МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ
по виконанню лабораторних робіт
з дисципліни
«Невизначене програмування»
для студентів 3 курсу
Освітньо-кваліфікаційний рівень – «Бакалавр»
Напрям підготовки – «Комп'ютерні науки»
Спеціальність – інформаційні управляючі системи та технології

Затверджено на засіданні
методичної комісії ф-ту
Комп'ютерних наук
Протокол №__ від _____
Декан факультету
_____Коваленко Л.Б.

Затверджено на засіданні
кафедри Інформатики
Протокол №___від_____
Зав. кафедри
_____Мещеряков В.І.

Одеса, 2014

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

по виконанню лабораторних робіт

з дисципліни

«Невизначене програмування»

для студентів 3 курсу

Освітньо-кваліфікаційний рівень – «Бакалавр»

Напрям підготовки – «Комп’ютерні науки»

Спеціальність – інформаційні управляючі системи та технології

Одеса, 2014

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

по виконанню лабораторних робіт

з дисципліни

«Невизначене програмування»

для студентів 3 курсу

Освітньо-кваліфікаційний рівень – «Бакалавр»

Напрям підготовки – «Комп’ютерні науки»

Спеціальність – інформаційні управляючі системи та технології

Затверджено на засіданні

методичної комісії ф-ту

Комп’ютерних наук

Протокол №__ від _____

Одеса, 2014

Методичні вказівки до виконання лабораторних робіт з дисципліни «Невизначене програмування» для студентів для студентів 3 курсу, освітньо-кваліфікаційний рівень – «Бакалавр», напрям підготовки – «Комп'ютерні науки». спеціальність – інформаційні управляючі системи та технології

Укладачі:

Мещеряков В.І., д.т.н., професор, зав. кафедри інформатики

Черепанова К.В., асистент кафедри інформатики.

ЗМІСТ

Передмова

Мета методичних вказівок – поглиблене вивчення та закріплення лекційного матеріалу щодо вивчення дисципліни «Невизначене програмування».

Метою дисципліни є ознайомлення студентів з основними задачами математичного програмування, заснованих на принципах нечіткої логіки, обробки даних на базі нейронних мереж та генетичних алгоритмів.

Метою викладання дисципліни також є формування у студентів знань про новітні інформаційні системи та методи програмування, їх використання для вирішення практичних завдань, набуття навичок роботи з програмним забезпеченням, що використовується у сучасній практиці.

У результаті вивчення дисципліни «Невизначене програмування» студент повинен:

ЗНАТИ:

- принципи нечіткого програмування
- принципи побудови нейронних мереж
- принципи генетичних алгоритмів

ВМІТИ:

- розробляти автоматичні пристрої на нечіткій логіці
- розробляти структури нейронних мереж
- проводити навчання простих нейронних мереж

По кожній лабораторній роботі студент повинен скласти **звіт**, який містить в собі:

1. Назву роботи. Мету.
2. Умову завдання згідно варіанта.
3. Хід виконання роботи.
4. Відповіді на контрольні питання.

Оформлений звіт захищається студентом усно.

Варіант індивідуального завдання надається викладачем.

Перед виконанням лабораторних робіт у комп'ютерному класі студенти зобов'язані пройти інструктаж з техніки безпеки та охорони праці.

Правила техніки безпеки та охорона праці

Згідно з «Правилами техніки безпеки в лабораторіях інформатики» студентам забороняється:

- з`являтися та знаходитись приміщенні в нетверезому стані;
- ставити поруч з клавіатурою ємності з рідиною;
- перебувати в приміщенні в верхній одежі та завалювати нею робочі столи та стільці;
- працювати в лабораторії більше 6-ти годин на день (для вагітних жінок – більше 4-х годин);
- за власною ініціативою змінювати закріплені за ними робочі місця та знаходитись в приміщенні під час роботи іншої учбової групи;
- самостійно виконувати вмикання електроживлення лабораторії та заміну складових частин ПК, що вийшли із ладу.

У випадку виявлення несправностей обчислювальної техніки студент повинен сповістити про це викладача чи будь-кого з навчально-допоміжного персоналу лабораторії.

Теоретичні відомості

Нечітка логіка (англ. Fuzzy logic) — розділ математики, що є узагальненням класичної логіки і теорії множин, що базує на понятті нечіткої множини, як об'єкту з функцією приналежності елементу до множини, що набуває будь-яких значень в інтервалі, а не лише 0 або 1. На основі цього поняття вводяться різні логічні операції над нечіткою множиною і формулюється поняття лінгвістичної змінної, як значення якої виступає нечітка множина.

Предметом нечіткої логіки вважається дослідження міркувань в умовах нечіткості, розмитості, схожих з міркуваннями в звичайному сенсі, і їх вживання в обчислювальних системах.

Напевно, самою вражаючою здатністю людського інтелекту є здатність приймати вірні рішення в умовах неповної і нечіткої інформації. Побудова моделей наближених роздумів людини і використання їх в комп'ютерних системах представляє сьогодні одну з найважливіших проблем науки.

Штучна нейронна мережа (ШНМ) — математична модель, а також її програмна або апаратна реалізація, побудована за принципом організації і функціонування біологічних нейронних мереж — мереж нервових клітин живого організму. Це поняття виникло при вивченні процесів, що протікають в мозку, і при спробі змоделювати ці процеси. Першою такою спробою були нейронні мережі У. Маккалока і В. Піттса. Після розробки алгоритмів вчення, отримувані моделі стали використовувати в практичних цілях: у завданнях прогнозування, для розпізнавання образів, в завданнях управління і ін.

ШНМ є системою сполучених і взаємодіючих між собою простих процесорів (штучних нейронів). Такі процесори зазвичай досить прості (особливо порівняно з процесорами, використовуваними в персональних комп'ютерах). Кожен процесор подібної мережі має справу лише з сигналами, які він періодично отримує, і сигналами, які він періодично посилає іншим процесорам. І, проте,

будучи сполученими в чималу мережу з керованою взаємодією, такі локально прості процесори разом здатні виконувати досить складні завдання.

З точки зору машинного вчення, нейронна мережа є окремим випадком методів розпізнавання образів, аналізу дискримінанта, методів кластеризації і тому подібне З математичної точки зору, вчення нейронних мереж — це багатопараметричне завдання нелінійної оптимізації. З точки зору кібернетики, нейронна мережа використовується в завданнях адаптивного управління і як алгоритми для робототехникі. З точки зору розвитку обчислювальної техніки і програмування, нейронна мережа — спосіб вирішення проблеми ефективного паралелізму. А з точки зору штучного інтелекту, ШНМ є основою філософського перебігу коннективізма і основним напрямом в структурному підході по вивченню можливості побудови (моделювання) природного інтелекту за допомогою комп'ютерних алгоритмів.

Нейронні мережі не програмуються в звичному сенсі цього слова, вони вивчаються. Можливість вчення — одна з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно вчення полягає в знаходженні коефіцієнтів зв'язків між нейронами. В процесі вчення нейронна мережа здатна виявляти складні залежності між вхідними даними і вихідними, а також виконувати узагальнення. Це означає, що в разі успішного вчення мережа зможе повернути вірний результат на підставі даних, які були відсутні в повчальній вибірці, а також неповних і «зашумлених», частково спотворених даних.

Лабораторна робота № 1

Мета: Знайомство з технологією .NET Framework та мовою C#. Скласти та відлагодити програму обчислення значення арифметичного виразу.

Теоретичні відомості:

.NET Framework - програмна платформа, випущена компанією Microsoft у 2002 році. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), яка підходить для різних мов програмування. Функціональні можливості CLR доступні в будь-яких мовах програмування, що використовують цю середу. В склад платформи .NET Framework входять мови програмування C# і Visual Basic і велика бібліотека класів.

Однією з основних ідей Microsoft .NET є сумісність програмних частин, написаних на різних мовах. Наприклад, служба, написана на C++ для Microsoft .NET, може звернутися до методу класу з бібліотеки, написаної на Delphi; на C# можна написати клас, одержаний від класу, написаного на Visual Basic .NET, а виключення, створене методом, написаних на C#, може бути перехоплено і опрацьовано в Delphi. Кожна бібліотека (збірка) .NET має відомості про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок.

Класи

Для представлення об'єктів у мовах C#, Java, C++, Delphi і т. п. використовується поняття клас, аналогічне буденного змісту цього слова в контексті клас членистоногих, клас задач і т. п. Клас є узагальненим поняттям, що визначає характеристики і поведінку деякої множини конкретних об'єктів цього класу, званих екземплярами класу.

Класичний клас містить дані, які визначають властивості об'єктів класу, і функції, які визначають їх поведінку. Останнім часом в клас часто додається третя складова - події, на які може реагувати об'єкт класу.

Всі класи бібліотеки .NET, а також всі класи, які програміст створює в середовищі .NET мають одного загального предка - клас **object** і організовані в єдину ієрархічну структуру. Всередині неї класи логічно згруповані в так звані простору імен, які служать для впорядкування імен класів і запобігання конфліктів імен: в різних просторах імена можуть збігатися. Простору імен можуть бути вкладеними, їх ідея аналогічна знайомої вам ієрархічній структурі каталогів на комп'ютері.

Будь-яка програма, створювана в .NET використовує простір імен **System**. У ньому визначені класи, які забезпечують базову функціональність, наприклад, підтримують виконання математичних операцій, управління пам'яттю і введення-виведення.

Зазвичай в один простір імен об'єднують взаємопов'язані класи. Наприклад, простір System. Net містить класи, що відносяться до передачі даних по мережі, System.Windows.Forms - елементи графічного інтерфейсу користувача, такі як форми, кнопки і т. д. Ім'я кожного простору імен являє собою неподільну сутність, однозначно його визначальну.

C# - об'єктно-орієнтована мова, тому що написана на ньому програма являє собою сукупність взаємодіючих між собою класів.

При запуску VisualStudio з'являється початкова сторінка зі списком останніх проектів, а також командами Створити проект і Відкрити проект. Натисніть на посилання Створити проект або виберіть у меню Файл команду Створити проект, на екрані з'явиться діалог для створення нового проекту (рис. 1.1).

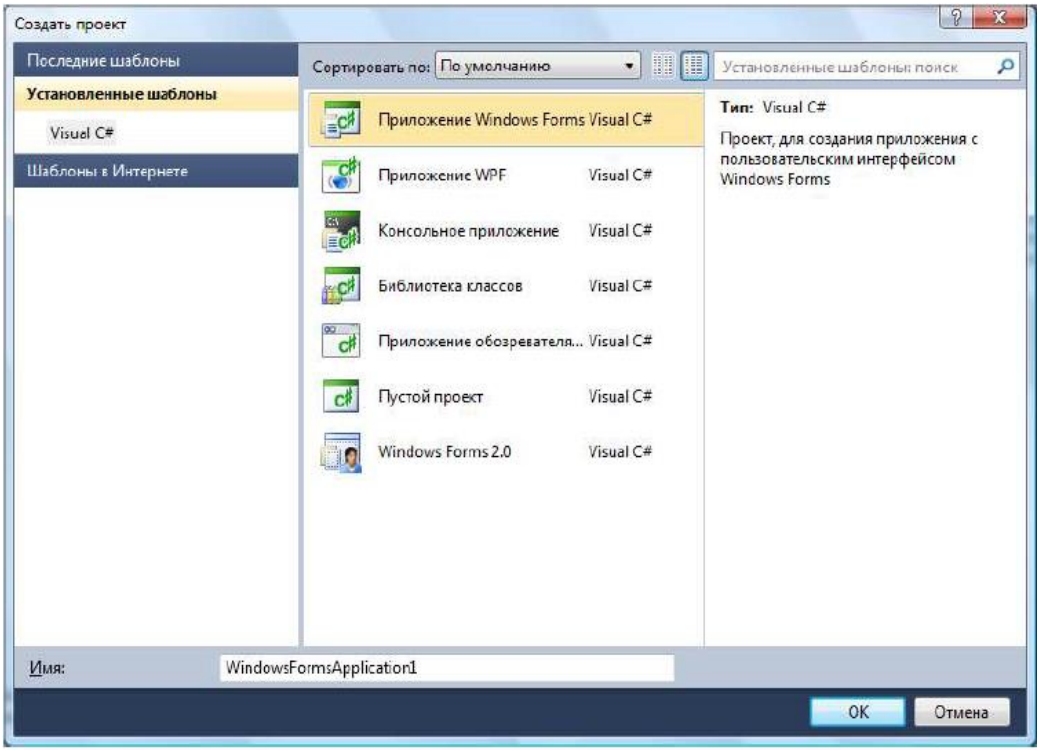

Рисунок 1.1 Створення нового проекту

Зліва в списку шаблонів наведено мови програмування, які підтримує ця версія Visual Studio: переконайтеся, що там виділено розділ Visual C#. У середній частині наведені типи проектів, які можна створити. У наших лабораторних роботах будуть використовуватися два типи проектів:

• Додаток Windows Forms - даний тип проекту дозволяє створити повноцінну програму з вікнами і елементами управління (кнопки, поля введення тощо) Такий вид додатка найбільш звичний для більшості користувачів.

• Консольний додаток - в цьому типі проекту вікно являє собою текстову консоль, в яку додаток може виводити тексти або чекати введення інформації користувача. Консольні програми часто використовуються для обчислювальних задач, для яких не потрібен складний або гарний користувальницький інтерфейс.

У головному вікні Visual Studio є кілька основних елементів, які будуть допомагати нам в роботі. Насамперед, це форма, рис. 1.2 - майбутнє вікно нашої програми, на якому будуть розміщуватися елементи управління.

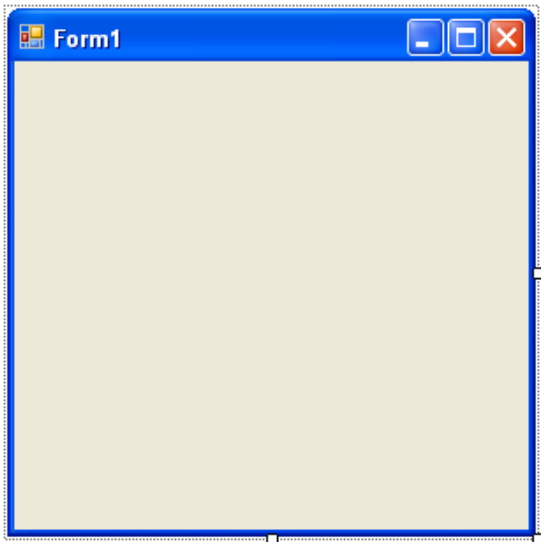

Рисунок 1.2 Form1

При виконанні програми поміщені елементи управління будуть мати той же вигляд, що і на етапі проектування.

Другий за важливістю об'єкт - це вікно властивостей, в якому наведені всі основні властивості виділеного елемента керування або вікна.

Самі елементи управління можна брати на панелі елементів, рис. 1.3

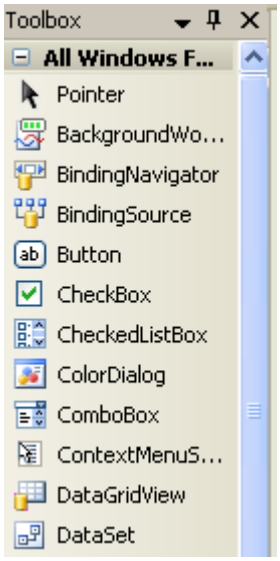

Рисунок 1.3 Панель елементів

Всі елементи управління розбиті на логічні групи, що полегшує пошук потрібних елементів. Якщо панелі немає на екрані, її потрібно активувати командою Вид -► Панель елементів.

Нарешті, оглядач рішень містить список всіх файлів, що входять в проект, включаючи додані зображення і службові файли. Активується командою Вид -► Оглядач рішенні.

Зазначені панелі можуть вже бути присутнім на екрані, але бути прихованими за іншими панелями або згорнуті до бічній стороні вікна. В цьому випадку достатньо клацнути на відповідному ярлику, щоб вивести панель на передній план.

Програма на мові C# складається як опис алгоритмів, які необхідно виконати, якщо виникає певна подія, пов'язана з формою (наприклад, клацання «мишею» на кнопці - подія Click, завантаження форми - Load). Для кожного оброблюваного у формі події, за допомогою вікна властивостей, у тексті програми організовується метод, в якому програміст записує на мові C# необхідний алгоритм.

Кожен елемент має свій набір обробників подій. однак деякі з них присуши більшості елементів керування. Найбільш часто вживані події:

Activated- Форма отримує це подія при активації.

Load- Виникає при завантаженні форми. В обробнику данної події слід задавати дії, які повинні відбуватися в момент створення форми, наприклад, встановлення початкових значень.

Keypress- Виникає при натисканні кнопки на клавіатурі Зазвичай це подія використовується в тому випадку, коли необхідна реакція на натиснення однієї клавіші.

Keydown - Виникає при натисканні клавіші на клавіатурі. Обробник цієї події отримує інформацію про натиснуту клавішу і стан клавіш Shift,Alt і Ctrl, а також про натиснуту кнопку миші.

Click - Виникає при натисканні кнопки миші в області елемента управління.

DoubleClick - Виникає при подвійному натисканні кнопки миші в області елемента управління.

Якщо якийсь обробник був доданий помилково або більше не потрібен, то для його видалення не можна просто видалити програмний код обробника! Спочатку потрібно видалити рядок з ім'ям обробника у вікні властивостей на закладці. В іншому випадку програма може перестати компілюватися і навіть відображати форму в дизайнері VisualStudio.

Опис даних

Типи даних мають особливе значення в C#, оскільки це суворо типізований мову. Це означає, що всі операції піддаються суворому контролю з боку компілятора на відповідність типів, причому неприпустимі операції не компілюються. Така сувора перевірка типів дозволяє запобігти помилок і підвищити надійність програм. Для забезпечення контролю типів всі змінні, вирази і значення повинні належати до певного типу.

В C# визначені дев'ять цілочисельних типів: **char, byte, sbyte, short, ushort, int, uint, long та ulong**. Тип **char** може зберігати числа, але частіше використовується для представлення символів. Інші вісім призначенні для числових розрахунків. Існує три типи даних з плаваючою точкою: **float, decimal та double**. Основним типом при роботі з рядками є тип **string**, який задає рядок змінної довжини.

Введення/виведення даних в програму

Розглянемо один із способів введення даних через елементи, які розташовані на формі. Для введення даних найчастіше використовують елемент управління TextBox, через звернення до його властивості Text. Властивість Text зберігає в собі рядок введених символів. Тому дані можна вважати таким чином:

```
Privatevoid buttonl_Click(object sender,EventArgse)
{
     strings = textBox1.Text;
}
```

Однак з рядком символів важко виробляти арифметичні операції, тому краще всього при введенні числових даних перевести рядок в ціле або дійсне число. Для цього у типів, або int і double існують методи Parse для перетворення рядків у числа. З цими числами можна виконувати різні арифметичні дії. Таким чином, попередній приклад можна переробити наступним чином:

```
Private void button1_Click(object sender, EventArgs e)
{
     string s = textBox1.Text;
     int a = int.Parse(s);
     int b = a * a;
}
```

Перед виведенням числові дані слід перетворити назад у рядок. Для цього у кожної змінної існує метод ToString(), який повертає в результаті рядок з символьним поданням значення. Висновок даних можна здійснювати в елементи TextBox або Label, використовую властивість Text. Наприклад:

```
Private void buttonl_Click(object sender, EventArgse)
{
     strings = textBoxl.Text; inta = int.Parse(s);
     intb = a * a;
     labell.Text = b.ToString();
}
```

Практична частина.

Необхідно скласти програму обчислення для заданих значень x, y, z арифметичного виразу.

$$u = tan^2\,(x + y) - e^{y-z}\sqrt{\cos x^2 + sinz^2}$$

Форма діалогу з користувачем повинна виглядати подібним чином, рис. 1.4., на ній необхідно розташувати поля для введення даних і текстове вікно для виведення результатів.

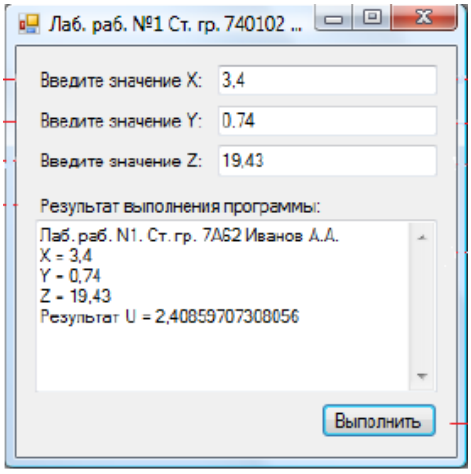

Рисунок 1.4

Робота з програмою буде така - при натисканні на кнопку "Виконати" у вікні Textbox4 з'являється результат. Змінивши вхідні значення в Textbox 1-3 можна буде отримати новий результат.

Наберіть наведений нижче текст програми, при цьому необхідно заздалегідь створити обробники подій для форми та кнопки, інакше програма працювати не буде.

```
usingSystem;
usingSystem.Windows.Forms;
namespaceMyFirstApp
```

```
{
    publicpartialclassForml: Form{
        publicForml() {
            InitializeComponent();
        }
        privatevoidForml_Load(object sender,
            EventArgse)
        {  // Початкове значення X
            textBox1.Text = "3,4";
            // Початкове значення Y
            textBox2.Text = "0,74";
            // Початкове значення Z
            textBox3.Text = "19,43";   }
        privatevoidbuttonl_Click(object sender,
            EventArgse)
        { // Зчитування значення X
            double x = double.Parse(textBoxl.Text);
            // Вивід значення X у вікно
            textBox4.Text += Environment.NewLine+
            "X = " + x.ToString();
        // Зчитування значення Y
        doubley = double.Parse(textBox2.Text)
        // Вивід значення Y у вікно
        textBox4.Text += Environment.NewLine+
        "Y = " + y.ToString();
         // Зчитування значення Z
         double z = double.Parse(textBox3.Text)
         // Вивід значення Z в окно
         textBox4.Text += Environment.NewLine+
         "Z = " + z.ToString();
         // Обчислюємо арифметичний вираз
         double a = Math.Tan(x + y) *
```

```
            Math.Tan(x + y);
        double b = Math.Exp(y - z);
        doublec = Math.Sqrt(Math.Cos(x*x) +
        Math.Sin(z*z));
        double u = a -b * c;
// Виводимо результат у вікно
textBox4.Text += Environment.NewLine+ "Результат U = " + u.ToString();
```

Контрольні запитання:

1. Що розуміється під терміном «.NET Framework»?

2. Чи залежать програми, які розроблюють в .NET від платформи?

3. Наведіть узагальнений синтаксис оголошення змінної мовою C#.

4. Наведіть узагальнений синтаксис ініціалізації змінної на мовою C#.

5. Перелічіть п'ять простих типів мови C#.

6. Що розуміється під областю видимості змінної в мові C#?

7. Як співвідноситься час життя змінної і область видимості?

8. Наведіть синтаксис умовного оператора в загальному вигляді. Проілюструйте його фрагментом програми на мові C#.

9. Наведіть синтаксис оператора вибору у загальному вигляді. Проілюструйте його фрагментом програми на мові C#.

Індивідуальне завдання:

Необхідно розробити програму для обчислення рішення виразу. Завдання виконується згідно з варіантом. Вхідні дані та результат для перевірки наведені в кожному варіанті.

Встановіть необхідну кількість вікон для введення даних, тексти заголовків на формі, типи змінних і функції перетворення при введенні і виведенні результатів.

1. $$t=\frac{2\cos\left(x-\frac{\pi}{6}\right)}{0.5+\sin^2 y}\left(1+\frac{z^2}{3-z^2/5}\right).$$

При x=14.26, y=-1.22, $z=3.5\times10^{-2}$ t=0.564849.

2. $$u=\frac{\sqrt[3]{8+|x-y|^2+1}}{x^2+y^2+2}-e^{|x-y|}\left(tg^2 z+1\right)^x.$$

При x=-4.5, $y=0.75\times10^{-4}$, $z=0.845\times10^{2}$ u=-55.6848.

3. $$v=\frac{1+\sin^2(x+y)}{\left|x-\frac{2y}{1+x^2y^2}\right|}x^{|y|}+\cos^2\left(arctg\frac{1}{z}\right).$$

При x=3.74×10^{-2}, y=-0.825, $z=0.16\times10^{2}$, v=1.0553.

4. $$w=\left|\cos x-\cos y\right|^{\left(1+2\sin^2 y\right)}\left(1+z+\frac{z^2}{2}+\frac{z^3}{3}+\frac{z^4}{4}\right).$$

При x=0.4×10^{4}, y=-0.875, z=-0.475×10^{-3} w=1.9873.

5. $$\alpha=\ln\left(y^{-\sqrt{|x|}}\right)\left(x-\frac{y}{2}\right)+\sin^2 arctg(z).$$

При x=-15.246, y=4.642×10^{-2}, $z=20.001\times10^{2}$ α=-182.036.

6. $$\beta=\sqrt{10\left(\sqrt[3]{x}+x^{y+2}\right)}\left(\arcsin^2 z-|x-y|\right).$$

При x=16.55×10^{-3}, y=-2.75, z=0.15 β=-38.902.

7. $$\gamma=5arctg(x)-\frac{1}{4}\arccos(x)\frac{x+3|x-y|+x^2}{|x-y|z+x^2}.$$

При x=0.1722, y=6.33, $z=3.25\times10^{-4}$ γ=-172.025.

8. $$\varphi=\frac{e^{|x-y|}|x-y|^{x+y}}{arctg(x)+arctg(z)}+\sqrt[3]{x^6+\ln^2 y}.$$

При x=-2.235×10^{-2}, y=2.23, z=15.221 φ=39.374.

9. $$\psi = \left| x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}} \right| + (y-x)\frac{\cos y - \frac{z}{(y-x)}}{1+(y-x)^2}.$$

При x=1.825×10^{2}, y=18.225, z=-3.298×10^{-2} ψ=1.2131.

10. $$a = 2^{-x}\sqrt{x+\sqrt[4]{|y|}}\sqrt[3]{e^{x-1/\sin z}}.$$

При x=3.981×10^{-2}, y=-1.625×10^{3}, z=0.512 a=1.26185.

11. $$b = y^{\sqrt[3]{|x|}} + \cos^3(y)\frac{|x-y|\left(1+\frac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{|x-y|}+\frac{x}{2}}.$$

При x=6.251, y=0.827, z=25.001 b=0.7121.

12. $$c = 2^{(y^x)} + (3^x)^y - \frac{y\left(arctgz - \frac{\pi}{6}\right)}{|x| + \frac{1}{y^2+1}}.$$

При x=3.251, y=0.325, z=0.466×10^{-4} c=4.025.

13. $$f = \frac{\sqrt[4]{y+\sqrt[3]{x-1}}}{|x-y|\left(\sin^2 z + tgz\right)}.$$

При x=17.421, y=10.365×10^{-3}, z=0.828×10^{5} f=0.33056.

14. $$g = \frac{y^{x+1}}{\sqrt[3]{|y-2|+3}} + \frac{x+\frac{y}{2}}{2|x+y|}(x+1)^{-1/\sin z}.$$

При x=12.3×10^{-1}, y=15.4, z=0.252×10^{3} g=82.8257.

15. $$h = \frac{x^{y+1}+e^{y-1}}{1+x|y-tgz|}\left(1+|y-x|\right) + \frac{|y-x|^2}{2} - \frac{|y-x|^3}{3}.$$

При x=2.444, y=0.869×10^{-2}, z=-0.13×10^{3} h=-0.49871.

Лабораторна робота № 2

Мета: навчитися користуватися елементами управління для організації перемикань. Скласти та відлагодити програму обчислення значення функції.

Теоретичні відомості.

Логічні змінні та операції над ними

Змінні логічного типу описуються за допомогою службового слова bool. Вони можуть приймати тільки два значення - False (неправда) і True (істина). Описуються логічні змінні наступним чином:

bool b;

В мові C# є логічні операції, що застосовуються до змінних логічного типу. Це операції логічного заперечення (!), логічне І (&&) і логічне АБО (| |). Операція логічного заперечення є унарной операцією. Результат операції ! є False, якщо операнд правдивий, і True, якщо операнд має значення false. Так.

!TrueFalse (неправда є брехня)

! FalseTrue (не брехня є правда)

Результат операції логічне І (&&) є true, тільки якщо обидва операнда істинні, і брехня у всіх інших випадках. Результат операції логічне АБО (| |) є істина, якщо який-небудь з її операндів правдивий, і помилкова тільки тоді, коли обидва операнди помилкові.

Умовні оператори

Оператори розгалуження дозволяють змінити порядок виконання операторів у програмі. До операторів розгалуження відносяться умовний оператор if і оператор вибору switch.

Умовний оператор if використовується для розгалуження процесу обробки даних на два напрямки. Він може мати одну з форм: скорочену чи повну.

Форма скороченого оператора if:

If b (s);

де b - логічне або арифметичне вираження, істинність якого перевіряється;

s - оператор.

Форма повного оператора if:

if (b) S1;

else S2;

де в - логічне або арифметичне вираження, істинність якого перевіряється;

s1, s2 - оператори.

Оператор вибору switch призначений для розгалуження процесу обчислень по декількох напрямках. Формат оператора:

```
Switch(<вираз>)
{
case<константний_вираз_1>:
[<оператор 1>];
<оператор перехода>
case< константний _ вираз _2>:
[<оператор 2>];
<оператор перехода>;
…
      case< константний _вираз_п>:
      [<оператор п>];
      <оператор переходу>;
      [default:
      <оператор>;]
}
```

Кнопки-перемикачі

При створенні програм VisualStudio для організації розгалужень часто використовуються елементи керування у вигляді кнопок-перемикачів (RadioButton). Стан такої кнопки (увімкнено-вимкнено) візуально

відображається на формі, а в програмі можна дізнатися його з допомогою властивості Checked: якщо кнопка включена, це властивість буде містити True, інакше-False. Якщо користувач вибирає один з варіантів перемикача в групі, всі інші автоматично відключаються.

Групуються радіокнопки за допомогою будь-якого контейнера-часто це буває елемент GroupBox. Радіокнопки, розміщені в різних контейнерах, утворюють незалежні групи.

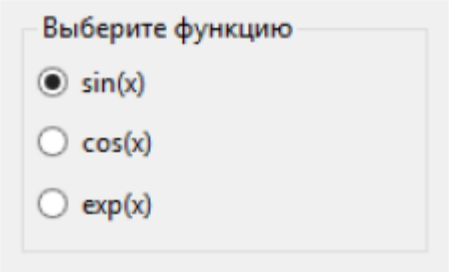

Рисунок 2.1

```
if(radioButton1.Checked)
MessageBox.Show("Вибрана функція: синус");
     elseif(radioButton2.Checked)
MessageBox.show("Вибрана функція: косинус");
     elseif(radioButton1.Checked)
MessageBox.show("Вибрана функція: експонента")
```

Практична частина.

Необхідно ввести три числа - x, y, z та обчислити значення виразу

$$U = \begin{cases} y * \sin(x) + z, & \text{при} z - x = 0 \\ y * e^{\sin(x)} - z, & \text{при } z - x < 0 \\ y * \sin(\sin(x)) + z, & \text{при } z - x > 0 \end{cases}$$

Створити форму, див. рис. 2.2

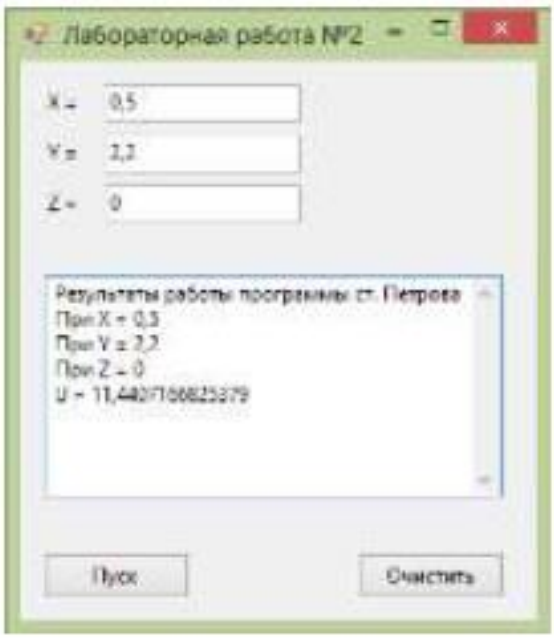

Рисунок 2.2

Розмістіть на формі елементи Label, TextBox і Button. Поле для виводу результатів також є елементом TextBox з встановленим в True властивістю Multiline і властивість ScrollBars встановленим у Both.

Створення обробників подій

Обробники подій створюються аналогічно тому, як і в попередніх лабораторних роботах. Текст обробника події натискання на кнопку «Пуск» наведено нижче.

```
privatevoidbuttonl_Click(object sender, EventArgse)
{
// Одержання вихідних даних з TextBox
doublex = Convert.ToDouble(textBox2.Text);
doubley = Convert.ToDouble(textBoxl.Text);
doublez = Convert.ToDouble(textBox3.Text);
// Введення вихідних даних у вікно результатів
textBox4.Text = "Результати роботи програми " +
"ст. Петрова И.И. " +
```

```
      Environment.NewLine;
      textBox4.Text += "При X = " + textBox2.Text +
                Environment.NewLine;
textBox4.Text += "При Y = " + textBoxl.Text+
                Environment.NewLine;
textBox4.Text += "При Z = " + textBox3.Text +
                Environment.NewLine;
// Обчислення виразу
doubleu;
if((z - x) == 0)
u = y * Math.Sin(x) * Math.Sin(x) + z;
else
if((z - x) < 0)
u = y * Math.Exp(Math.Sin(x)) - z;
else
                u = y * Math.Sin(Math.Sin(x)) + z;
// Вивід результату
textBox4.Text += "U = " + u.ToString() +
Environment.NewLine;
      }
```

Індивідуальне завдання:

За вказівкою викладача виберіть індивідуальне завдання з нижчеподаного списку. В якості f(x) використовувати за вибором: sin(x), x^2, e^x. За допомогою радіокнопок (RadioButton) дати користувачеві можливість під час роботи програми вибрати одну з трьох наведених вище функцій.

1. $$a=\begin{cases}(f(x)+y)^2-\sqrt{f(x)y}, & xy\rangle 0\\ (f(x)+y)^2+\sqrt{|\mathrm{f(x)\ y}|}, & xy\langle 0\\ (f(x)+y)^2+1, & xy=0.\end{cases}$$

2. $$b=\begin{cases}\ln(f(x))+(f(x)^2+y)^3, & x/y\rangle 0\\ \ln|f(x)/y|+(f(x)+y)^3, & x/y\langle 0\\ (f(x)^2+y)^3, & x=0\\ 0, & y=0.\end{cases}$$

3. $$c=\begin{cases}f(x)^2+y^2+\sin(y), & x-y=0\\ (f(x)-y)^2+\cos(y), & x-y\rangle 0\\ (y-f(x))^2+tg(y), & x-y\langle 0.\end{cases}$$

4. $$d=\begin{cases}(f(x)-y)^3+arctg(f(x)), & x\rangle y\\ (y-f(x))^3+arctg(f(x)), & y\rangle x\\ (y+f(x))^3+0.5, & y=x.\end{cases}$$

5. $$e=\begin{cases}i\sqrt{f(x)}, & i-\text{нечетное}, \ x\rangle 0\\ i/2\sqrt{|f(x)|}, & i-\text{четное}, \ x\langle 0\\ \sqrt{|if(x)|}, & \text{иначе.}\end{cases}$$

6. $$g=\begin{cases}e^{f(x)-|b|}, & 0.5\langle xb\langle 10\\ \sqrt{|f(x)+b|}, & 0.1\langle xb\langle 0.5\\ 2f(x)^2, & \text{иначе.}\end{cases}$$

7. $$s=\begin{cases}e^{f(x)}, & 1\langle xb\langle 10\\ \sqrt{|f(x)+4*b|}, & 12\langle xb\langle 40\\ bf(x)^2, & \text{иначе.}\end{cases}$$

8. $$j=\begin{cases}\sin(5f(x)+3m|f(x)|), & -1\langle m\langle x\\ \cos(3f(x)+5m|f(x)|), & x\rangle m\\ (f(x)+m)^2, & x=m.\end{cases}$$

9. $$l=\begin{cases}2f(x)^3+3p^2, & x\rangle|p|\\ |f(x)-p|, & 3\langle x\langle|p|\\ (f(x)-p)^2, & x=|p|.\end{cases}$$

10. $$k=\begin{cases}\ln(|f(x)|+|q|), & |xq|\rangle 10\\ e^{f(x)+q}, & |xq|\langle 10\\ f(x)+q, & |xq|=10\end{cases}$$

Контрольні запитання:

1. Для чого використовують радіо кнопки?
2. Опишіть властивості умовних операторів
3. Умови використання логічних типів даних
4. Як створити обробник подій?

Лабораторна робота № 3

Мета: вивчити найпростіші засоби налагодження програм у середовищі VisualStudio. Скласти та відлагодити програму циклічного алгоритму.

Теоретичні відомості.

Оператори організації циклів

Під циклом розуміється багаторазове виконання одних і тих же операторів при різних значеннях проміжних даних. Число повторень може бути задано у явній або неявній формі.

До операторів циклу відносяться: цикл з передумовою **while**, цикл з після умовою **do while**, цикл з параметром **for** і цикл перебору **foreach**. Розглянемо деякі з них.

Цикл з передумовою

Оператор циклу while організовує виконання одного оператора (простого або складного) заздалегідь невідоме число раз. Формат циклу while:

while (b) S;

де b - висловлення, істинність якого перевіряється умова завершення циклу);
s - тіло циклу - оператор (простий чи складений).

Перед кожним виконанням тіла циклу аналізується значення виразу: якщо істина - то виконується тіло циклу, і управління передається на повторну перевірку умови: якщо значення хибно - цикл завершується і управління передається на оператор, наступний за оператором s.

Якщо результат виразу виявиться помилковим при першій перевірці, то тіло циклу не виконується жодного разу. Відзначимо, що якщо умова під час роботи циклу не буде змінюватися, то можлива ситуація зациклювання, тобто неможливість виходу з циклу. Тому всередині тіла повинні перебувати оператори, що призводять до зміни значення виразу так. щоб цикл міг коректно завершитися.

В якості ілюстрації виконання циклу while розглянемо програму виведення цілих чисел від 1 до п по натисненню кнопки на формі:

```
     private void buttonl_Click(object sender, EventArgse)
     {
      intn = 10; // Кількість повторень циклу
      inti = 1; // Початкове значення
           while(i <= n) // Поки і менше або дорівнює n{
            MessageBox.Show(i.ToString());// Показуєм і
i++; // Збільшуємо і на 1
      }
}
```

Цикл з постумовою

Оператор циклу **dowhile** також організовує виконання одного оператора (простого або складного) заздалегідь невідоме число раз. Однак на відміну від циклу **while** умова завершення циклу перевіряється після виконання тіла циклу. Формат циклу **dowhile**:

do *s* while (*b*);

де **b** - висловлення, істинність якого перевіряється умова завершення циклу);

s - тіло циклу - оператор (простий або блок).

Спочатку виконується оператор **s**, а потім аналізується значення виразу: якщо воно істинне, то управління передається оператору S, якщо хибно - цикл завершується, і управління передається на оператор, наступний за умовою ст. Так як умова перевіряється після виконання тіла циклу, то в будь-якому випадку тіло циклу виконується хоча б один раз.

В операторі **dowhile**, так само як і в операторі **while**, можлива ситуація зациклення у разі, якщо умова завжди буде залишатися істиною.

Цикл з параметром

Цикл з параметром має наступну структуру:

for *(<ініціалізація>; <вираз>; <модифікація>) <оператор>;*

Ініціалізація використовується для оголошення та/або присвоєння початкових значень величин, використовуваних в циклі в якості параметрів (лічильників). У цій частині можна записати декілька операторів. розділених комою. Областю дії змінних, оголошених в частині ініціалізації циклу, є цикл і вкладені блоки. Ініціалізація виконується один разів на початку виконання циклу.

Вираз визначає умова виконання циклу: якщо його результат правдивий, цикл виконується. Істинність висловлювання перевіряється перед кожним виконанням тіла циклу, таким чином, цикл з параметром реалізований як цикл з передумовою. У блоці вираз через кому можна записати декілька логічних виразів, тоді кома рівносильна операції логічне І (&&).

Модифікація виконується після кожної ітерації циклу і служить зазвичай для зміни параметрів циклу. У частині модифікація можна записати декілька операторів через кому.

Оператор (простий або складений) являє собою тіло циклу.

Кожна з частин оператора **for** (ініціалізація, вираз, модифікація, оператор) може бути відсутня, але крапку з комою, яка визначає позицію частини, яку пропускають, треба залишити.

Приклад формування рядка складається з чисел від 0 до 9 розділених пробілами:

```
strings= “”; // Ініціалізація рядка
for (inti = 0; i <= 9; i++) // Перерахування всіх чисел
s+= i.ToString() + “ ”;// Додаємо число і пробіл
MessageBox.Show(s.ToString()); // Показуємо результат
```

Даний приклад працює наступним чином. Спочатку обчислюється початкове значення змінної **i**. Потім, поки значення **i** менше або дорівнює 9, виконується тіло циклу і знову обчислюється значення **i**. Коли значення **i** стає більше 9, умова стає хибною і управління передається за межі циклу.

Практична частина.

Обчислити і вивести на екран таблицю значень функції y=a*ln(x) при x, що змінюється від x_0 до x_k з кроком dx, a - константа.

Панель діалогу представлена на рис. 3.1 Текст обробника натискання кнопки *Обчислити* наведено нижче.

```
private void buttonl_Click(object sender, EventArgse) {
    //    Считываниеначальныхданныхdouble    x0    =
    Convert.ToDouble(textBoxl.Text);   double   xk=
    Convert.ToDouble(textBox2.Text);  double  dx  =
    Convert.ToDouble(textBox3.Text);   double   a   =
    Convert.ToDouble(textBox4.Text);
    textBoxS.Text=  "Работувыполнилст.  ИвановМ.А."
    +
        Environment.NewLine;
        // Цикл для табулюванняфункції
        doublex = x0;
    while(x<= (xk+ dx/ 2)) {
    doubley = a * Math.Log(x);
        textBoxS.Text+= "x=" + Convert.ToString(x) +
                            "; y=" + Convert.ToString(y) +
                            Environment.NewLine;
              x = x + dx;
     }
}
```

Після налагодження програми слід перевірити правильність роботи програми за допомогою контрольного прикладу, наведеного на рис 3.1. Встановіть точку зупину на оператор перед циклом і запустить програму. Після

потрапляння на точку зупинки, виконайте покроково програму і простежте, як змінюються всі змінні в процесі виконання.

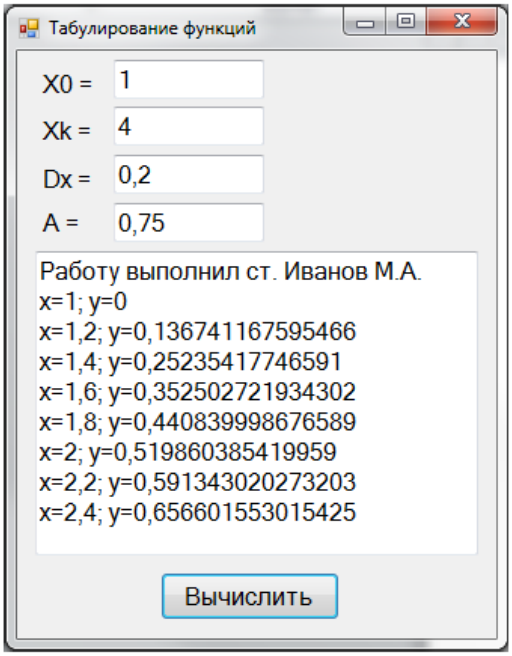

Рисунок 3.1 Вікно програми для табулювання функцій

Контрольні запитання:

1. Що таке цикл? Які вони бувають.
2. Опишіть поняття шаг циклу
3. Опишіть поняття зациклення, як його позбутися
4. Чим відрізняються цикли?
5. Для чого потрібна точка зупину?

Індивідуальне завдання:

Складіть програму табулювання функції, виведіть на екран значення x і y. Відкоригуйте елементи управління на формі у відповідності зі своїм варіантом завдання.

1) $y = 10^{-2} bc / x + \cos\sqrt{a^3 x}$,
$x_0 = -1.5; x_k = 3.5; dx = 0.5;$
$a = -1.25; b = -1.5; c = 0.75;$

2) $y = 1.2(a - b)^3 e^{x^2} + x$,
$x_0 = -0.75; x_k = -1.5; dx = -0.05;$
$a = 1.5; b = 1.2;$

3) $y = 10^{-1} ax^3 tg(a - bx)$,
$x_0 = -0.5; x_k = 2.5; dx = 0.05;$
$a = 10.2; b = 1.25;$

4) $y = ax^3 + \cos^2(x^3 - b)$,
$x_0 = 5.3; x_k = 10.3; dx = 0.25;$
$a = 1.35; b = -6.25;$

5) $y = x^4 + \cos(2 + x^3 - d)$,
$x_0 = 4.6; x_k = 5.8; dx = 0.2;$
$d = 1.3;$

6) $y = x^2 + tg(5x + b / x)$,
$x_0 = -1.5; x_k = -2.5; dx = -0.5;$
$b = -0.8;$

7) $y = 9(x + 15\sqrt{x^3 + b^3})$,
$x_0 = -2.4; x_k = 1; dx = 0.2;$
$b = 2.5;$

8) $y = 9x^4 + \sin(57.2 + x)$,
$x_0 = -0.75; x_k = -2.05; dx = -0.2;$

9) $y = 0.0025bx^3 + \sqrt{x + e^{0.82}}$,
$x_0 = -1; x_k = 4; dx = 0.5;$
$b = 2.3;$

10) $y = x \cdot \sin(\sqrt{x + b - 0.0084})$,
$x_0 = -2.05; x_k = -3.05; dx = -0.2;$
$b = 3.4;$

11) $y = x + \sqrt{|x^3 + a - be^x|}$,
$x_0 = -4; x_k = -6.2; dx = -0.2;$
$a = 0.1;$

12) $y = 9(x^3 + b^3) tgx$,
$x_0 = 1; x_k = 2.2; dx = 0.2;$
$b = 3.2;$

13) $y = |x - b|^{1/2} / |b^3 - x^3|^{3/2} + \ln|x - b|$,
$x_0 = -0.73; x_k = -1.73; dx = -0.1;$
$b = -2;$

14) $y = (x^{5/2} - b)\ln(x^2 + 12.7)$,
$x_0 = 0.25; x_k = 5.2; dx = 0.3;$
$b = 0.8;$

15) $y = 10^{-3}|x|^{5/2} + \ln|x + b|$,
$x_0 = 1.75; x_k = -2.5; dx = -0.25;$
$b = 35.4;$

16) $y = 15.28|x|^{-3/2} + \cos(\ln|x| + b)$,
$x_0 = 1.23; x_k = -2.4; dx = -0.3;$
$b = 12.6;$

Лабораторна робота № 4

Мета: вивчити основні поняття, що відносяться до класів і об'єктів, освоїти динамічне створення об'єктів в програмному коді. Скласти програму, використовуючи вивчені поняття.

Теоретичні відомості.

Класи і об'єкти

В об'єктно-орієнтованому підході існують поняття клас і об'єкт.

Клас - це програмна одиниця, яка задає загальний шаблон для конкретних об'єктів. Клас містить всі необхідні описи змінних, властивостей і методів, які ставляться до об'єкту. Прикладом класу в реальному житті являється поняття «автомобіль»: як правило, автомобіль містить деяку кількість коліс, дверей, має якийсь колір, але ці конкретні деталі в класі не описуються.

Об'єкт - це екземпляр класу. Властивості об'єкта містять конкретні дані, характерні для даного екземпляра. У реальному житті прикладом об'єкта буде конкретний екземпляр автомобіля з 4 колесами, 5 дверцятами і синього кольору.

Динамічне створення об'єктів

Найчастіше для розміщення на формі кнопки, поля введення або інших керуючих елементів використовується дизайнер середовища VisualStudio: потрібний елемент виділяється в панелі елементів і розміщується на формі. Однак іноді створювати елементи потрібно вже в процесі виконання програми. Оскільки кожен елемент керування являє собою окремий клас, його приміщення на форму програмним способом включає кілька кроків:

• Створення екземпляра класу

• Прив'язка його до форми

• Настройка розташування, розмірів, тексту і т. п.

Наприклад, щоб створити кнопку, потрібно виконати наступний код (його слід розмістити в обробника повідомлення **Load** або в якому-небудь іншому методі):

Button b = newButton();

Тут оголошується змінна **b** відноситься до класу **Button**, як і в попередніх лабораторних роботах. Однак далі йде щось нове: з допомогою оператора **new** створюється екземпляр класу **Button** і посилання на нього присвоюється змінній **b**. При цьому виконується цілий ряд додаткових дій: виділяється пам'ять під об'єкт, ініціалізуються всі властивості і змінні.

Далі потрібно додати об'єкт на форму. Для цього служить властивість Parent, яке визначає батьківський елемент, на якому буде розміщена кнопка:

b.Parent = this;

Ключове слово **this** відноситься до того об'єкта, в якому розміщений що виконується в даний момент метод.

Оскільки всі методи в лабораторних роботах розміщуються в класі форми, то й **this** відноситься до цього конкретного екземпляра форми.

Замість форми кнопку можна помістити на інший контейнер. Наприклад, якщо на формі є елемент управління Panel, то можна помістити кнопку на нього таким чином:

```
b.Parent = panell;
```

Щоб визначити положення і розміри кнопки потрібно використовувати властивості Location та Size:

```
b.Location = new Point(10y 20);
b.Size = new Size(200y 100);
```

Зверніть увагу, що Location та size - це теж об'єкти. Хоча всередині у Location містяться координати x і y, які визначають лівий верхній кут об'єкта, не

вийде поміняти одну з координат, потрібно міняти цілком весь об'єкт Location. Те ж саме відноситься і до властивості Size.

Насправді, кожен раз, коли форму поміщається новий елемент керування або вносяться якісь зміни властивостей елементів управління,VisualStudio генерує спеціальний службовий код. який проробляє наведені вище операції по створенню та налагодженню елементів управління. Спробуйте помістити на форму кнопку, змінити в неї які-небудь властивості, а потім знайдіть у браузері рішень гілку форми Form1, розгорніть її і зробіть подвійне клацання по гілці Form1.Designer.cs. Відкриється файл з текстом програми на мові C#, яку середовище створило автоматично. Змінювати цей код вручну вкрай не рекомендується! Однак можна його вивчити, щоб зрозуміти принципи створення елементів управління в ході виконання програми.

Область видимості

Змінні, оголошені в програмі, мають область видимості. Це означає, що змінна, описана в одній частині програми, не обов'язково буде видно в інший. Ось найбільш часто зустрічаються ситуації:

• Змінні, описані усередині методу, не буде видно за межами цього методу. Наприклад:

```
     Void Method A ()
     {
     // Описуємо змінну delta
          Int delta = 7;
}
     Void Method B()
     {
          // Помилка: змінна delta в цьому методі невідома!
          Int gamma= delta + 1;
     }
```

• Змінні, описані всередині блоку або складеного оператора, видно тільки всередині цього блоку. Наприклад:

```
Void Method() {
if (a == 7)
{
Int b = a + 5;
}
// Помилка: змінна b невідома!
MessageBox.Show(b.ToString());
}
```

• Змінні, описані усередині класу, є глобальними і доступні для всіх методів цього класу, наприклад:

```
classForml : Form
{
int a = 5;
void Method()
{    // Змінnaa дійсна у цьому випадку
     MessageBox.Show(a.ToString());
}}
```

Операції as та is

Часто буває зручно змінні різних класів записати в один список, щоб було легше його обробляти. Щоб перевірити, до якого класу належить об'єкт, можна використовувати оператор is: він повертає істину, якщо об'єкт належить вказаного класу. Приклад:

```
Button b = new Button();
if(b is Button)
```

```
            MessageBox.Show("Це кнопка!");
    else
                MessageBox.Show("це щось інше...");
```

Як правило, в загальних списках об'єкти зберігаються в “знеособленому” стані, так. щоб у всіх у них був лише мінімальний загальний для всіх набір методів і властивостей. Для того щоб отримати доступ до розширених властивостей об'єкта, потрібно привести його до класу з допомогою операції приведення **as**:

```
(some Object as Button).Text = "Це кнопка!";
```

Слід пам'ятати, що операція приведення спрацює тільки в тому випадку, якщо об'єкт спочатку належать того класу, до якого його намагаються привести (або сумісний з ним), в іншому випадку оператор **as** викине виняток і зупинить виконання програми. Тому більш безпечний підхід полягає в комбінованому застосуванні операторів **as** і **is**: спочатку перевіряємо сумісність об'єкта і класу, і тільки потім виконуємо операцію приведення:

```
if(someObject is Button)
        (someObject as Button).Text = "Це кнопка!";
```

В якості практичного прикладу використання цих операцій розглянемо приклад програми, яка перебирає всі елементи управління на формі, та на кнопках (але не у інших елементів управління!) замінює текст на п'ять зірочок «*****»;

```
private void Forml_Load(object sender, EventArgs e)
{  // Перебираємо всі елементи управління
        foreach (Control cin this.Controls)
            if (c isButton) // Кнопка?
                (c asButton).Text = "*****"; // так!
}
```

Відомості, що передаються в події

Коли відбувається якась подія (наприклад, подія click при натисканні на кнопку), в обробник цієї події передаються додаткові відомості про цю подію в параметрі **e**.

Наприклад, при натисканні кнопки миші на об'єкті виникає подія **Mousedick**. Для цього події параметр **e** містить цілий ряд змінних, які дозволяють дізнатися інформацію про натискання:

• **Button** - яка кнопка була натиснута;

• **Clicks** - скільки разів була натиснута і відпущена кнопка миші:

• **Location** - координати точки, на яку вказував курсор в момент натискання, у вигляді об'єкта класу **Point**:

• X I Y - ті ж координати, тільки у вигляді окремих змінних.

Індивідуальні завдання

Розробити програму, згідно виданого варіанту. Якщо в індивідуальному завданні використовується елемент Panel, змінити його колір, щоб він візуально виділявся на формі. Якщо використовується елемент Label, не забудьте дати йому який-небудь текст, інакше він не буде видний на формі.

1. Розробити програму, яка динамічно породжує на вікні кнопки. Лівий верхній кут кнопки визначається місцем розташування курсору при натисканні. Вивести напис на кнопці з координатами її лівого верхнього кута.

2. Розробити програму, яка динамічно породжує на вікні кнопки та поля вводу. Лівий верхній кут елемента керування визначається місцем розташування курсору при натисканні. Кнопка породжується, якщо курсор знаходиться в лівій половині вікна, в іншому випадку породжується поле вводу.

3. На формі розміщений елемент управління Panel. Написати програму. яка при клацанні миші на елементі управління Panel додає в нього кнопки Button, а при клацнути на формі у неї додаються поля введення TextBox.

4. Написати програму, яка добавляє на форму кнопки. Кнопки додаються у вузли прямокутної сітки. Відстані між кнопками і відстані між крайньою кнопкою і межею вікна повинні бути рівні як по горизонталі, так і по вертикалі.

5. Розробити програму, яка при клацанні миші динамічно породжує на вікні кнопки або поля для вводу. Кожний парний елемент управління є кнопкою, непарний - полем введення. Лівий верхній кут кнопки визначається місцем розташування курсору при натисканні. Для поля введення положення курсору визначає координати правого нижнього кута.

6. Створити програму з кнопкою, влучної і полем введення. При клацанні на відповідний елемент на формі динамічно повинен створюватися подібний йому елемент. Передбачити можливість виведення кількості кнопок, міток та полів введення.

7. Створити програму, яка додаєрізні елементи керування на форму і на панелі Panel. Тип елементів керування вибирається випадковим чином. Передбачити можливість виведення інформації про кількість елементів за типами та інформацію про розташування елементів.

8. Розробити програму, яка додає на форму послідовність елементів управління випадкової довжини. Тип елементів управління визначається випадковим чином. Передбачити можливість виведення інформації про кількість елементів за типами.

9. Написати програму, яка динамічно породжує на вікні кнопки або позначки. Лівий верхній кут елемента керування визначається місцем розташування курсору при натисканні. При натисканні правої кнопки миші на формі з неї видаляються всі кнопки.

10. Розробити програму з двома кнопками на формі. При натисканні на першу на форму додається одна панель Panel. При натисканні на другу кнопку у кожну панель додається поле вводу.

Контрольні запитання:

1. Що розуміється під терміном «клас»?
2. Які елементи визначаються у складі класу?
3. Яке співвідношення понять «клас» і «об'єкт»?
4. Що розуміється під терміном «члени класу»?
5. Які члени класу Вам відомі?
6. Які члени класу містять код?
7. Які члени класу містять дані?
8. Перерахуйте п'ять різновидів членів класу специфічних для мови C#.
9. Що розуміється під терміном «конструктор»?
10. Скільки конструкторів може містити клас мови C#?
11. Наведіть синтаксис опису класу в загальному вигляді. Проілюструйте його фрагментом програми на мові C#.

Лабораторна робота № 5

Мета: Вивчити способи отримання випадкових чисел. Написати програму для роботи з масивами.

Теоретичні відомості.

Робота з масивами

Масив - це набір елементів одного і того ж типу, об'єднаних спільним ім'ям. Масиви в C# можна використовувати за аналогією з тим, як вони використовуються в інших мовах програмування. Однак З#-масиви мають суттєві відмінності: вони відносяться до посилальних типів даних, більше того - реалізовані як об'єкти. Фактично ім'я масиву є посиланням на область купи (динамічної пам'яті), в якій послідовно розміщується набір елементів певного типу. Виділення пам'яті під елементи відбувається на етапі ініціалізації масиву. А за звільненням пам'яті стежить система збирання сміття - невикористані масиви автоматично утилізуються даною системою.

Одновимірний масив - це фіксована кількість елементів одного і того ж типу, об'єднаних загальним ім'ям, де кожен елемент має свій номер. Нумерація елементів масиву в C# починається з нуля, тобто, якщо масив складається з 10 елементів, то його елементи матимуть такі номери: 0, 1,2, 3, 4. 5. 6, 7. 8. 9.

Одновимірний масив в C# реалізується як об'єкт, тому його створення являє собою двоступінчастий процес. Спочатку оголошується посилальна змінна на масив, потім виділяється пам'ять під необхідну кількість елементів базового типу, і посилальної змінної присвоюється адреса нульового елемента в масиві. Базовий тип визначає тип даних кожного елемента масиву. Кількість елементів, які будуть зберігатися в масиві, визначається розмір масиву.

У загальному випадку процес оголошення змінної типу масив, і виділення необхідного обсягу пам'яті може бути розділене. Крім того на етапі оголошення

масиву можна зробити його ініціалізацію. Тому для оголошення одновимірного масиву може використовуватися одна з наступних форм запису:

тип[] ім'я_масиву;

В цьому випадку описується посилання на одновимірний масив, яке в подальшому може бути використано для адресації на вже існуючий масив. Розмір масиву при такому оголошенні не встановлюється. Приклад, в якому оголошується масив цілих чисел з ім'ям a:

Int[] a;

Інша форма оголошення масиву включає і його ініціалізацію зазначеною кількістю елементів:

Тип[] ім'я_масиву = new тип[розмір];

В цьому випадку оголошується одновимірний масив зазначеного типу і виділяється пам'ять під вказана кількість елементів. Адреса цієї області пам'яті записується в посилальну змінну. Елементи масиву ініціалізуються значеннями, які прийняті за замовчуванням для даного типу: масиви числових типів ініціалізуються нулями, рядкові змінні - порожніми рядками, символи пробілами, об'єкти посилальних типів - null. Приклад такого оголошення:

int [] a = new int[10];

Тут виділяється пам'ять під 10 елементів типу **int**.

Нарешті, третя форма запису дає можливість відразу ініціалізувати масив конкретними значеннями:

Тип [] имя_массива = {список ініціалізації};

При такому записі виділяється пам'ять під одновимірний масив, розмірність якого відповідає кількості елементів у списку ініціалізації. Адресу цієї області пам'яті записано в посилальну змінну. Значення елементів масиву відповідає списку ініціалізації. Приклад:

Int[] a= { 0, 1, 2, 3 };

В даному випадку буде створений масив **a**, що складається з чотирьох елементів. і кожен елемент буде ініціалізований черговим значенням зі списку.

Звернення до елементів масиву відбувається за допомогою індексу: для цього потрібно вказати ім'я масиву та в квадратних дужках його номер. Наприклад: a[0], b[10], c [i]. Слід пам'ятати, що нумерація елементів починається з нуля!

Так як масив являє собою набір елементів, об'єднаних спільним ім'ям, то обробка масиву зазвичай проводиться в циклі. Наприклад:

```
int[] туArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
for (inti = 0; i < 10; i++)
MessageBox.Show(myArray[i]);
```

Випадкові числа

Одним із способів ініціалізації масиву є завдання визначення елементів через випадкові числа. Для роботи з випадковими числами використовуються клас Random. Метод Random.Next створює випадкове число в діапазоні значень від нуля до максимального значення типу int (його можна дізнатися за допомогою властивості lnt32.MaxValue). Для створення випадкового числа в діапазоні від нуля до якого-небудь іншого позитивного числа використовується перевантаження методу Random. Next (int 32) - єдиний параметр методу вказує верхню межу діапазону, сама межа в діапазон не включається. Для створення випадкового числа в іншому діапазоні використовується перевантаження методу Random.Next(Int32, Int32) - перший аргумент задає нижню межу діапазону, а другий - верхню.

Двомірні масиви

Багатовимірні масиви мають більше одного виміру. Найчастіше використовуються двовимірні масиви, які являють собою таблиці. Кожен елемент такого масиву має два індексу, перший визначає номер рядка, другий - номер стовпця, на перетині яких знаходиться елемент. Нумерація рядків та стовпчиків

починається з нуля. Оголосити двовимірний масив можна одним із запропонованих способів:

```
тип[,] ім'я_масиву;
тип[,] ім'я_масиву = new тип[размер1, размер2];
тип[,] ім'я_масиву =
          {{елементи 1-оїстроки},
          ...
          {елементи п-оїстроки}};
тип[,]ім'я_масиву = newтип[,]
          {{елементи 1-оїстроки}, • • •
          {елементи п-оїстроки}};
```

В якості прикладу розглянемо код. який будує «таблицю множення» - кожна комірка буде містити значення, що дорівнює добутку номера рядка і номера стовпця:

```
// Оголошення двовимірного масиву
int[j] mul= new int[10,10];
// Заповнення масиву
for (inti = 0; i < 10; i++)
     for (intj=0;  j< 10; j++)
     mul[i, j] = i * j;
```

Елемент управління DataGridView

При роботі з двомірними масивами введення і виведення інформації на екран зручно організовувати у вигляді таблиць. Елемент управління DataGridView може бути використаний для відображення інформації у вигляді двовимірної таблиці. Для звернення до комірки в цьому елементі необхідно вказати номер рядка та номер стовпця. Наприклад:

```
DataGridViewl.Rows[2].Cells[7].Value= "*";
```

Цей код запише у другий рядок і сьомий стовпець знак зірочки.

Практична частина

Створіть форму з елементами управління як наведено на рис. 5.1. Опишіть одновимірний масив. Створіть обробник події для кнопок (код наведений нижче). Ця програма замінює всі від'ємні числа нулями. Перевірте правильність виконання програми. Модифікуйте програму у відповідності з індивідуальним завданням.

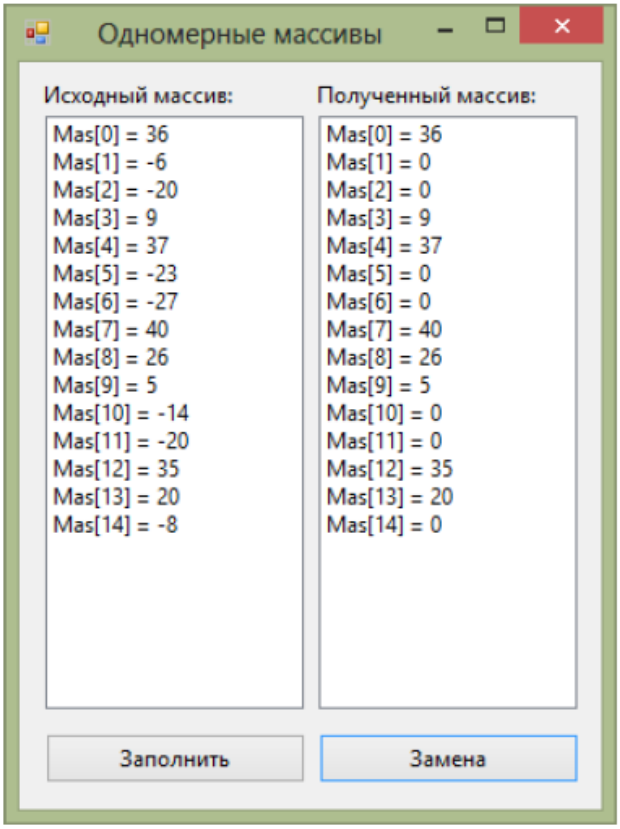

Рисунок 5.1 Вікно програми для роботи з одновимірними масивами

Приклад.

```
     // Глобальну змінну можуть бачити всі методи
 Int [] Mas = new int [15];
     // Заповнення вихідного масиву
Privat void button1_Click (object sender, EventArgs e)
{
     //Очищаємо елемент управління
listBox1.Items.Clear();
```

```
     //Ініціалізуємо клас випадкових чисел
Random rand = new Random();
     //Генеруємо та виводимо 15 елементів
For (int I = 0; I < 15; i++)
  {
Mas[i] = rand.Next(-50,50);
listBox1.Items.Add(“Mas[“+i.ToString() + ”] = ” + Mas[i].ToString());
   }
}
     //Заміна негативних елементів нулями
Privat void button2_Click (object sender, EventArgs e)
 {
   //Очищуємо елемент управління
   listBox.2.Items.Clear();
  //Оброблюємо всі елементи
    For (int I = 0; I < 15; i++)
     {
      If (Mas[i] < 0 )
          Mas[i] = 0;
listBox2.Items.Add(“Mas[“+Convert.ToString(i)+“]= “+ Mas[i].ToString());
}   }
```

Індивідуальні завдання

1. Знайти суму від'ємних елементів матриці.
2. Знайти мінімальний елемент матриці і вказати його індекси.
3. Змінити порядок стовпців матриці за спаданням елементів.
4. Знайти добуток додатних елементів матриці, відмінних від 0.
5. Знайти суму парних елементів матриці.
6. У матриці упорядкувати рядки за спаданням елементів.
7. Знайти максимальний елемент матриці і вказати його індекси.

8. Знайти суму парних елементів матриці.

9. Підрахувати кількість нульових елементів матриці.

10. Знайти суму всіх елементів матриці, кратних 3.

Контрольні запитання:

1. Дайте визначення масиву.
2. Як описується двомірний масив у програмі?
3. Що таке розмірність матриці?
4. Як одержати доступ до певного елемента матриці?
5. Як у програмі організувати введення й вивід матриці?
6. Що таке індекс елементу?

Література

1. Павловская Т. А. «C#. Программирование на языкевысокогоуровня», Учебник для вузов
2. Фролов А. В., Фролов Г. В. «Язык C#. Самоучитель», Учебно-справочноеиздание
3. Ватсон Б. «C#4.0 на примерах». — СПб.: БХВ-Петербург, 2011. — 608 с: ил.
4. Мак-Дональд, Мэтью, Шпушта, Марио «MicrosoftASP.NET 2.0 с примерами на C# 2005 для профессионалов»
5. Мартынов Н.Н. «C# для начинающих»