загальному виді, таким чином, щоб він міг бути застосованим для цілого класу задач, що різняться лише початковими даними.

Розрізняють:

– лінійний алгоритм;

– алгоритм з розгалуженням (за порядком розгалуження);

– циклічний алгоритм.

Алгоритм розв’язку будь-якої задачі, яка піддається формалізації, може бути поданий як комбінація цих трьох вище названих.

Алгоритм можна формалізувати словесно, у вигляді графічної схеми чи програмного коду на одній з мов програмування.

Побудова блок-схеми алгоритму дозволяє суттєво спростити подальший процес його формалізації у формі програмного коду. Основні графічні примітиви (блоки) з яких складається блок-схема та їх формати наведені на рис. А.1.

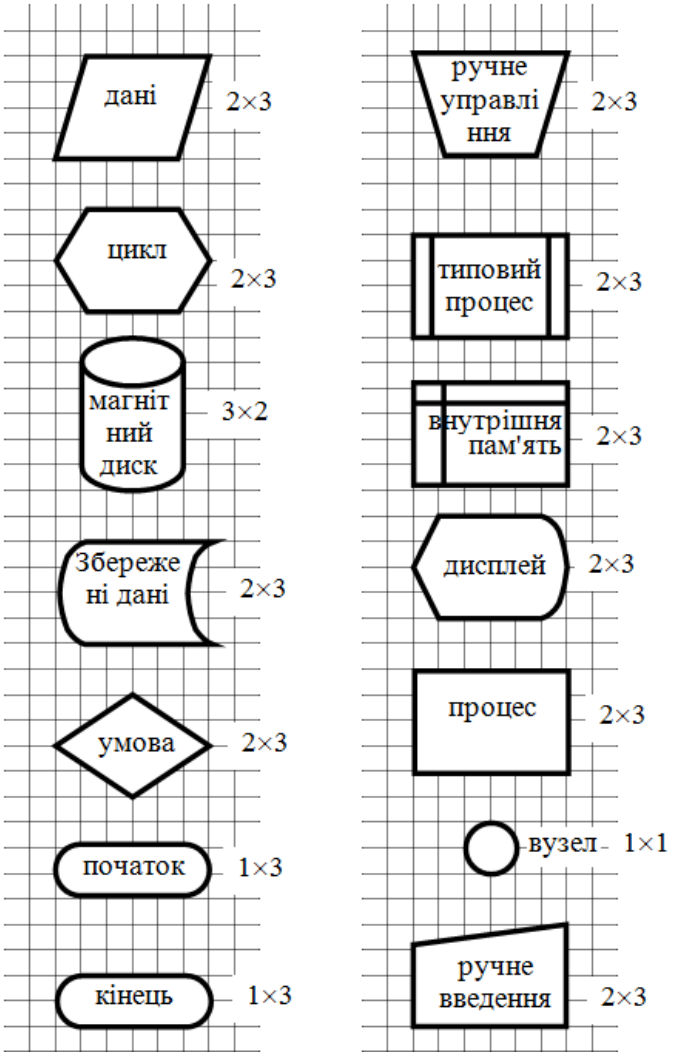


Рисунок А.1 – Формат основних блоків схем

Розглянемо більш детально кожний з типів алгоритмів.

1. Лінійний алгоритм характеризується однонаправленим послідовним переходом від блоку до блоку (від оператора до оператора) по мірі їх виконання.

2. Розгалужений алгоритм має місце в тому разі, коли в залежності від виконання чи невиконання умови виконується одна чи друга дія.

3. Множинний вибір є узагальненням розгалуження, коли в залежності від значення змінної виконується одна з багатьох дій.

4. Циклічний алгоритм передбачає багаторазове повторення програмних дій (обчислень, введення даних, виведення результатів, тощо) при різних початкових умовах.

ДОДАТОК Б

Б.1 Клас Windows Forms. Відомості про об’єктно-орієнтоване програмування

У Windows Forms термін «форма» – синонім вікна верхнього рівня. Головне вікно програми – форма. Будь-які інші вікна верхнього рівня, які має додаток – також форми. Вікна діалогу також вважаються формами. Незважаючи на назву, додатки, що використовують Windows Forms, не виглядають як форми. Подібно традиційним Windows-додаткам програми здійснюють повний контроль над подіями у власних вікнах.

Програми, що використовують Windows Forms використовують класи System.Windows.Forms. Цей розділ включає такі класи, як Form, який моделює поведінку вікон або форм; Menu, який представляє меню; Clipboard, який дає можливість додаткам Windows Forms використовувати буфер обміну. Він також містить численні класи, які надають засоби управління, наприклад: Button, TextBox, ListView, MonthCalendar і т. д. Ці класи можуть бути включені в додаток: з використанням тільки імені класу, або з використанням повного імені, наприклад: System.WinForms.Button.

В основі майже кожної програми, написаної зі застосуванням Windows Forms, – похідний клас від System.Windows.Forms. Зразок цього класу представляє головне вікно програми. System.Windows.Forms має безліч властивостей і методів, які мають багатий програмний інтерфейс до форм.

Додатки, засновані на Windows Forms, які використовують кнопки, списки та інші типи компонентів Windows, використовують класи управління System.Windows.Forms, що значно спрощує програмування інтерфейсу.

Основні поняття об’єктно-орієнтованого програмування

До появи персональних комп'ютерів технологія створення комп'ютерних програм базувалася на процедурному програмуванні, в якому основою були функції й процедури, тобто дії. Програміст визначав, які дії й обчислення потрібні для вирішення поставленої задачі, потім описував ці дії у вигляді процедур і функцій та об'єднував їх в програму. Створена в такий спосіб комп'ютерна програма відрізнялася чітким алгоритмом роботи – послідовністю дій для досягнення поставленої мети.

Об'єктно-орієнтоване програмування (ООП) зародилося з появою ПК в мовах програмування Pascal, Ада, Smalltalk, С. У ОПП головною відправною точкою при проектуванні програми є не процедура, не дія, а об'єкт. Такий підхід є досить зрозумілим, оскільки в реальному світі нас оточують саме об'єкти, які взаємодіють один з одним.

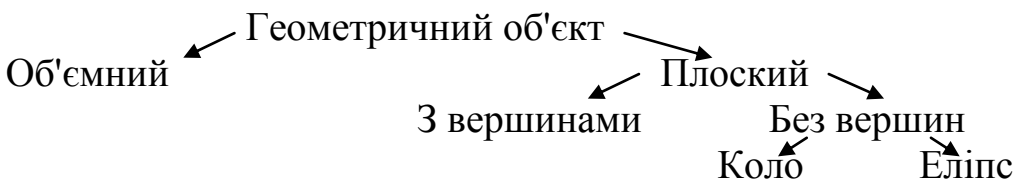
Об'єкти

Назвемо об'єктом поняття, абстракцію або будь-який предмет з чітко окресленими межами, що має сенс у контексті розглянутої прикладної проблеми. Всі об'єкти можна ідентифікувати, між об'єктами можна встановити відношення тотожності (два Сидорових Івана – це різні люди, але вони студенти однієї групи).

Класи об'єктів

Класом називають особливу структуру, яка може мати у своєму складі поля, методи та властивості. Клас виступає в якості об'єктного типу даних, а об'єкт – це конкретний екземпляр класу. Наприклад, два Сидорових Івана належать одному й тому ж класу об'єктів, вони – студенти групи. Саме з цим пов'язана їх однаковість (однаковий шифр групи, один розклад занять і т. д.).

Кожен конкретний клас має свої особливості поведінки і характеристики, що визначають цей клас.



Найвищий рівень – загальний і найпростіший, кожний наступний рівень більш специфічний і менш загальний. На самому останньому рівні можна визначити колір, стиль заповнення, величину радіусу кола і т. п.

Якщо характеристика вже одного разу визначена для більш високого рівня, то всі рівні, розташовані нижче мають ту ж характеристику (якщо вже визначена окружність, зрозуміло, що вершин у неї немає).

Таким чином, класи-спадкоємці можуть успадковувати характеристики класів-батьків.

Властивості

Властивості – перелік параметрів об'єкта, які визначають зовнішній вигляд і поведінку об'єкта, виділяють унікальні особливості кожного екземпляра. До властивостей відносяться: ім'я, тип, значення, колір, розмір та ін. Стан – сукупність всіх властивостей даного об'єкта.

Метод

Метод – це деяка дія (операція), яку можна виконувати над даним об'єктом. У результаті цієї дії змінюється об'єкт (наприклад, місце розташування, колір та ін.) Іншими словами, методом називається команда, яку може виконувати об'єкт. Для кожного класу об'єктів є свій перелік методів, які можна до нього застосувати або які він може виконати. Наприклад, об'єкт можна видалити з екрану, перемістити в інше місце.

Події

Кожен об'єкт здатний реагувати на певні події – це різновид властивостей об’єкта. При виникненні події проводиться її обробка.

Події – сигнали, що формуються зовнішнім середовищем, на які об'єкт повинен відреагувати відповідним чином.

Події настають в результаті дій користувача – переміщення курсору, натискання кнопок миші або клавіш на клавіатурі, а також у результаті роботи самих об'єктів. Для кожного об'єкта визначено безліч подій, на які він може реагувати. Для конкретних екземплярів об'єкта можуть бути визначені обробники якихось із цих подій, які й визначають реакцію даного екземпляра об'єкта.

Об'єкт можна визначити як сукупність властивостей і методів, а також подій, на які він може реагувати.

Зовнішнє управління об'єктом здійснюється через обробник подій. Ці обробники звертаються до методів та властивостей об'єкта.

Б.2 Загальні властивості для основних елементів управління

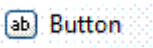
Таблиця Б.1 – Елементи управління VB

Тип об’єкта	Призначення	Префікс
Label	Напис	lbl
TextBox	Текстове поле	txt
CommandButton	Кнопка	cmd
CheckBox	Прапорець	chk
OptionButton	Перемикач	opt
Frame	Рамка	fra
ListBox	Список	lst
ComboBox	Поле зі списком	cbo
PictureBox	Графічний фрейм	pic
Form	Форма	frm

Таблиця Б.2 – Загальні властивості елементів управління VB

Властивості	Описання
BackColor	Задає фоновий колір об’єкта. Колір можна задати, вибравши його з палітри кольорів або вказавши прив'язку до поточної колірної схеми операційної системи
Backstyle	Управляє відображенням фону області виведення тексту. Область виводу тексту може бути зафарбована (Opaque) або бути прозорою (Transparent).
Font	Шрифт, який використовується для відображення тексту на об’єкті
Text	Напис, що розміщується на об'єкті
Name	Вказує ім’я, яке використовується для ідентифікації об’єкта в програмі
Left	Відстань від лівої межі об'єкта до лівої границі форми
Top	Відстань від верхньої межі об'єкта до верхньої межі форми
Height	Висота компонента
Width	Ширина компонента
Visible	Дозволяє приховати компонент (значення властивості – False) або зробити його видимим (значення властивості – True)
Enabled	Дозволяє активувати об’єкт

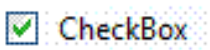
1. Button

Об'єкт управління **Button**(кнопка): 

.

За допомогою клікання мишею на кнопці можна задати виконання якої-небудь дії.

2. CheckBox

Об'єкт управління **CheckBox :** 

.

Для введення в форму даних, які можуть мати тільки одне з двох допустимих значень, призначені елементи управління називаються прапорцями.

Прапорці, наприклад, дозволяють користувачу дати відповідь на поставлене питання. У разі позитивної відповіді користувач встановлює прапорець, і він набуває вигляд квадрата, в якому розміщена галочка:  .

Таблиця Б.3 – Властивості елемента управління CheckBox

Властивості	Описання
Name Caption	Текст, який знаходиться праворуч від прапорця
Checked	Стан перемикача: прапорець встановлено (у елементі є «галочка»)
Unchecked	Прапорець скинутий (нема «галочки»)

3. Panel

Об'єкт управління Panel(панель): 

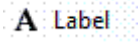
.

Елемент управління Panel являє собою панель і використовується в формах в основному для об'єднання елементів в групи, наприклад, для розміщення у формі двох і більше груп перемикачів. Елемент, що розташовується на елементі управління Panel, автоматично стає його елементом.

Таблиця Б.4 – Властивості елемента управління Panel

Властивості	Описання
AutoScroll	Керує появленням смуг прокрутки

4. Label

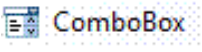
Об'єкт управління Label: 

.

Компонент Label призначений для виведення тексту на поверхню форми. Властивості компонента визначають вид і розташування тексту на поверхні форми.

Таблиця Б.5 – Властивості елемента управління Label

Властивості	Описання
Caption	Відображуваний текст
AutoSize	Ознака того, що розмір поля визначається його вмістом
Wordwrap	Ознака того, що слова, які не вміщуються у поточному рядку, автоматично переносяться на наступний рядок

5. Об'єкт управління ComboBox: 

.

Списки типу ComboBox називають полями, що розкриваються або полями зі списками. Для вибору значення із списку спочатку необхідно список відкрити, натиснувши кнопку зі стрілкою, розташовану з правого боку поля вводу:  .

ComboBox поєднує функції списку ListBox і поля введення TextBox. Використання списків ComboBox дозволяє представляти більший обсяг інформації, заощаджуючи при цьому місце у формі.

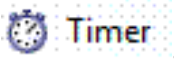
Події.

Для поля зі списком важливу роль відіграють події як поля введення, так і списку. Основні з них – Click, яка використовується для вибору елемента списку, та Change – зміна запису в полі введення тексту.

Таблиця Б.6 – Властивості елемента управління ComboBox

Властивості	Описання
Caption	Відображуваний текст
List	Елементи списку – масив рядків
ListCount	Кількість елементів списку
lndex	Номер елемента, вибраного у списку. Якщо жоден з елементів списку не був обраний, то значення властивості дорівнює –1. Нумерація елементів починається з нуля
Style	Стиль (вид) списку. Якщо значення властивості рівне DropdownCombo (0), то дані в полі редагування можна ввести з клавіатури або вибрати зі списку (щоб отримати доступ до списку, його треба розкрити). Якщо значення властивості рівне Simple-Combo (1), то дані можна ввести в полі редагування з клавіатури або вибрати зі списку, причому список доступний завжди. Якщо значення властивості рівне DropdownList (2), то дані в полі редагування можна ввести лише шляхом вибору зі списку
Text	Вміст поля редагування (дані, введенні користувачем з клавіатури або виділені зі списку)
Sorted	Ознака необхідності автоматичного сортування списку після додавання чергового елемента (значення властивості – True)
Locked	Блокування списку. Визначає заборону вибору елемента зі списку (рядок введення також блокується)

6. Timer

Об'єкт управління Timer: 

.

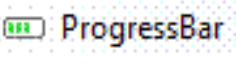
Використання таймера є гарним способом управління програмою. За допомогою таймера можна запускати або завершувати процеси додатки в певні моменти часу. Під час проектування таймер розміщується на формі, під час виконання програми його на формі не видно.

Таймер має єдину подію – Timer, яка викликається після закінчення встановленого тимчасового інтервалу.

Таблиця Б.7 – Властивості елемента управління Timer

Властивості	Описання
Name	Ім'я компоненту. Використовується для доступу до компонента та його властивостей
Interval	Період генерації події Timer. Задається в мілісекундах
Enabled	Дозвіл генерації (значення властивості – True) або блокування (значення властивості – False) події Timer

7. ProgressBar

Об'єкт управління ProgressBar: 

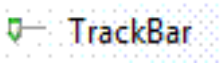
.

Відображення процесу виконання операцій. У Windows цей елемент відображає перебіг процесу копіювання, переміщення, завантаження й т. п.

Таблиця Б.8 – Властивості елемента управління ProgressBar

Властивості	Описання
Min	Нижня межа
Max	Верхня межа
Value	Поточне значенн
Orientation	Орієнтація елементів: – Вертикальна (ccOrientationVertical) – Горизонтальна (ccOrientationHorizontal)
Scrolling	Спосіб відображення процесу: – Безперервний (ccScrollingSmooth) – Сегментарний (ccScrollingStandard)

8. TrackBar

Об'єкт управління TrackBar: 

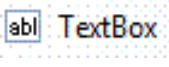
.

За допомогою TrackBar можна задавати значення змінних з діапазону значень.

Таблиця Б.9 – Властивості елемента управління TrackBar

Властивості	Описання
Min	Мінімальне значення для бігунка, коли він знаходиться на лівому краю
Max	Максимальне значення для бігунка, коли він знаходиться на правому краю
Value	Поточне положення бігунка
SmallChange	Мала зміна. Зрушення бігунка на одну позицію. Крок бігунка
LargeChange	Велика зміна. Зрушення бігунка при подвійному натисканні на ньому

9. TextBox

Об'єкт управління TextBox: 

.

Компонент TextBox являє собою поле введення-редагування тексту.

Події.

При використанні текстового поля викликає інтерес декілька подій.

Насамперед, подія Change, яка викликається при зміні змісту текстового поля, ця подія відбувається кожен раз при введенні, знищенні або зміні символу. Наприклад, при введенні в текстове поле слова «так», подія Change відбувається три рази – по одному разу для кожної літери.

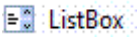
Для аналізу введеного в поле тексту найкраще всього підходить подія LostFocus. Ця подія викликається після того, як текстове поле зробиться неактивним (після передачі фокуса іншому елементу, коли користувач закінчить введення). Однак, якщо це поле є єдиним елементом управління в формі, то воно не може втратити фокус.

Для того, щоб видалити або ініціалізувати вміст текстового вікна, використовується подія GotFocus. Вона викликається тоді, коли користувач «входить» в текстове вікно.

Таблиця Б.10 – Властивості елемента управління TextBox

Властивості	Описання
Text	Текст, що знаходиться в полі редагування
Locked	Використовується для обмеження можливості змінити текст в полі редагування. Якщо значення властивості рівне False, то текст в полі редагування змінити не можна
MultiLine	Робить можливим багаторядкове відображення тексту
ScrollBars	Управляє відображенням смуг прокрутки: Vertical – тільки смуга вертикальної прокрутки; Horizontal – тільки смуга горизонтального прокрутки; Both – вертикальна і горизонтальна смуги прокрутки; None – без смуг прокрутки

10. ListBox

Об'єкт управління ListBox : 

Компонент ListBox являє собою список, в якому можна вибрати потрібний елемент.

Події.

Основна подія списку – Click. Ця подія викликається, якщо користувач за допомогою мишки або клавіш управління вибере елемент зі списку.

Методи.

Вікно списку – це перший з елементів управління, для якого важливу роль відіграють методи. Методи списку необхідні для обробки елементів списку – додавання або видалення. Для додавання нових елементів використовується метод AddItem: ListBox.AddItem Елемент [Індекс].

Для видалення елемента зі списку використовується метод RemoveItem, якому, як параметр передається індекс елементу, який видаляється. Індексація елементів починається з нуля (0):

```
ListBox.RemoveItem – Індекс елемента
```

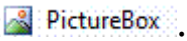
Для видалення усіх елементів використовується метод Clear:

```
ListBox.Clear
```

Таблиця Б.11 – Властивості елемента управління ListBox

Властивості	Описання
ForeColor	Колір, що використовується для відображення елементів списку
List	Елементи списку – масив рядків
ListCount	Кількість елементів списку
Sorted	Ознака необхідності автоматичного сортування списку після додавання чергового елемента
Listindex	Номер вибраного елементу списку (нумерація елементів списку починається з нуля)
MultiSelect	Дозволяє зробити можливим множинний вибір зі списку. Якщо значення цієї властивості – None (0), то вибрати кілька елементів зі списку не можна. При значенні, рівному Simple (1), кожен елемент, на який клікнули, стає обраним. Скасування вибору виробляється за допомогою повторного клікання миші або за допомогою клавіші-пробіл. Якщо значення властивості рівне Extended (2), то множинний вибір здійснюється за допомогою миші й клавіші <Shift> або <Ctrl> (клацання кнопкою миші на елементі списку, утримуючи клавішу <Shift> позначає елемент як обраний, повторне клацання скасовує вибір)
Style	Стиль (вид) списку. Якщо значення властивості дорівнює CheckBox (1), то перед кожним елементом списку відображається компонент Checkbox. При цьому для вибору елементу із списку потрібно встановити відповідний прапорець. При значенні властивості, рівному Standard (0), список має стандартний вигляд

11. PictureBox

Об'єкт управління PictureBox: 

.

Компонент PictureBox забезпечує відображення графіки. Зображення можна завантажити з файлу або сформувати з графічних примітивів (намалювати) під час роботи програми.

Таблиця Б.12 – Властивості елемента управління PictureBox

Властивості	Описання
Picture	Зображення (об'єкт BitMap), яке відображається у полі компонента. Завантажити картинку можна під час розробки форми або з файлу під час роботи програми (функція LoadPicture)
AutoSize	Властивість дозволяє (True) або забороняє (False) автоматичну зміну розміру компоненти (області виведення ілюстрації) відповідно до розміру картинки, завантаженої в компонент
BorderStyle	Стиль межі компонента. Якщо значення властивості – FixedSingle (1), то межа стандартна (тонка лінія), якщо – None (0), то межа не відображається
ForeColor	Шрифт, яким метод Print виводить текст. Для методу Print задає колір символів, для методів креслення графічних примітивів (об'єктів) – колір ліній
FillColor	Встановлює колір зафарбовування внутрішніх областей графічних примітивів (об'єктів), викреслює в полі (на поверхні) компонента

Закінчення таблиці Б.12

DrawStyle	Вид контуру графічних об'єктів, викреслює в полі компонента відповідними методами: Solid (0) – суцільна лінія; Dash (1) – пунктирна лінія; Dot (2) – лінія з точок; Dash – Dot (3) – лінія виду "точка -тире "; Dash – Dot – Dot (4) - лінія виду «тире – крапка – крапка»; Transparent (5) – «прозора» лінія
DrawWidth	Товщина ліній для графічних об'єкті
Sorted	Ознака необхідності автоматичного сортування списку після додавання чергового елемента
Listindex	Номер вибраного елементу списку (нумерація елементів списку починається з нуля)
MultiSelect	Дозволяє зробити можливим множинний вибір зі списку. Якщо значення цієї властивості – None (0), то вибрати кілька елементів зі списку не можна. При значенні, рівному Simple (1), кожен елемент, на який клікнули, стає обраним. Скасування вибору виробляється за допомогою повторного клікання миші або за допомогою клавіші пробіл. Якщо значення властивості рівне Extended (2), то множинний вибір здійснюється за допомогою миші й клавіші <Shift> або <Ctrl> (клацання кнопкою миші на елементі списку, утримуючи клавішу <Shift> позначає елемент як обраний, повторне клацання скасовує вибір)
Style	Стиль (вид) списку. Якщо значення властивості рівне CheckBox (1), то перед кожним елементом списку відображається компонент Checkbox. При цьому для вибору елементу із списку потрібно встановити відповідний прапорець. При значенні властивості, рівній Standard (0), список має стандартний вигляд

12. DataGridView

Об'єкт управління DataGridView:  DataGridView .

DataGridView надає потужний і гнучкий спосіб відображення даних в табличному форматі. DataGridView можна використовувати для подання в режимі тільки читання невеликих об'єктів даних; можна розширити цей елемент для представлення великих обсягів даних в режимі редагування.

Таблиця Б.13 – Властивості елемента управління PictureBox

Член, властивість, метод	Описання
DataGridView.Item (ИндСтолбца, ИндСтроки) значение = Str(dataGridView1(Столб, Стр).Value) ' получить значение ячейки для ячейки в столбце Столб, строке Стр значение = Str(dataGridView1("ИмяСт ", Стр).Value) ' получить значение ячейки для ячейки в столбце "ИмяСт", строке Стр	Надає індексатор для отримання або задання клітинки, розташованої на перетині стовпця (ИндСтолбца) і рядки (ИндСтроки) з заданими індексами
DataGridView1.Rows.Add()	Додає новий рядок в колекцію
DataGridView1.RowCount = число_строк	Повертає або задає кількість рядків, що відображаються в об'єкті DataGridView
DataGridView1.ColumnCount = число_столбцов	Повертає або задає число стовпців, які відображаються в об'єкті DataGridView
DataGridView1.ColumnHeadersVisible = True (False)	Повертає або задає значення, яке вказує, чи відображається рядок заголовків стовпців
DataGridView1.RowHeadersVisible = True (False)	Повертає або задає значення, яке вказує, чи відображається стовпець, що містить заголовки рядків

ДОДАТОК В

Індивідуальні завдання задачі «Автоматизація обчислень в MS Excel»

1. В діапазоні комірок B2:D4 дана довільна матриця. За допомогою мови програмування VBA розробити алгоритм обчислення визначника поданої матриці. Осередок, що міститиме відповідь, програмно «залити» жовтим кольором. Результат перевірити за допомогою вбудованої функції Excel.
2. В діапазоні комірок B2:D6 дана довільна матриця. За допомогою мови програмування VBA розробити алгоритм обчислення її транспонованої матриці. Осередок, що міститиме відповідь, програмно «залити» сірим кольором та розграфити. Результат перевірити за допомогою вбудованої функції Excel.
3. В діапазоні комірок B2:D6, B10:D14 дані довільні матриці. За допомогою мови програмування VBA розробити алгоритм пошуку поелементного добутку поданих матриць.
4. В діапазоні комірок B2:D6, B10:E12 дані довільні матриці. За допомогою мови програмування VBA розробити алгоритм пошуку матричного добутку поданих матриць. Результат перевірити за допомогою вбудованої функції Excel.
5. За допомогою мови програмування VBA створити користувальницьку функцію, аналогічну вбудованій функції Excel ОКРУГЛ(), не використовуючи стандартні функції VBA. Результат перевірити за допомогою вбудованої функції Excel.
6. За допомогою мови програмування VBA створити користувальницьку функцію, аналогічну вбудованій функції Excel ДЛСТР(), не використовуючи функцію VBA Len(). Результат перевірити за допомогою вбудованої функції Excel.
7. За допомогою мови програмування VBA створити користувальницьку функцію, аналогічну вбудованій функції Excel СЖПРОБЕЛЫ(), не використовуючи стандартну функцію VBA. Результат перевірити за допомогою вбудованої функції Excel.
8. За допомогою мови програмування VBA створити користувальницьку функцію, аналогічну вбудованій функції Excel СОВПАД(). Перевірку організувати посимвольно. Результат перевірити за допомогою вбудованої функції Excel.
9. За допомогою мови програмування VBA створити користувальницьку функцію, аналогічну вбудованій функції Excel ФАКТР(). Результат перевірити за допомогою вбудованої функції Excel.
10. За допомогою мови програмування VBA створити користувальницьку функцію (для 2 довільних чисел), аналогічну вбудованій функції Excel

НОД(). Результат перевірити за допомогою вбудованої функції Excel.

11. За допомогою мови програмування VBA створити користувальницьку функцію (для 2 довільних чисел), аналогічну вбудованій функції Excel НОК(). Результат перевірити за допомогою вбудованої функції Excel.
12. За допомогою мови програмування VBA створити користувальницьку функцію, яка дозволяє перевірити, чи є простим задане число.
13. В діапазоні комірок B2:D6 дано довільний масив. За допомогою мови програмування VBA створити користувальницьку функцію, аналогічну вбудованій функції Excel СУММКВ(), для пошуку суми квадратів масиву. Результат перевірити за допомогою функцій Excel.
14. В діапазоні комірок B2:D6, B10:D14 дані довільні масиви. За допомогою мови програмування VBA створити користувальницьку функцію, аналогічну вбудованій функції Excel СУММРАЗНКВ(), для пошуку суми різниць квадратів поданих масивів. Результат перевірити за допомогою функцій Excel.
15. В діапазоні комірок B2:D6, B10:D14 дані довільні масиви. За допомогою мови програмування VBA створити користувальницьку функцію, аналогічну вбудованій функції Excel СУММКВРАЗН(), для пошуку суми квадратів різниць поданих масивів. Результат перевірити за допомогою функцій Excel.
16. В діапазоні комірок B2:D6, B10:D14 дані довільні масиви. За допомогою мови програмування VBA створити користувальницьку функцію, аналогічну вбудованій функції Excel СУММСУММКВ(), для пошуку суми сум квадратів поданих масивів. Результат перевірити за допомогою функцій Excel.
17. В діапазоні комірок B2:D6, B10:D14 дані довільні масиви. За допомогою мови програмування VBA створити користувальницьку функцію, аналогічну вбудованій функції Excel СУММПРОИЗВ(), для пошуку суми добутків поданих масивів. Результат перевірити за допомогою функцій Excel.
18. В діапазоні комірок B2:D6 дано довільний масив. За допомогою мови програмування VBA створити користувальницьку функцію, аналогічну вбудованій функції Excel МАКС(), для пошуку максимального елементу масиву. Результат перевірити за допомогою функцій Excel.
19. В діапазоні комірок B2:D6 дано довільний масив. За допомогою мови програмування VBA створити користувальницьку функцію, аналогічну вбудованій функції Excel МИН(), для пошуку мінімального елементу масиву. Результат перевірити за допомогою функцій Excel.
20. В діапазоні комірок B2:D6 дано довільний масив (A). За допомогою мови програмування VBA створити користувальницьку функцію, яка дозволяє отримати новий масив (C) за формулою:

$C_{i1} = A_{i1},\ \ C_{i2} = A_{i2},\ C_{i3} = (A_{i1} + A_{i2}) \times A_{i3}$

21. В діапазоні комірок B2:D4 дано довільний масив (A). За допомогою мови програмування VBA створити користувальницьку функцію, яка дозволяє отримати новий масив (C) за формулою:
$C_{1i} = A_{1i},\ \ C_{2i} = A_{2i},\ C_{3i} = (A_{1i} + A_{2i}) \times A_{3i}$
22. В діапазоні комірок B2:D4 дано довільний масив (A). За допомогою мови програмування VBA створити користувальницьку функцію, яка дозволяє отримати новий масив (C) за формулою:
$C_{ij} = \frac{A_{ij}}{|A_{ij}|}$, де $|A_{ij}|$ – визначник матриці A. (для пошуку визначника не використовувати стандартну функцію VBA).
23. В діапазоні комірок B2:D4 дано довільний масив (A). За допомогою мови програмування VBA створити користувальницьку функцію, яка дозволяє отримати новий масив (C) за формулою:
$C_{ij} = \frac{A_{ij}^T}{i + j}$, де A_{ij}^T – транспонована матриця A. (для пошуку транспонованої матриці, не використовувати стандартну функцію VBA).
24. В діапазоні комірок B2:D4 дано довільну матрицю (A). За допомогою мови програмування VBA створити користувальницьку функцію, що дозволяє «піднести» матрицю до заданої степені:
$C = A^n = \underbrace{[A \times A \times ...]}_{n}$, де $A \times A$ – матричний добуток матриць.
25. За допомогою мови програмування VBA створити користувальницьку функцію, що дозволяє представити число в процентному форматі, з додаванням символу «%».
26. В діапазоні комірок B2:D4 дано довільний масив (A). За допомогою мови програмування VBA створити користувальницьку функцію, яка дозволяє поміняти місцями елементи головної (1,1; 2,2; 3,3) та побічної діагоналі (1,3; 2,2; 3,1). Отриманий масив записати в діапазон B10:D12. Функція повинна обробляти будь-який масив $A_{N \times N}$.
27. В діапазоні комірок B2:E4 дано довільний масив (A). За допомогою мови програмування VBA створити користувальницьку функцію, яка дозволяє замінити елементи периметру масиву A їх синусами (SIN(A)). Отриманий таким чином масив C записати в діапазон B10:D12. Функція повинна обробляти будь-який масив $A_{M \times N}$.
28. В діапазоні комірок B2:D4 дана довільна матриця (A). За допомогою мови програмування VBA розробити алгоритм визначення знаку (+ чи -) визначника поданої матриці.
29. В діапазоні комірок B2:D4 дана довільна матриця (A). За допомогою

мови програмування VBA розробити алгоритм перевірки визначника поданої матриці на нерівність 0.

30. В діапазоні комірок B2:E4 дано довільний масив (A). За допомогою мови програмування VBA створити користувальницьку функцію, яка дозволяє знайти мінімальний та максимальний елементи, серед елементів масиву, що знаходяться нижче головної діагоналі. Функція повинна обробляти будь-який масив $A_{M\times N}$.

ПЕРЕЛІК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Абрамян М. Э. Практикум по программированию на языках C# и VB.NET. 2-е изд. / М. Э. Абрамян. – Ростов н/Д.: «ЦВВР», 2007. – 514 с.

2. Балена Ф. Современная практика программирования на Microsoft Visual Basic и Visual C# / Ф. Балена, Дж. Димауро. – М.: Русская редакция, 2006. – 970 с.

3. Дубовцев А. В. Microsoft .NET в подлиннике / А. В. Дубовцев. – СПб.: БХВ-Петербург, 2004. – 850 с.

4. Нортроп Т. Основы разработки приложений на платформе Microsoft .NET Framework / Т. Нортроп, Ш. Уилдермьюс, Б. Райан. – М.: Русская редакция, 2007. – 730 с.

5. Рихтер Дж. Программирование на платформе Microsoft .NET Framework. Мастер-класс. 3-е изд. / Дж. Рихтер. – М.: Русская редакция, 2005. – 565 с.

6. Microsoft Corporation. Разработка Windows-приложений на Microsoft Visual Basic .NET и Microsoft Visual C# .NET. – М.: Русская редакция, 2003. – 645 с.

ЕЛЕКТРОННЕ НАВЧАЛЬНО-МЕТОДИЧНЕ ВИДАННЯ

Корольов Марк Євгенович
Кравченко Роман Сергійович

МЕТОДИЧНІ ВКАЗІВКИ ДО ПРАКТИКУМУ З ДИСЦИПЛІНИ «ІНФОРМАТИКА», «ІНФОРМАТИКА ТА СИСТЕМОЛОГІЯ». ЧАСТИНА 1

Підписано до випуску __.__.20__ р. Гарнітура Times New.
Умов. друк. арк. 5,13 Зам. № __

Державний вищий навчальний заклад
«Донецький національний технічний університет»
Автомобільно-дорожній інститут
84646, м. Горлівка, вул. Кірова, 51
E-mail: druknf@rambler.ru

Редакційно-видавничий відділ

Свідоцтво про внесення до Державного реєстру видавців, виготовників і розповсюджувачів видавничої продукції ДК № 2982 від 21.09.2007 р.