МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

ПРИКАРПАТСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ВАСИЛЯ СТЕФАНИКА

В.М.Ткачук

ЛАБОРАТОРНИЙ ПРАКТИКУМ

ОСНОВИ ВІЗУАЛЬНОГО ПРОГРАМУВАННЯ

Івано-Франківськ
-2015-

УДК 681.3 (075.8)

Основи візуального програмування Лабораторний практикум / Укладач – кандидат ф.-м. наук, доцент Ткачук В.М. – Івано-Франківськ: 2015 р. - __с.

Лабораторний практикум присвячено вивченню технології розробки візуального інтерфейсу для програм Windows Forms з використанням пакета Microsoft Visual Studio 2008. Практикум є продовженням базового курсу «Програмування на С++» та містить необхідну довідкову інформацію про основні компоненти Windows Forms, приклади розв'язку типових задач у вигляді готових графічних додатків із ілюстрацією результату їх роботи та набір задач для індивідуального виконання. Тематика робіт відповідає робочій програмі курсу «Основи програмування» для студентів 1-го курсу спеціальності «програмна інженерія». Практикум орієнтований на студентів, що мають достатні навики роботи із сучасними засобами розробки програмного забезпечення та може використовуватися при проведенні лабораторних та практичних занять для студентів інших спеціальностей, пов'язаних з вивченням сучасного програмного забезпечення.

Рекомендовано до друку Вченою радою факультету математики та інформатики Прикарпатського національного університету імені Василя Стефаника, протокол № _ від _ листопада 2015 року.

Рецензенти:

КозленкоМ.І кандидат технічних наук,
доцент кафедри інформаційних технологій факультету математики та інформатики Прикарпатського національного університету імені Василя Стефаника

Горєлов В.О. кандидат технічних наук,
доцент кафедри інформатики
факультету математики та інформатики Прикарпатського національного університету імені Василя Стефаника

Лабораторна робота №1

Тема робота: Створення Windows додатків в середовищі MS Visual C++

Мета роботи: формування навичок використання різних елементів управління у додатках **Windows Forms Application**

Для виконання роботи необхідно знати:

- порядок створення додатку Windows Forms Application;
- призначення та можливості конструктора форм;
- використання компонентів Windows Forms;
- властивості компонентів та їх налаштування;
- основні елементи управління Windows Forms;
- процедури обробки подій;
- особливості вводу та обробки коду обробки подій.

Теоретичні відомості

Додатки, що розробляються у візуальному середовищі, фактично є графічними додатками для Windows, і це принципово відрізняє їх від консольних додатків. Характерною особливістю графічних додатків на мові C++ є використання можливостей об’єктно-орієнтованого програмування, і всі елементи програми є об’єктами. При створенні нових об’єктів на візуальній формі вони будуються із уже існуючих об’єктів. Завдяки цьому нові об’єкти наслідують властивості та можливості вихідних об’єктів, і які необхідно тільки доповнити необхідними новими функціями.

Додатки із графічним інтерфейсом, що використовують форми, створюються та управляються в середовищі CLR (Common Language Runtime –

загальномовне виконуюче середовище). Середовище виконує код та пропонує служби, які полегшують сам процес розробки.

Додаток Windows Forms Application дозволяє помістити у виконуваний код стандартні графічні елементи управління Windows. В процесі створення додатку у візуальному середовищі програмування MS Visual C++ користувачу надається можливість створення графічного інтерфейсу за допомогою Windows Forms. При цьому в ході створення графічного інтерфейсу в додатку на мові C/C++ необхідно забезпечити взаємодію користувача із елементами управління інтерфейсу. Для цього необхідно створити додаток, який являє собою повнофункціональну програму Forms.

Для цього необхідно знати:

- можливості використання конструктора форм **Form Design** для побудови графічного інтерфейсу користувача в додатку;
- можливості використання в формі елементів управління;
- можливості обробки подій, пов’язаних із елементами управління.

Розробка додатку для Windows в середовищі MS Visual C++ за допомогою конструктора Windows Forms

Створення нового додатку починається із створення порожньої графічної форми. Для цього після звпуску середовища MS Visual C++ необхідно виконати команду File/New/Project, вибрати мову програмування (Visual C++), тип проекту – CLR , а тип додатку - Windows Forms Application. Необхідно також ввести ім’я додатку та задати повний шлях до файлу проекту.

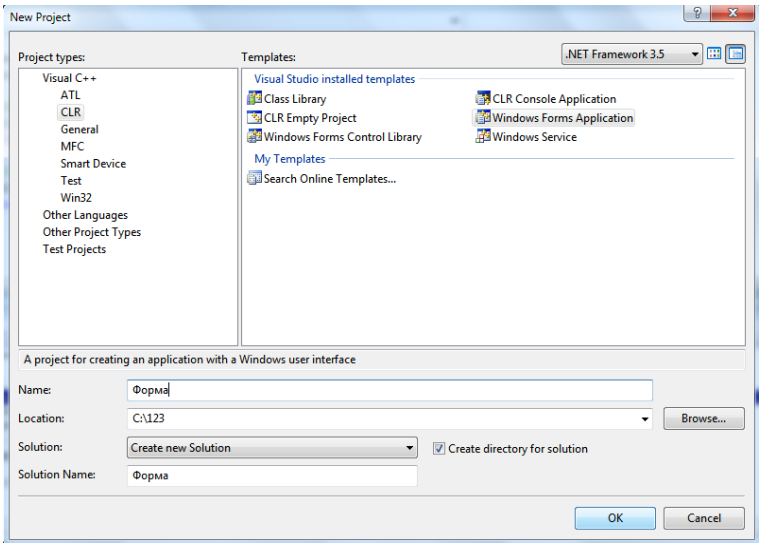


Рис.1. Вигляд вікна середовища **Microsoft Visual Studio** при створенні нового проекту типу Windows Forms Application.

Після створення нового проекту відкриється конструктор Windows Forms, у якому буде відображено порожню форму нового проекту.

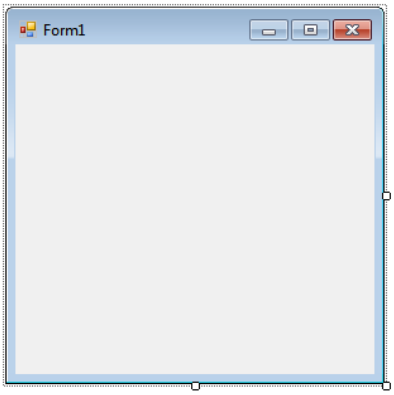


Рис.2. Вікно нової форми Form1.

Структура створеного проекту приведене на рисунку нижче

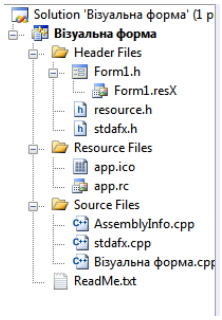


Рис.3. Структура новоствореного проекту.

Як і будь-який інший об’єкт у візуальному середовищі, форма характеризується набором властивостей, значеннями яких можна управляти за допомогою вікна властивостей об’єктів – **Properties.** Для виводу вікна властивостей форми необхідно відкрити контекстне меню форми, для чого помістити курсор мишки на форму, клікнути правою клавішею миші та виконати команду **Properties.** Властивості можна переглядати як в алфавітному порядку так і за категоріями.

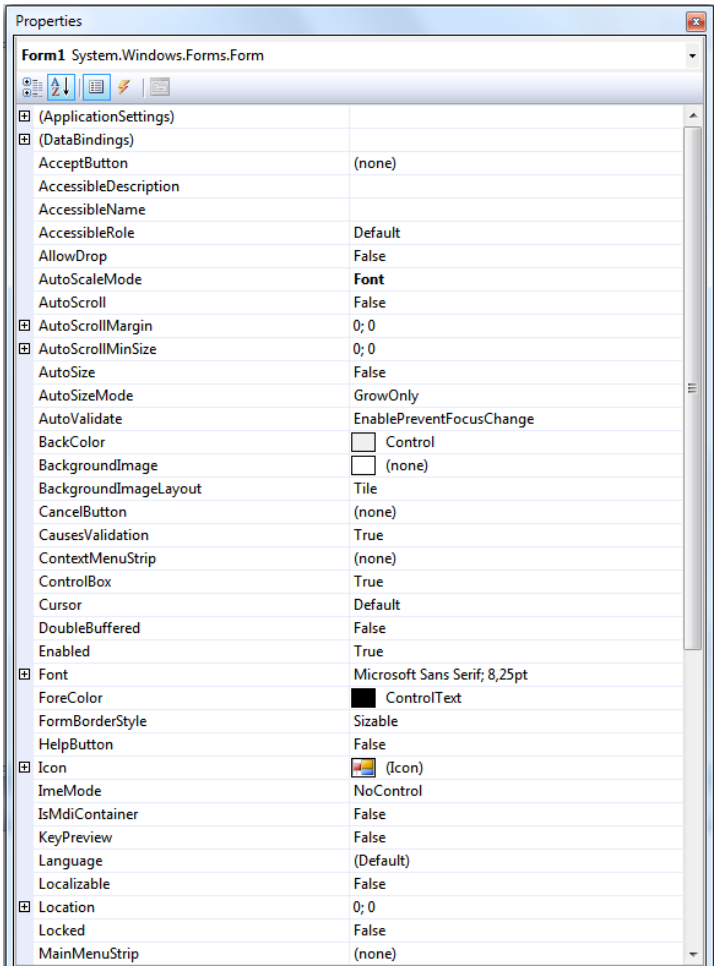


Рис.4. Частина вікна властивостей нової форми Form1.

Вікно **Properties** дозволяє задати в першу чергу загальний дизайн форми та її елементів управління. В таблиці приведено опис деяких найбільш часто використовуваних властивостей. При виборі значення властивостей, відмінних від прийнятого по замовчування, воно виділяється жирним шрифтом, що полегшує подальше визначення введених змін.

Таблиця 1. Деякі властивості форми

Властивість	Опис	Значення по замовчуванню
AcceptButton	Встановлюється значення кнопки, яка буде спрацьовувати при натисканні клавіші ***Enter***. Для того, щоби ця властивість була активною, необхідна наявність принаймі однієї кнопки, розміщеної на формі	None
BackColor	Колір форми.	Control
BackgroundImage	Використання зображення в якості фону	None
CancelButton	Задає кнопку, яка буде спрацьовувати при натисненні ***Esc***. Для того, щоби ця властивість була активною необхідна наявність хоча-б онієї кнопки на формі	None
ControlBox	Задає наявність чи відсутність трьох стандартних кнопок в верхньому правому куті форми: "Згорнути", "Розгорнути" та "Закрити"	
Cursor	Визначає вигляд курсору при його розміщенні на формі	Default
DrawGrid	Задає наявність чи відсутність сітки, яка допомогає формувати елементи управління. Сітка відображається тільки на етапі створення додатку	True
Font	Формат шрифту, що використовується для відображення тексту на формі в елементах управління	Microsoft Sans Serif; 8,25pt
FormBorderStyle	Визначає вигляд границь форми. Можливі значення: - ***None*** — форма без границь і рядка заголовку; - ***FixedSingle*** — тонкі границі без можливості зміни розміру користувачем; - ***Fixed3D*** — границі без можливості зміни розміру із тремірним ефектом; - ***FixedDialog*** — границі без можливості зміни, без іконки додатку; - ***Sizable*** — звичайні границі: користувач може міняти розмір границь; ***-FixedToolWindow*** — фіксовані границі, є тільки кнопка закриття форми. Такий вигляд мають панелі інструментів в додатках; - ***SizableToolWindow*** — границі із можливістю зміни розмірів, є тільки кнопка закриття	Sizable
Icon	Зображення іконки, що розміщується в заголовку форми. Підтримуються формати ***.ico***	
MaximizeBox	Визначає активність стандартної кнопки "Розгорнути" у верхньому правому куті форми	True

Таблиця 1(продовження). Деякі властивості форми

MaximumSize	Максимальний розмір ширини і висоти форми в пікселях. Форма приймає заданий розмір при натисненні на стандартну кнопку "Розгорнути"	0;0 (Во весь экран)
MinimizeBox	Визначає активність стандартної кнопки "Згорнути" в верхньому правому куті форми	True
MinimumSize	Мінімальний розмір ширини и висоти форми в пікселях. Форма приймає заданий розмір при зміні її границь користувачем	0;0
Size	Ширина і висота форми	300; 300
StartPosition	Визначає розміщення форми при запуску додатку. Можливими є наступні значення: - ***Manual*** — форма відкривається у верхньому лівому куті екрану; - ***CenterScreen*** — в центрі екрану; - ***WindowsDefaultLocation*** - розміщення форми по замовчуванню. Якщо користувач поміняв розмір форми, то при наступних її запусках вона буде мати той же самий вигляд та розміщення; - ***WindowsDefaultBounds*** — границі форми мають фіксований розмір; - ***CenterParent*** — в центрі батьківської форми	WindowsDefaultLocation
Text	Заголовок форми. У відмінності від властивості ***Name***, це власне назва форми, яка не використовується в коді	Form1, Form 2 и т.д.
WindowState	Визначає положення форми при її запуску. Можливими є наступні значення: - ***Normal*** — форма запускається із розмірами, заданими у властивості ***Size***; - ***Minimized*** — форма запускається із мінімальними розмірами, заданими у властивості ***MinimumSize***; - ***Maximized*** — форма розгортається на весь екран	Normal

Використання компонентів Windows Forms

Створення інтерфейсу додатку у середовищі MS Visual C++ зводиться до розміщення на формі необхідних компонент, ініціалізації подій та написання програм обробки цих подій у відповідності до розв’язуваної задачі. Компоненти Windows Forms розбиті на функціональні групи та знаходяться в палітрі компонентів **Toolbox.**

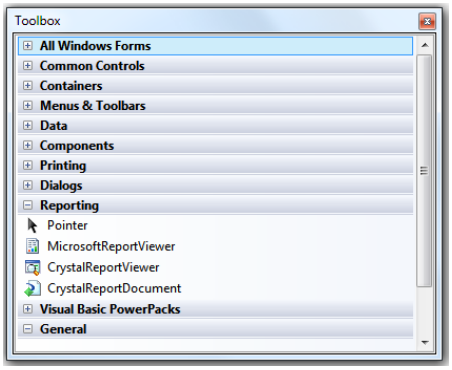


Рис.5. Функціональні групи палітри **Toolbox**.

Для відображення палітри **Toolbox** необхідно вибрати пункт головного меню **View** і списку, що появиться вибрати пункт ToolBox. Перелік всіх доступних компонент палітри можна побачити при активізації функціональної групи **All Windows Forms**.

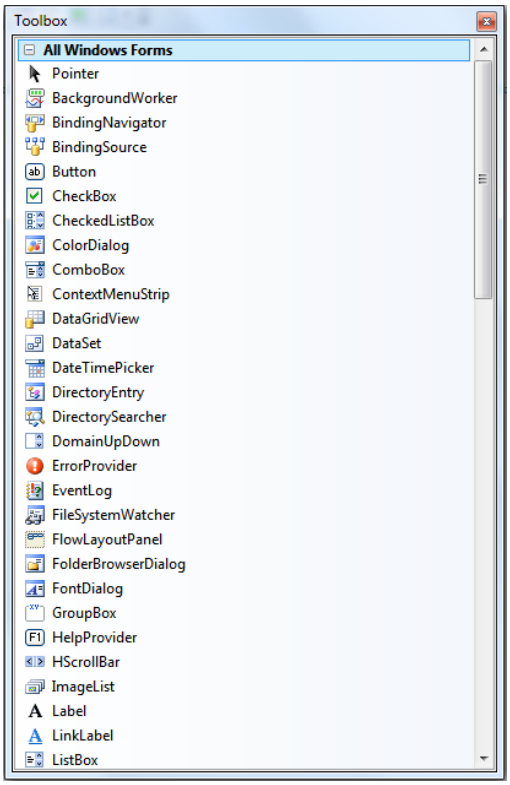


Рис.6. Частина повного переліку компоненти палітри **Toolbox.**

Для розміщення будь-якої компоненти на форму необхідно його знайти у палітрі компонент, клікнути на ньому мишкою, перевести курсор миші в необхідне місце на формі та знову клікнути мишкою. Будь-який об’єкт на робочому столі форми характеризується певним набором властивостей, таких

як імя, видимість, координати розміщення на формі, розмір і т.д. Властивості компонентів можна переглянути чи змінити за допомогою вікна властивостей Properties, в якому відображаються всі властивості активного об’єкта. Вікно відкривається при виконанні контекстного меню Properties.

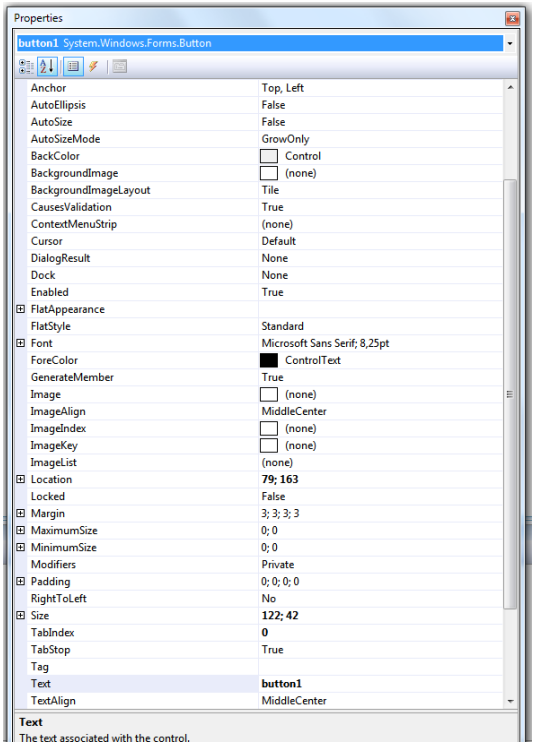


Рис.7. Частина меню Properties компоненти палітри **Toolbox** кнопка **(Button)**.

Windows Forms являє собою цілий набір засобів створення Windows - додатків, що виконуються в середовищі CLR. Форма являє собою головний контейнер, в який поміщаються віртуальні компоненти (кнопки, мітки, таблиці, поля прокрутки і т. д.), за допомогою яких і реалізується алгоритм розв’язку конкретної задачі.

До пакету Visual C++ входить стандартний набір понад 60-ти елементів управління, які можна використовувати при створенні форми для взаємодії із користувачем. Кожен із елементів управління має певне призначення та виконує певні функції.

Найбільш часто використовуваними елементами управління і компонентами Windows Forms оглянуті нижче

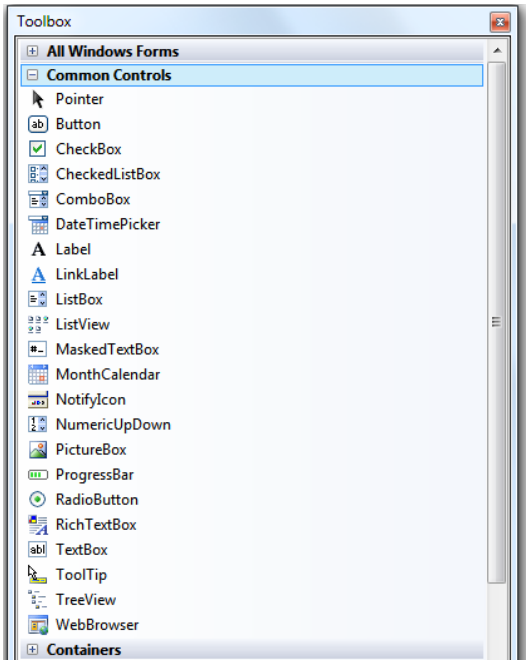


Рис.8. Елементи групи **Common Controls**.

Елементи групи **Common Controls** дозволяють реалізовувати ввід/вивід, відображення та редагування елементів управління. Деякі з них описано нижче:

- ***Button*** – дозволяє створювати елемент типу «кнопка» для запуску, зупинки чи переривання деякого обчислювального процесу;
- ***Label*** – відображає текст, який недоступний до редагування користувачем в ході виконання програми;
- ***TextBox*** –відображає текст, введений в процесі створення форми, який доступний до редагування в процесі виконання програми;
- ***ListBox*** –відображає список текстових елементів;
- ***ComboBox*** –відображає список, що розкривається;
- ***CheckedListBox*** –відображає список із полосою прокрутки, який складається із елементів із прапорцями;
- ***CheckBox*** –відображає прапорець із полем для тексту;
- ***RadioButton*** - відображає кнопку, що може бути включена або виключена;

- ***RichTextBox*** – дозволяє задавати текст у звичайному текстовому форматі;
- ***MaskedTextBox*** –обмежує формат даних, що вводяться користувачем;
- ***WebBrowser*** –дозволяє переміщатися по веб-сторінці в межах форми;
- ***LinkLabel*** – дозволяє добавляти веб-посилання на інше вікно чи на веб-вузол;
- ***DateTimePicker*** - відображає графічний календар, який дозволяє користувачу вибирати дату/час;
- ***ProgressBar*** –дозволяє спостерігати за ходом виконання в часі деякого процесу;
- ***PictureBox*** – дозволяє відобразити графічне зображення потрібного компонента.

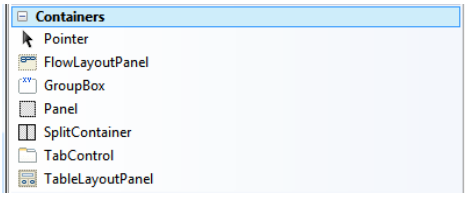


Рис.9. Елементи групи **Containers**

Елементи управління групи **Containers** палітри компонентів**:**

- ***Panel*** – групує набір елементів управління для забезпечення загальної поведінки;
- ***GroupBox*** – груповий контейнер, що використовується для розділення компонентів на підгрупи;

Рис.10. Елементи групи **Menus & Toolbars**

Елементи управління группи **Menus & Toolbars** палитри компонентів пов’язані із організацією меню:

- **MenuStrip** – створює головне меню додатку, за допомогою якого здійснюється управління роботою як всього додатку, так і його окремих блоків;
- **ContextMenuStrip** – створює контексні меню;
- **ToolStrip** – дозволяє створювати панелі інструментів із використанням різних стилів та принципів.

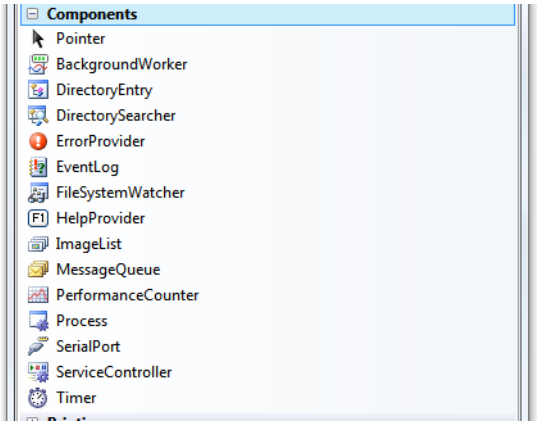


Рис.11. Елементи групи **Components**

Елементи управління групи **Components** палітри компонентів пов’язані із часом**:**

- ***Timer*** **–** задає лічильник часу;

Обробка подій

Після того, як потрібні компоненти інтерфейсу вибрані та поміщені на форму, необхідно створити процедуру обробки подій за допомогою вікна ***Properties***. Для цього вибираємо необхідний елемент управління і у вікні ***Properties*** натискаємо кнопку **Events** і зі списку доступних вибираємо необхідну подію

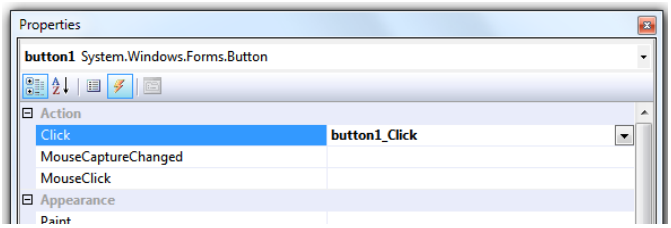


Рис.12. Налаштування процедури обробки події для кнопки **(Button)**.

Наступним кроком є написання відповідного коду для обробки події – наприклад натискання кнопки. Подальший процес програмування полягає в тому, що в тіло шаблона необхідно записати оператори, які визначають реакцію компонента на подію з урахуванням переданих функції фактичних значень її параметрів. Шаблони всіх графічних елементів форми генеруються автоматично по мірі їх розміщення на робочому столі. Для переходу до шаблону відповідного графічного елементу достатньо двічі клікнути мишкою на ньому.

```
private: System::Void
button1_Click(System::Object^  sender,
System::EventArgs^  e)
{

}
```

Рис.12. Приклад пустого обробника події для кнопки button1 (після подвійного клікання на ній)

Особливості редагування коду

Одночасно із створенням нової форми створюється спеціальний програмний модуль із ім’ям форми та розширенням h - Form1.h. Це h–файл (заголовочний файл), в якому знаходиться опис форми та обробки подій компонентів, які використовуються в проекті. Перейти до редактора коду із вікна форми можна за допомогою комбінації клавіш **<Ctrl>+<Alt>+<0**>, або виконавши команду **Code** головного меню **View**.

Із режиму написання коду в режим дизайнера форми можна переключитися комбінацією клавіш <Shift>+<F7>.

Робота із рядковими змінними

Із рядковими змінними типу ***String^*** працювати набагато зручніше на відміну від традиційних рядкових змінних типу ***char***. Для них визначено значна множина зручних методів (***Insert, Remove*** і.т.д), текстові поля вводу/виводу працюють тільки із ними, та і всі текстові поля властивостей компонент форми також є типу ***String^***. Приклад опису рядкової змінної із її одночасною ініціалізацією :

```
String^ st = "Текст рядкової змінної";
```

Приклад виводу рядкової змінної ***st*** типу ***String^*** у текстове поле ***textBox1***

```
textBox1->Text = Convert::ToString(st);
```

Приклад №1

Створення форми для вводу трьох чисел та їх впорядкування за зростанням при натисканні кнопки.

Використовуючи панель ***ToolBox*** розташовуємо на формі потрібні елементи керування та надаємо їй бажаного дизайну. В нашому прикладі:

- три текстові поля ***textBox1, textBox2, textBox3***;
- три підписи ***label*** для позначення цих полів;
- кнопка ***button1*** для запуску програми на виконання (обробки події – натискання кнопки).

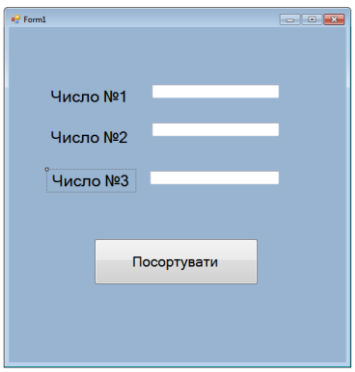


Рис.13. Вигляд форми після розміщення на ній елементів управління

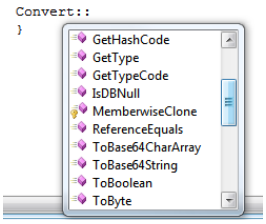
Після формування форми переходимо власне до написання коду програми. Обробка події (виконання операції сортування) повинна викликатися кнопкою «Посортувати», тому двічі клацаємо на ній і переходимо до вікна редагування коду із заготовкою функції для обробки події клік кнопки:

```
private: System::Void button1_Click(System::Object^
sender, System::EventArgs^  e) {

        }
```

В тіло цієї функції необхідно вписати код, що власне реалізує алгоритм сортування даних. Для перетворення даних, введених у елемент керування ***textBox*** в ціле число необхідно звернутися до класу ***Convert*** згідно наступного синтаксису:

```
int a;
a=Convert::ToInt32(textBox1->Text);
```

Тут дані, введені в текстове поле ***textBox1*** будуть перетворені до цілого формату і результат буде присвоєний змінній `a`. Можливі інші варіанти перетворення можна побачити після введення команди ***Convert::***



Наприклад, для перетворення введених до текстового поля даних до дійсного формату необхідно ввести наступний фрагмент коду:

```
double a;
a=Convert::ToDouble(textBox1->Text);
```

Для виводу результату в текстове поле необхідно скористатися зворотнім перетворенням даних до текстового формату:

```
Convert::ToString(a);
```

Вивід тексту у текстове поле **textBox1** матиме наступний вигляд:

```
textBox1->Text = Convert::ToString(a);
```

Тут **a** –ім'я змінної, значення якої буде перетворене до текстового формату.

Процедура читання даних із текстових полів **textBox1, textBox2, textBox3** та їх перетворення до дійсного формату матиме наступний вигляд:

```
double a,b,c;
a=Convert::ToDouble(textBox1->Text);
   b=Convert::ToDouble(textBox2->Text);
      c=Convert::ToDouble(textBox3->Text);
```

Остаточний код функції для обробки натискання кнопки **button1**:

```
double a,b,c;
         a=Convert::ToDouble(textBox1->Text);
            b=Convert::ToDouble(textBox2->Text);
               c=Convert::ToDouble(textBox3->Text);

if (a<b && a<c){
//вивід значення змінної a у текстове поле textBox1
//попередньо змінна a перетворюється до типу String
     textBox1->Text = Convert::ToString(a);

if (b<c){textBox2->Text = Convert::ToString(b);
     textBox3->Text = Convert::ToString(c);}
     else {textBox2->Text = Convert::ToString(c);
          textBox3->Text = Convert::ToString(b);};};

if (b<a && b<c){ Convert::ToString(b);
     textBox1->Text =
if (a<c) {textBox2->Text = Convert::ToString(a);
```

```
        textBox3->Text = Convert::ToString(c);}
        else {textBox2->Text = Convert::ToString(c);
            textBox3->Text = Convert::ToString(a);};};

if (c<a && c<b){
        textBox1->Text = Convert::ToString(c);

if (a<b) {textBox2->Text = Convert::ToString(a);
textBox3->Text = Convert::ToString(b);}
        else {textBox2->Text = Convert::ToString(b);
        textBox3->Text = Convert::ToString(a);};};
```

Примітка: при введенні дійсних чисел в текстові поля при виконанні графічної форми розділовий знак між цілою та дробовою частиною – кома!!!

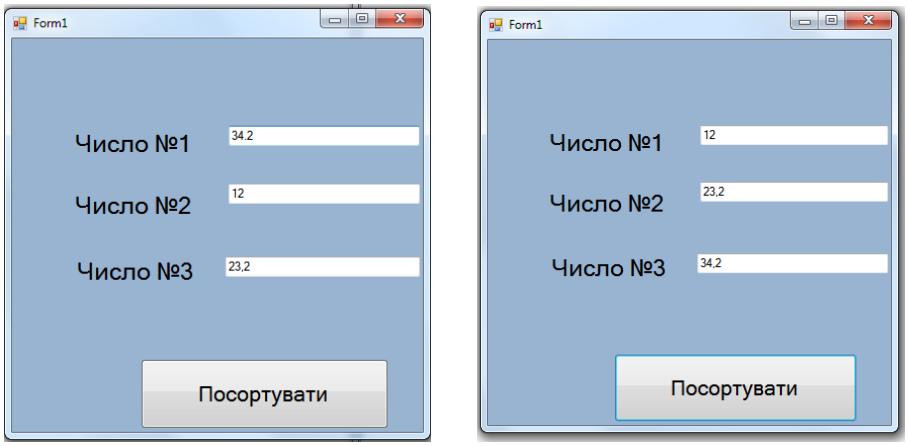


Рис.14. Результат роботи – до і після виконання операції сортування

Завдання для індивідуального виконання

Створити та запрограмувати візуальний проект ***Windows Form*** для розв'язання задачі згідно варіанту.

1. Дано три цілих числа. Знайти кількість додатних чисел в цьому наборі.
2. Дано два числа. Вивести спочатку більше, а потім менше з них.
3. Дано дві змінні дійсного типу: A, B. Перерозподілити значення даних змінних так, щоб A виявилось меншим із заданих значень, а B - більше. Вивести нові значення змінних A і B.
4. Дано дві змінні цілого типу: A і B. Якщо їхні значення не рівні, то присвоїти кожній змінної суму цих значень, а якщо рівні, то присвоїти змінним нульові значення. Вивести нові значення змінних.
5. Дано три числа. Знайти найменше з них.
6. Дано три числа. Знайти середнє з них (тобто число, розташоване між найменшим і найбільшим).
7. Дано три числа. Вивести спочатку найменше, а потім найбільше з даних чисел.
8. Дано три числа. Знайти суму двох більших з них.
9. На числовій осі розташовані три точки: A, B, C. Визначити, яка з точок B і C розташована ближче до A, і вивести цю точку і її відстань від точки A.
10. Дано координати точки, що не лежить на координатних осях OX та OY. Визначити номер координатної чверті, в якій знаходиться дана точка.
11. Дано цілочисельні координати трьох вершин прямокутника, сторони якого паралельні координатним осям. Знайти координати його четвертої вершини.
12. Дано номер року (додатне ціле число). Визначити кількість днів в цьому році, враховуючи, що звичайний рік нараховує 365 днів, а високосний.

13. Написати програму, яка вводить вік користувача і, якщо йому більше 18 років, повідомляє, що він має право голосу. В іншому разі вона обчислює, через скільки років користувач буде мати право голосу.

14. Дано координати двох точок. Визначити, чи розміщені вони на одному колі із центром в початку координат.

15. Із клавіатури вводиться номер місяця в році. Виводиться повідомлення про назву місяця та пору року. Якщо місяця із таким номером не існує – виводиться відповідне повідомлення.

16. Дано радіус кола та довжина сторони квадрата. Перевірити, чи пройде квадрат крізь коло.

17. Ввести із клавіатури три числа та знайти найменше та найбільше серед цих чисел.

18. Дано радіус кола та довжина сторони квадрата. Визначити, чи пройде круг через квадрат.

19. На площині задано коло радіусу R із центром в початку координат. Ввести координати точки та визначити, чи знаходиться вона на колі, чи в середині кола, чи за його межами.

20. Прямокутник задано координатами його вершин. Визначити, чи належить введена точка області прямокутника.

21. Два кола задано їхніми радіусами та координатами центрів. Визначити кількість спільних точок у цих кіл (одна, дві, чи ні однієї).

22. Із клавіатури вводиться порядковий номер місяця. Програма виводить назву пори року. Передбачити перевірку коректності вводу номера місяці.

23. Дано координати точки $M(x, y)$. Визначити, якому квадранту системи координат належить дана точка.

24. Дано функцію виду $y = ax^2 + bx + c$ та координати двох точок. Визначити, яка із точок знаходиться ближче до параболи.

25. Дано дві прямі лінії $ax + by + c = 0$ та $a_1x + b_1y + c_1 = 0$. Визначити, чи перетинаються вони.

26. Із клавіатури вводиться п’ять цілих чисел. Визначити кількість парник чисел та кількість чисел, кратних 3.

27. Коло задано радіусом та координатами центру. Визначити, як лежить початок координат відносно кола – в середині, на колі чи за його межами.

28. Із клавіатури вводиться оцінка в п’ятибальній системі. На екран виводиться відповідна оцінка в університетській системі: відмінно, добре, задовільно або незадовільно.

29. Дано три числа (в межах від 1 до 9 як константи). Із клавіатури вводяться три довільні числа, а на екран виводиться повідомлення про кількість вгаданих чисел.

30. Дано пряма $ax + by + c = 0$ та парабола $y = a_1x^2 + b_1x + c_1$. Визначити кількість точок перетину прямої з параболою: дві, одна, жодної.

Лабораторна робота №2

Тема робота: Створення Windows додатків в середовищі MS Visual C++ із використанням додаткових елементів вводу/виводу даних

Мета роботи: формування навичок використання різних елементів управління у програмах **Windows Forms**

Для виконання роботи необхідно знати:

- використання та налаштування *MessageBox;*
- вставка зображення на графічну форму;
- використання та налаштування ***DataGridView***.

Теоретичні відомості

Виведення повідомлень із використанням MessageBox

Для виведення повідомлень у окремому від графічного додатку вікні використовується стандартний клас **MessageBox**. Метод **Show** цього класу виводить заданий текст повідомлення користувачу:

Приклад:

```
MessageBox::Show("Сортування успішно виконано");
```

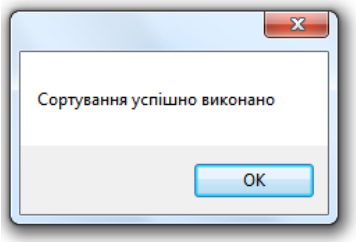


Рис.15. Результат роботи **MessageBox**

Крім тексту повідомлення, можна задати інші параметри діалогового вікна ***MessageBox***. Так, можна вказати, які кнопки необхідно розташувати у вікні діалогового вікна ***MessageBox***. Цей параметр задається константами з переліку ***MessageBoxButtons***.

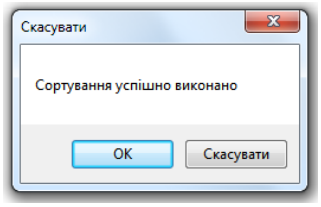
Таблиця 2.Можливі значення параметра *MessageBoxButtons*

Константа	**Кнопки, які відображаються у вікні MessageBox**
OK	OK
OKCancel	OK, Cancel
YesNo	Yes, No
YesNoCancel	Yes, No, Cancel
RetryCancel	Retry, Cancel
AbortRetryIgnore	Abort, Retry, Ignore

Наприклад

```
MessageBox::Show("Сортування успішно виконано","Скасувати",MessageBoxButtons::OKCancel);
```

Виведе наступне вікно ***MessageBox***



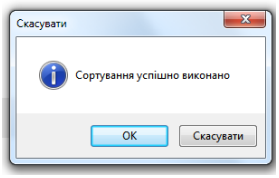
Метод ***MessageBox.Show*** дозволяє також вибирати один із кількох можливих значків для розташування в лівій частині діалогового вікна. Він задається у вигляді константи з переліку MessageBoxIcon

Константа	**Значок**
Asterisk, Information	
Error, Stop	
Exclamation, Darning	
Question	
None	Значок не виводиться

Наприклад

```
MessageBox::Show("Сортування успішно виконано","Скасувати",MessageBoxButtons::OKCancel,MessageBoxIcon::Asterisk);
```

виведе наступне вікно ***MessageBox***



Якщо у вікні ***MessageBox*** є декілька кнопок, то для того, щоб визначити, яку кнопку клацнув користувач, програма повинна проаналізувати значення, яке повертає метод ***MessageBox.Show***. Він може повернути одне з декількох значень з переліку ***DialogResult***

Можливі значення DialogResult

Константа	Кнопка, при виборі якої повертається ця константа
Abort	Abort
Cancel	Cancel
Ignore	Ignore
No	No
None	Модальне вікно діалогу продовжує працювати
OK	OK
Retry	Retry
Yes	Yes

Наприклад

```
DialogResult=(MessageBox::Show("Сортування успішно виконано","Скасувати",MessageBoxButtons::OKCancel,
        MessageBoxIcon::Asterisk)) ;
if(Convert::ToString(DialogResult)=="Cancel")
        textBox3->Text = "Скасовано";
```

Тут якщо буде натиснута кнопка ***Скасувати*** у третьому полі буде виведено повідомлення «***Скасовано***».

Вставка зображень на графічну форму

Для вставлення у форму зображень служить ***PictureBox***. Після розміщення за допомогою конструктора на формі ***PictureBox*** необхідно за допомогою вікна ***Properties***, вкладка ***Image*** задати малюнок із файлової системи комп'ютера та виконати його налаштування:

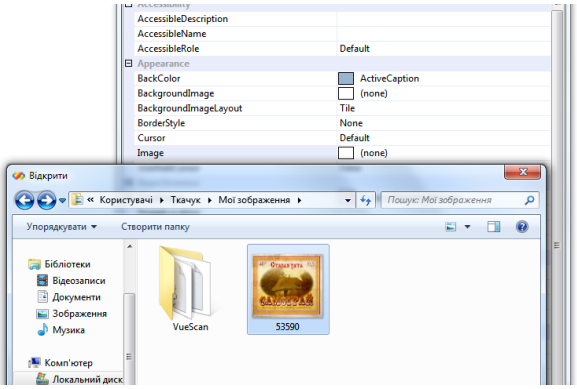


Рис.16. Вставка зображення за допомогою ***PictureBox***

За допомогою властивості ***SizeMode*** можна вписати зображення в межі всього створеного блоку ***PictureBox1***.вибравши пункт ***StretchImage***.

Створення форми з використанням *DataGridView*.

DataGridView – це елемент управління, який дозволяє відображати будь-який вид даних в комірках прямокутної сітки. Елементи управління ***DataGridView*** дозволяють відображати та редагувати прямокутний масив даних. Його модна також використовувати для відображення практично будь-яких даних, згенерованих безпосередньо при виконанні програми. Дані елементу управління відображаються в прямокутному масиві комірок, які можна набір рядків та стовпців. Кожен стовпець комірок має у верхній частині комірку заголовка, яка, як правило, містить пояснювальну текстову інформацію, а на початку кожного рядка знаходиться комірка його заголовка.

Кількість стовпців у ***DataGridView*** можна задати наступним чином (буде створено три стовпці):

```
dataGridView1->ColumnCount = 3;
```

Кількість рядків ***DataGridView*** можна задати наступним чином (буде створено пять рядків):

```
dataGridView1->Rows->Add(5);
```

Для кожного стовпця, при бажанні, можна задати заголовок наступним чином:

```
dataGridView->Columns[0]->Name="Name";
dataGrldView->Columns[1]->Name="Phone Number";
  dataGridView->Columns[2]->Name="Address";
```

Приклад створення Windows – додатку, який дозволяє створити матрицю заданої розмірності, ввести елементи матриці, або згенерувати їх за допомогою генератора випадкових чисел та підрахувати суму діагональних елементів матриці та кількість парних елементів.

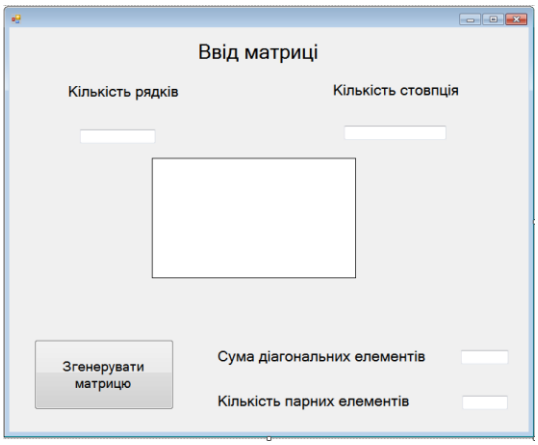


Рис.17. Можливий вигляд форми після розміщення на ній елементів управління

Код функції для обробки події - натискання кнопки ***button1*** матиме наступний вигляд:

```
int m, n,kil,sum;
/*Очищення стовпців компоненту DataGridView, якщо вони не порожні */
dataGridView1->Columns->Clear();
//Перевірка, чи введено кількість рядків та стовпців
if ((textBox1->Text!="")&&(textBox2->Text!=""))
{
```

```
//Читання кількості рядків
m = Convert::ToInt32(textBox1->Text);
//Читання кількості стовпців
n = Convert::ToInt32(textBox2->Text);
//Заповнення DataGridView стовпцями
dataGridView1->ColumnCount = n;
//Заповнення DataGridView рядками
dataGridView1->RowCount = m;}
else
//Вивід повідомлення у вікно MessageBox
{MessageBox::Show( "Заповніть форму даними", "Помилка
вволу даних ",
MessageBoxButtons::OK, MessageBoxIcon::Exclamation );}
//Опис масиву та аналіз його елементів
int A[50][50];
sum=0;
kil=0;
for (int i = 0; i < m; i++)
for (int j = 0; j < n; j++)
{
A[i][j] = rand()%100-50;
//Вивід згенерованого масиву у компоненту DataGridView
dataGridView1->Rows[i]->Cells[j]->Value=
Convert::ToString(A[i][j] );
if (i==j) sum=sum+A[i][j];
if (A[i][j]%2==0) {kil++;}
}
//Вивід результатів у відповідні вікна textBox
textBox3->Text=Convert::ToString (sum);
textBox4->Text=Convert::ToString (kil);
```

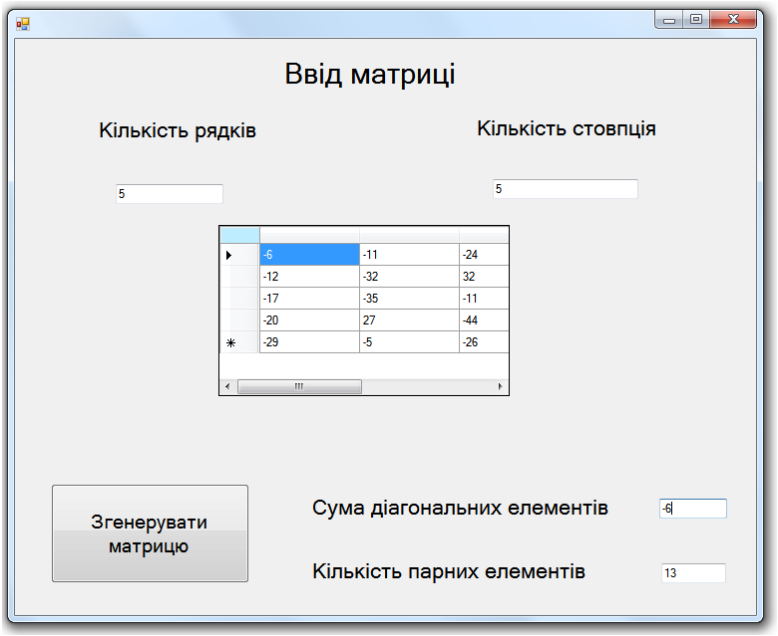


Рис.18. Результат роботи форми після виконання операції генерації матриці

Завдання для індивідуального виконання

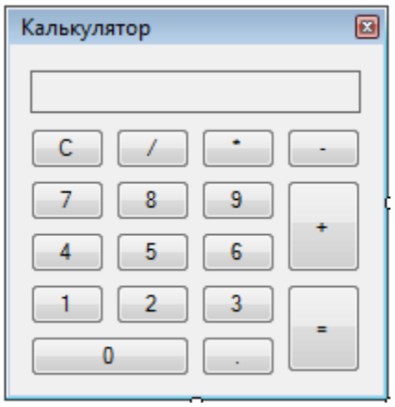
Завдання №1 Створити проект ***Windows Form*** для розв'язання задачі згідно варіанту.

№ варіанту	Завдання
1.	Визначити, чи є задана квадратна матриця A (5,5) симетричної щодо головної діагоналі.
2.	Дано цілочисловий масив A, який складається з 12 елементів. Створити та вивести масив C, який складається з остач ділення елементів масиву A на ціле число k
3.	Задано матриця У (4,4). Визначити, чи відсортовані всі елементи першого стовпчика за зростанням.
4.	Задано матриця К (5,5). Поміняти місцями максимальний елемент кожного рядка з першим елементом відповідного рядка.
5.	Переписати перші елементи кожного рядка матриці D (3,3), які більше 10, в масив B.
6.	Задано матриця Q (5,5). Замінити останній нуль в кожному рядку на 5.
7.	Задано матриця D (4,4). Визначити максимальний серед позитивних, мінімальний серед негативних і поміняти їх місцями.
8.	Задано матриця A (4,4). Замінити перший нуль в кожному стовпці на кількість нулів у цьому стовпці.
9.	Задано матриця F (9,3). визначити, чи рівні всі елементи першого стовпчика відповідних елементів головної діагоналі. Якщо ні, то поміняти їх місцями.
10.	Задано матриця C (5,5). Отримати вектор B, кожен елемент якого дорівнює кількості нулів, що стоять у стовпці матриці.
11.	Задано матриця У (4,4). Якщо в рядку є хоча б одна одиниця, то замінити цей рядок нулями.
12.	Задано матриця Q (3,3). Якщо на головній діагоналі стоїть нуль, то відповідний рядок замінити одиницями.
13.	Задано матриця D (4,4). Якщо максимальний елемент матриці стоїть на головній діагоналі, то всі елементи головної діагоналі зробити

	рівними максимальному.
14.	Задано матриця З (5,5). Якщо мінімальний елемент коштує в першому рядку, то всі елементи, які стоять в рядку за ним, замінити нулями.
15.	Задано матриця А (4,4). Якщо максимальний елемент матриці дорівнює сумі елементів першого рядка, то поміняти місцями перший рядок з тим рядком, де знаходиться максимальний елемент.
16.	Задано матриця А (4,4). Якщо максимальний елемент матриці дорівнює сумі елементів першого рядка, то поміняти місцями перший рядок з тим рядком, де знаходиться максимальний елемент.
17.	Знайти мінімальний по мордулю елемент та номер рядка і стовпця, у якому він знаходиться
18.	Дано цілочисловий масив А, який складається з 12 елементів. Створити та вивести масив С, який складається з непарних чисел масиву А, полічити кількість елементів масиву С.
19.	Дано вектор, який містить К елементів. Вилучити з нього елементи, які знаходяться між максимальним і мінімальним елементами. Вивести значення максимального і мінімального елементів.
20.	Дано масив А, який складається з 19 елементів. Вивести три перші від’ємні елементи цього масиву у порядку зростання та їх порядкові номерами.
21.	В заданих двох векторах А і В однакової розмірності N знайти окремо найбільше і найменше значення сум їх елементів.
22.	Дано одновимірний масив С, який складається з 15 елементів. Обчислити і вивести добуток непарних елементів і їх кількість.
23.	Дано одновимірний масив С, який складається з 16 елементів. Обчислити середнє арифметичне значення парних елементів масиву, які діляться на 3 з остачею 1.
24.	Дано цілочисловий одновимірний масив А, який складається з 14 елементів. Обчислити і надрукувати суму парних елементів, які знаходяться на непарних позиціях, і їх кількість.
25.	Дано матрицю Е розміром 4х6. Сформувати та вивести матрицю Q, значення елементів кожного стовпця якої обчислюється як різниця відповідних елементів двох суміжних стовпців матриці Е.
26.	Дано матрицю В розміром 5х6. Поділити елементи кожного рядка на

	елемент, який знаходиться в третьому стовпці цього рядка.
27.	Дано квадратну матрицю А 6-го порядку. Знайти суму елементів матриці, які розміщені в рядках з від'ємним елементом на головній діагоналі. Обчислити кількість таких рядків.
28.	Дано матрицю Т розміром 6х7. Знайти максимальний і мінімальний елементи для кожного стовпця матриці Т.
29.	Дана квадратна матриця С. Деякі елементи цієї матриці, які розміщені на головній діагоналі рівні нулю. Вилучити з матриці С ті стовпці, в яких елементи головної діагоналі рівні нулю. Вивести нову матрицю.
30.	Дано два одновимірні масиви, які складаються не більш як з 30 елементів кожний. Знайти півсуму максимальних елементів заданих масивів.

Завдання №2 Написати програму «калькулятор» з графічним інтерфейсом, що виконує чотири основні арифметичні дії.



Інтерфейс калькулятора

Лабораторна робота №3

Тема робота: Елементи управління роботою Windows додатків

Мета роботи: формування навичок використання різних елементів управління у програмах **Windows Forms**

Для виконання роботи необхідно знати:

- використання та налаштування радіо кнопки (***RadioButton***)*;*
- призначення та можливості використання елементу ***CheckBox***;
- призначення та використання елементу ***GroupBox***;
- використанням ***ListBox*** для відображення списку елементів;
- використання **`panel`** для побудови графіка функції;
- Використання горизонтальних та вертикальних полів прокрутки.

Теоретичні відомості

Створення форми з різними видами кнопок.

Для ілюстрації використання різного виду кнопок розглянемо наступний приклад:

додаток повинен виконувати наступні функції: радіокнопки (***RadioButton***) задають текст повідомлення, що буде виводитися по натисканню на звичайну кнопку. Прапорець (***CheckBox***) повинен визначати — чи виводити повідомлення, чи ні. Для цього створюємо новий проект, додаємо на форму один елемент керування ***GroupBox***, три елементи ***RadioButton***, один елемент ***CheckBox*** й один елемент ***Button***. Важливо, що всі три радіокнопки поміщені в один ***GroupBox*** для їх зв’язаної роботи. Інакше вони не будуть зв'язані між собою і працюватимуть незалежно одна від одної.

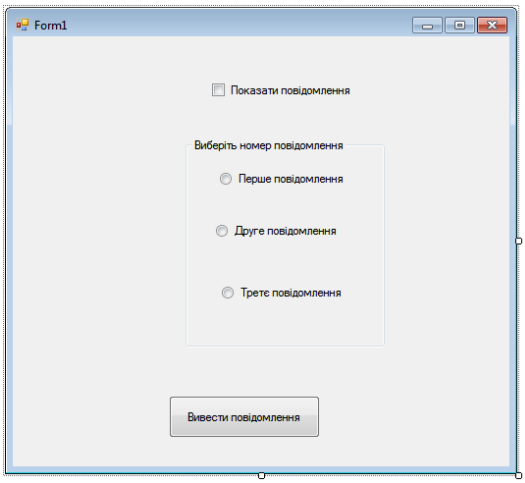
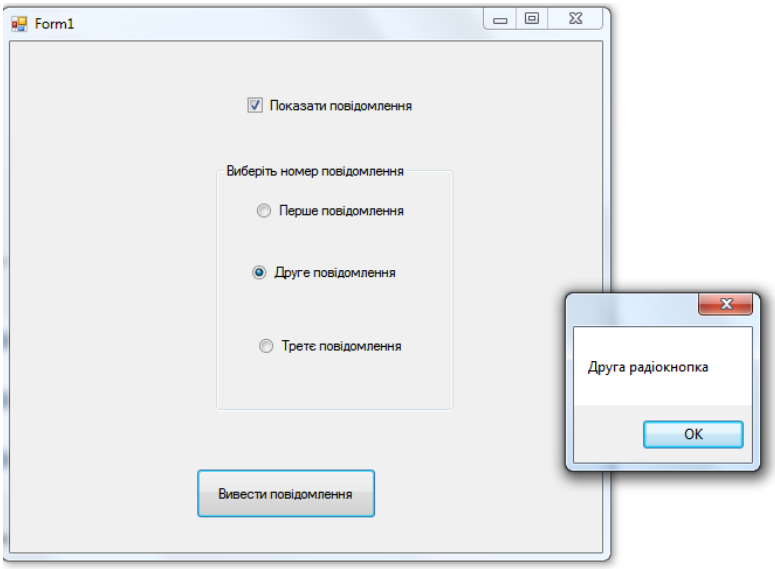


Рис.17. Проект форми для ілюстрації використання кнопок.

В тіло функції для кнопки ***button1*** введено наступний код:

```
// перевіряємо першу радіокнопку
     if (radioButton1->Checked == true)

     //перевіряємо, чи встановлений прапорець
if (checkBox1->Checked ==true) MessageBox::Show("Перша
          радіокнопка");;
// перевіряємо другу радіокнопку

     if (radioButton2->Checked == true)
//перевіряємо, чи встановлений прапорець

          if (checkBox1->Checked ==true)
          MessageBox::Show("Друга радіокнопка");;

// перевіряємо третю радіокнопку
     if (radioButton3->Checked == true)

     //перевіряємо, чи встановлений прапорець
          if (checkBox1->Checked ==true)
          MessageBox::Show("Третя радіокнопка");;
```

Відкомпілювавши та запустивши програму отримаємо:



На екрані нічого не буде виведено, якщо немає прапорця в полі «Показати повідомлення».

Створення форми з використанням бігунка, індикатора прогресу й регулятора чисельних значень.

Для ілюстрації роботи перерахованих компонент розглянемо приклад додатку у якому бігунок й елемент керування ***NumericUpDown*** управляють індикатором прогресу. При цьому бігунок і ***NumericUpDown*** повинні працювати синхронно: при зміні значення одного елемента значення іншого повинне змінюватися автоматично на ту ж величину.

Створимо новий Windows-додаток та додамо на форму наступні елементи керування:

- ***TrackBar*** ;
- ***ProgressBar*** ;
- ***NumericUpDown***.

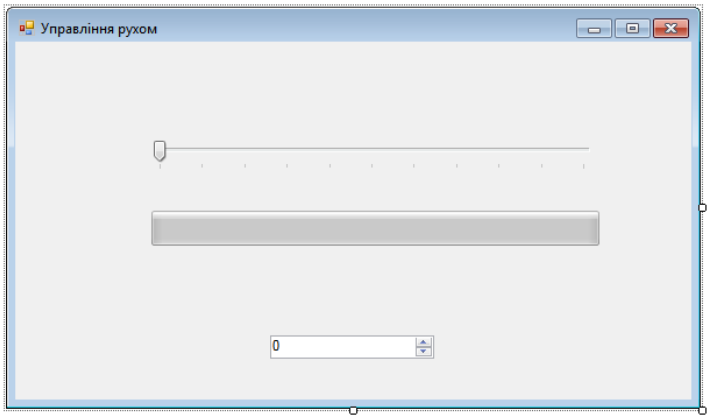


Рис. 18. Можливий варіант дизайну форми.

Для елементу ***TrackBar1***задамо наступні властивості:

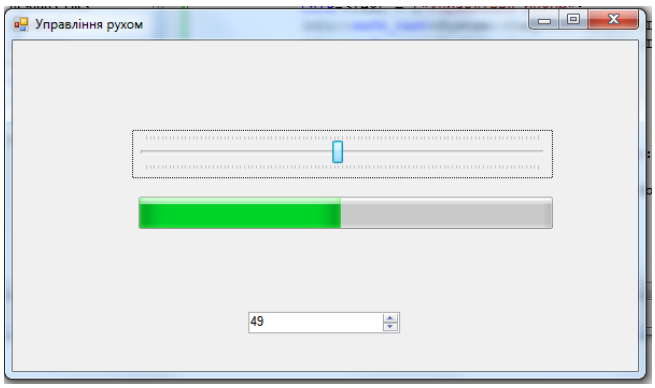
Maximum —100

TickStyle — Both

NumericUpDown1 і ***trackBar1*** є керуючими елементами, а ***progressBarl*** - керованим. Компонент ***TrackBar*** (індикатором прогресу) має подію ***Scroll***, що обробляє переміщення покажчика бігунка. У функції з ім'ям ***trackBar1_Scroll***. додамо наступний код:

```
int Value = trackBar1->Value;
   numericUpDown1->Value = Value;
     progressBar1->Value = Value;
```

Тепер при русі курсору бігунка ***TrackBar*** будуть змінюватися як положення індикатора прогресу ***progressBarl*** та значення елементу ***numericUpDown1***, як це показано нижче:



Для можливості синхронного керування за допомогою регулятора числових значень ***NumericUpDown*** додамо оброблювач події ***ValueChanged***

для елемента ***numericUpDown1***. В код програми для функції із імям ***numericUpDown1_ValueChanged*** введемо наступний код

```
int Value = (int)numericUpDown1->Value;
trackBar1->Value = Value;
progressBar1->Value = Value;
```

Як наслідок, при зміні положення бігунка індикатор прогресу і числовий регулятор змінять свої значення на відповідні величини. Індикатором прогресу можна також управляти за допомогою числового індикатора.

Створення форми з використанням ListBox.

ListBox являє собою елемент управління ***Windows*** для відображення списку елементів. Проілюструємо його використання на прикладі створення форми для табуляції функції. При використанні математичних функцій необхідно обов'язково підключити математичну бібліотеку `#include <cmath>`.

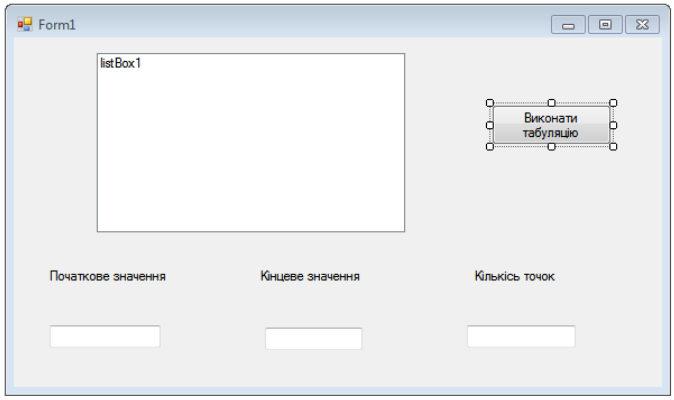


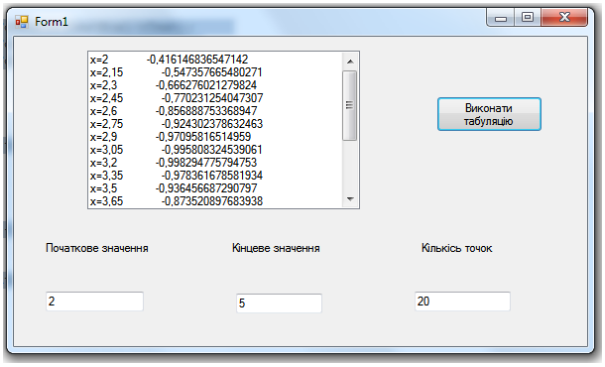
Рис.18. Проект форми, що ілюструє використання ***ListBox***

В тіло функції для кнопки ***button1*** введено наступний код:

```
double a,b,c;
// початок області табуляції
    a=Convert::ToDouble(textBox1->Text);
// кінець області табуляції
    b=Convert::ToDouble(textBox2->Text);
// кількість точок табуляції
    c=Convert::ToDouble(textBox3->Text);
double h;
```

```
double R;
// визначення кроку табуляції
h=(b-a)/c;
for(double t=a; t<=b; t=t+h)
{
// обчислення значення функції, яка табулюється
R=cos(t);
 //Вивід даних в listBox1 із пояснювальним текстом
 listBox1->Items->Add("x="+t      +"                  "+R);
}
```

Відкомпілювавши та запустивши програму отримаємо:



Створення форми з використанням *PictureBox*.

В даному прикладі проілюстровано використання ***PictureBox*** для побудови у формі графіка функції

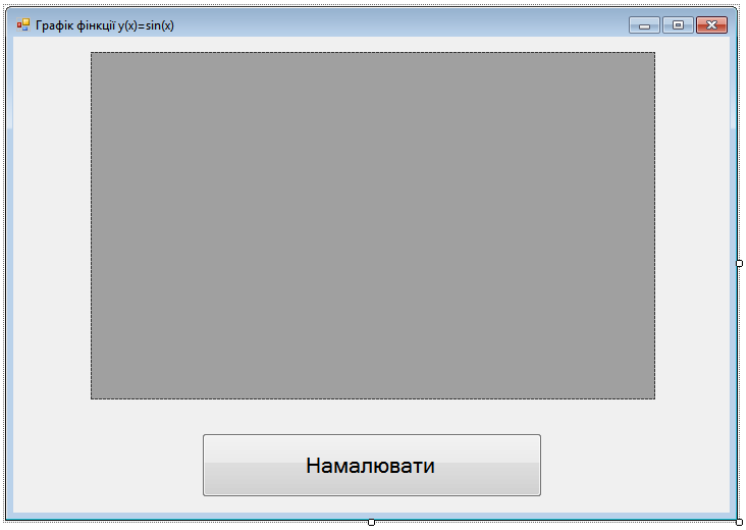
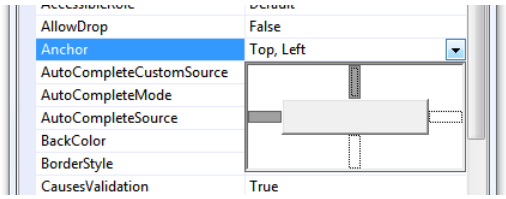
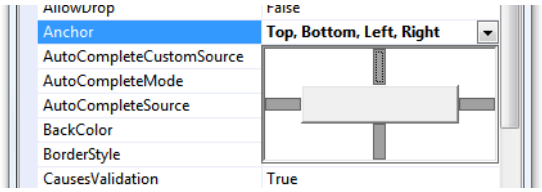


Рис.18. Проект форми для ілюстрації використання **panel**

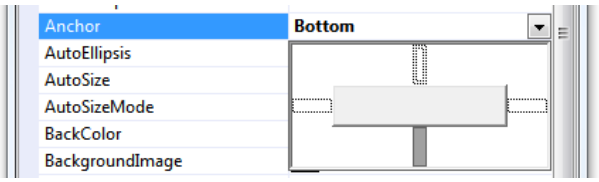
Для прив’язки розміру елементу ***PictureBox*** до форми, розміри якої можуть мінятися користувачем служить властивість ***Achor***:



Як видно, по замовчування прив’язка є до лівої верхньої частини форми. Для пропорційної зміни розміру елементу разом із вікном необхідно виділити праву і нижню частину:



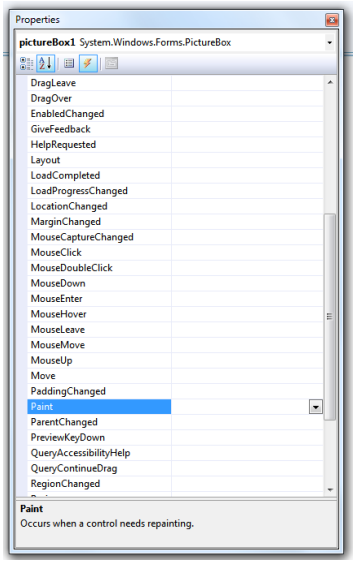
Для елементу ***button1*** задамо наступну властивість ***Achor***:



В тіло функції для кнопки ***button1*** введено наступний код (змінні ***W*** та ***H*** описати як глобальні):

```
/*визначення параметрів вікна, у якому здійснюється
побудова графіка функції*/
     W = PictureBox1->Width;
    H = PictureBox1->Height;
 //оновлення вікна PictureBox1_Paint, якщо помінялися
вхідні значення для побудови графіка функції
     pictureBox1->Refresh();
```

Для добавлення події ***PictureBox1_Paint*** необхідно скористатися панеллю властивостей даного елементу згідно приведеного нижче малюнку:

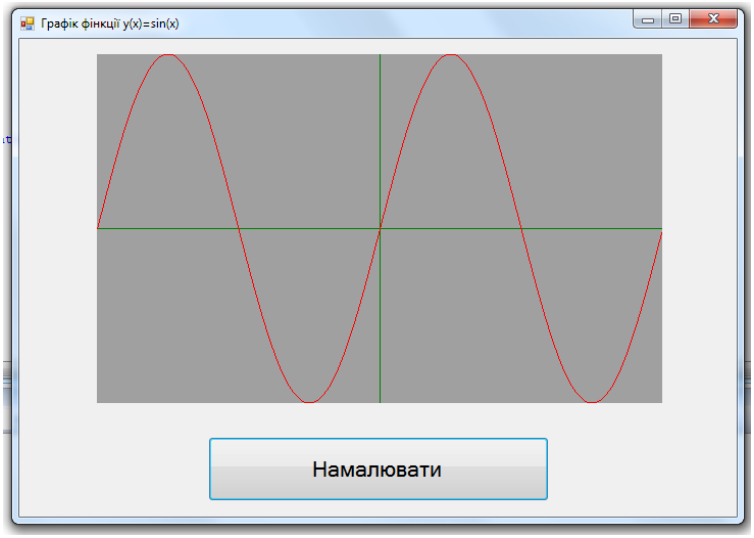


Та два рази клікнути лівою клавішею миші на вибраному пункті меню. Для добавленої події ***PictureBox1_Paint*** введено наступний код:

```
//заадння тексту
String^ Text="Графік фунуції";
//колір тексту
Brush^ Кисть = gcnew SolidBrush(Color::Black);
//шрифт(діє для всього додатку)
Font = gcnew System::Drawing::Font("Times New Roman", 14,
FontStyle::Bold);
//вивід тексту у елемент PictureBox
e->Graphics->DrawString(Text, Font, Кисть, 150, 50);
// Координаты размещения текста
double halfW = W / 2., halfH = H / 2.;
//побудова системи координат
 e->Graphics->DrawLine(System::Drawing::Pens::Green, 0,
halfH, W, halfH);
e->Graphics->DrawLine(System::Drawing::Pens::Green,
halfW, 0, halfW, H);
//початкові дані для побудови графіка функції
int ixPrev = -1, iyPrev = (int)halfH;
// побудова графіка функціх;
 for (int ix = 0; ix < W; ix++)
             {
// перевід значення x в діапазон -1..1
float x = (ix - halfW) / halfW;
// перевід значення x в діапазон  -2*pi..2*pi
x =2*x*(float)Math.PI;
 // обчислення значення функції sin(x)
float y = (float)((sin(x)));
// переводим y із -1..1 в пікселі на формі
int iy = (int)(halfH - y * halfH);
```

```
// Color визначає колір лінії графіка
//графік будується як відрізки, що з'єднують попереднє та
наступне //обчислене значення функції
e->Graphics->DrawLine(System::Drawing::Pens::Red, ixPrev,
iyPrev, ix, iy);
ixPrev = ix;
    iyPrev = iy;
            }
```

Відкомпілювавши та запустивши програму отримаємо (текст на графічній формі не виводився):



Якщо приведений вище код помістити не в тіло функції для кнопки ***button1***, а в тіло функції ***panel1***, то графік функції буде будуватися зразу після відкриття форми без необхідності натиснути кнопку «***Намалювати***» .

Використання горизонтальних та вертикальних полів прокрутки.

Для ілюстрації створимо форму, у якій значення у відповідних текстових полях можна задавати за допомогою вертикальної та горизонтальної полос прокрутки.

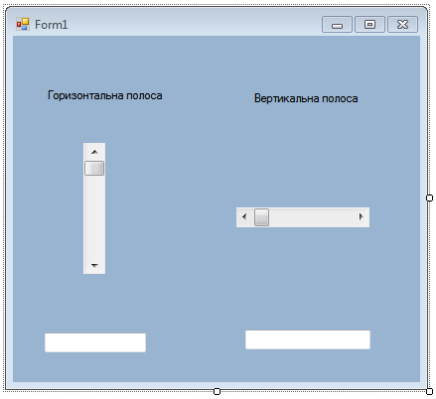


Рис.19. Проект форми для ілюстрації використання **panel**

В тіло функції для вертикальної полоси прокрутки ***vScrollBar*** введемо наступний код:

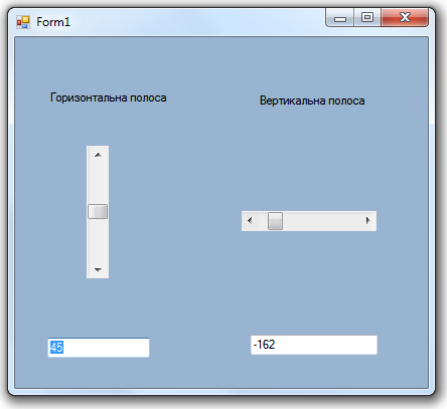
```
int rezult;
rezult=(vScrollBar1->Value);
textBox1->Text = Convert::ToString(rezult);
```

В тіло функції для горизонтальної полоси прокрутки ***hScrollBar*** введено наступний код:

```
int rezult1;
rezult1=(hScrollBar1->Value);
textBox2->Text = Convert::ToString(rezult1);
```

Діапазон зміни значень, що отримуються із полоси прокрутки можна налаштувати за допомогою пунктів меню вікна ***Properties Minimum*** та ***Maximum*** відповідно для ***vScrollBar*** та ***hScrollBar***

Відкомпілювавши та запустивши програму отримаємо:



Значення, що виводяться в полях ***textBox1*** та ***textBox2*** можуть бути задані за допомогою відповідних полос прокрутки

Завдання для індивідуального виконання

Використовуючи елементи управління роботою Windows додатків, описаних у теоретичній частині до лабораторної роботи створити форму для табуляції функції у заданому діапазоні та побудувати її графік.

№ п/п	Функції	Область побудови та крок
1.	$f(x)=5\cos(2x+\pi)$	$[-1,2]$, крок $h=0.25$
2.	$f(x)=\cos^2(x)+2x$	$[-2,2]$, крок $h=0.1$
3.	$f(x)=3x^5+20x^2-20x+3$	$[1,4]$, крок $h=0.2$
4.	$f(x)=\sin(x-0.5)$	$[1,5]$, крок $h=0.4$
5.	$f(x)=\cos(\sqrt{x})$	$[2,5]$, крок $h=0.2$
6.	$f(x)=\left(x+\frac{2}{x}\right)$	$[1,3]$, крок $h=0.1$
7.	$f(x)=\cos\left(x+\frac{1}{x}\right)$	$[1.3,5.1]$, крок $h=0.3$
8.	$f(x)=\frac{x}{x^2-2.5}$	$[3,7]$, крок $h=0.15$
9.	$f(x)=\frac{1}{\sqrt{2x+3}}$	$[-1,1]$, крок $h=0.1$
10.	$f(x)=\sin\left(\frac{x^2}{2}\right)$	$[1,4]$, крок $h=0.2$
11.	$f(x)=e^x+2$	$[-1,1]$, крок $h=0.05$
12.	$f(x)=\cos(x)-10\cdot x+1$	$[-3,3]$, крок $h=0.5$
13.	$f(x)=x^3+x-1$	$[-2,2]$, крок $h=0.2$
14.	$f(x)=x-2x\sin(0.5x)$	$[0,2]$, крок $h=0.1$
15.	$f(x)=x+3\ln(x)$	$[2,8]$, крок $h=0.5$

16.	$f(x)=10x-4\cdot\sin(0.5x)$	$[0,3]$, крок $h=0.15$
17.	$f(x)=\cos(0.5x)+3$	$[1,5]$, крок $h=0.4$
18.	$f(x)=\frac{x}{e^x}$	$[-1,2]$, крок $h=0.2$
19.	$f(x)=e^{\frac{x}{2}}$	$[-2,2]$, крок $h=0.25$
20.	$f(x)=\cos(x^2+2)-1$	$[3,10]$, крок $h=0.5$
21.	$f(x)=x\cos(5x)$	$[2,4]$, крок $h=0.1$
22.	$f(x)=\sin(3x)$	$[0,2]$, крок $h=0.2$
23.	$f(x)=x-\sqrt{x}$	$[0,1]$, крок $h=0.05$
24.	$f(x)=-x^3+0.3x^2-3.1x-1$	$[0,2]$, крок $h=0.15$
25.	$f(x)=2-\sin(2x)$	$[-3,3]$, крок $h=0.5$
26.	$f(x)=\frac{\sin(3x)}{x}$	$[0.1,4.2]$, крок $h=0.2$
27.	$f(x)=\sqrt{3-x^2}$	$[-1,1]$, крок $h=0.1$
28.	$f(x)=\sin(7x+3)$	$[-2,1]$, крок $h=0.25$
29.	$f(x)=\frac{x}{\cos^2(x)}$	$[-0.5,0.5]$, крок $h=0.05$
30.	$f(x)=2x+3\cos(x)$	$[-1.5,1.5]$, крок $h=0.15$

Лабораторна робота №4

Тема робота: Створення багатовіконних додатків із можливістю вводу-виводу даних у файл.

Мета роботи: формування навичок використання різних елементів управління у програмах **Windows Forms**

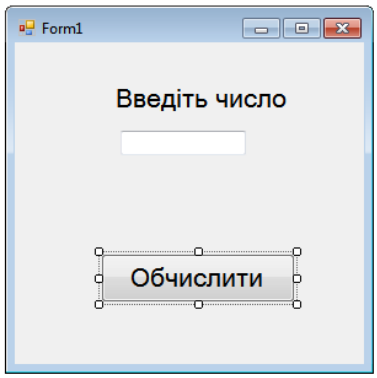
Для виконання роботи необхідно знати:

- створення двох звязаних форм;
- форматний вивід даних;
- створення меню за допомогою MenuStrip;
- призначення та можливості елементаму ***openFiledialog***;
- призначення та можливості елементаму ***saveFiledialog***;
- використання параметра ***Achor*** для налаштування роботи елементу ***textBox***

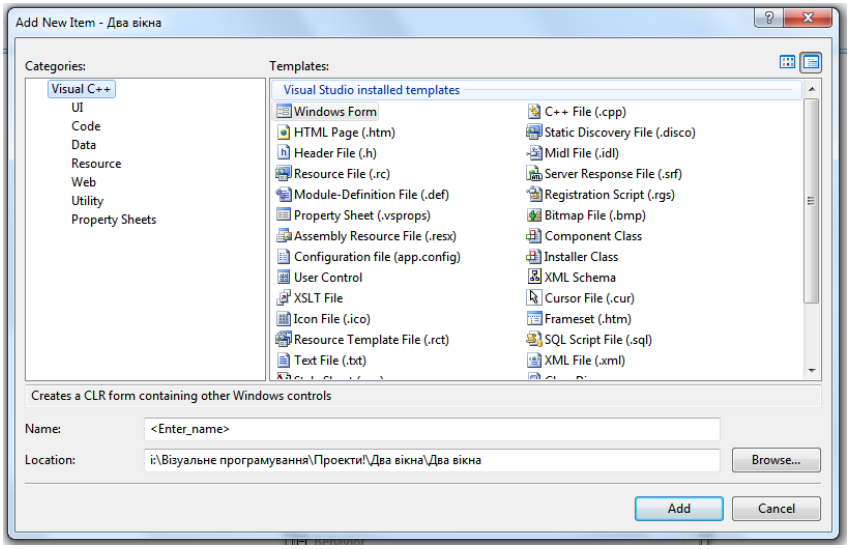
Теоретичні відомості

Створення двох звязаних форм.

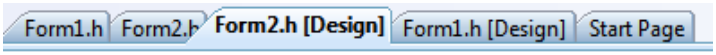
Вихідна (батьківська) форма може містити кілька додаткових (дочірніх) вікон. Проілюструємо порядок створення таких двох зв'язаних форм на наступному прикладі: перша форма містить два елементи управління - ***Toolbox*** та ***Button*.** Після вводу числа у текстовому полі та натискання кнопки результат піднесення числа до квадрату відкривається в дочірній формі. Для прикладу дизайн батьківської форми наступний:



Другу форма створюємо в цьому – проекті наступним чином: в пунуті меню розділу ***"Project"*** вибираємо ***"AddNewItem…"***.



Після цього вибираємо новий елемент ***" Windows Forms"***, задавши для нього імя – ***Form2***. Після цього на панелі появиться друга форма конструктора:



На другій формі розміщеємо елемент - ***Toolbox***:

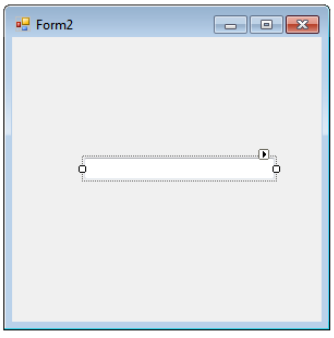
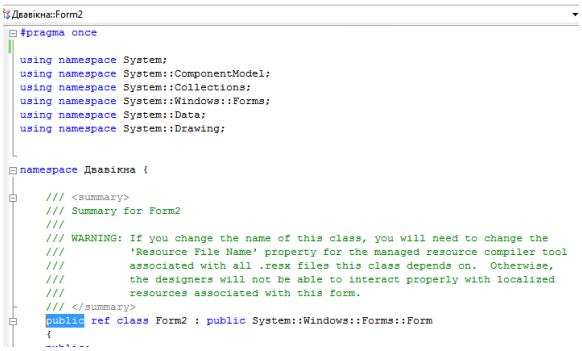
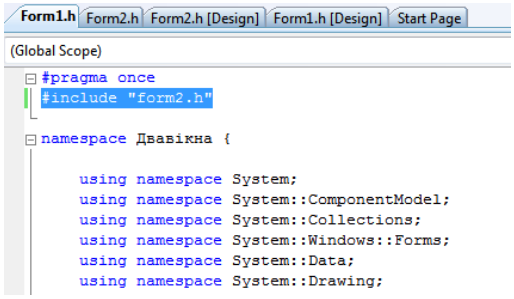


Рис.20. Приклад дизайну дочірної форми ***Form2***

Важливим є те, що елемент ***Toolbox1*** одночасно біде знаходитись на обох формах, тому його необхідно оголосити відритим членом класу (***public***) у верху коду форми ***Form2***(виділений фрагмент коду):



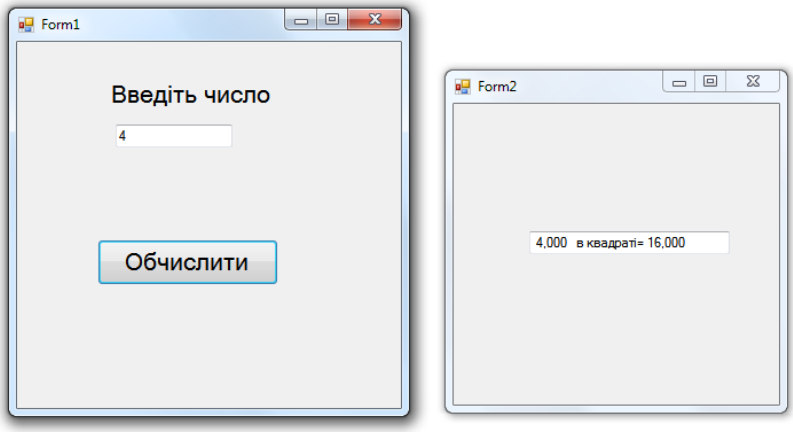
В коді форми ***Form1*** необхідно підключити бібліотеку другої форми ***#include "form2.h"***:



Для обробки події, пов'язаної із ***Button1***, що знаходиться на ***Form1*** необхідно написати наступний код:

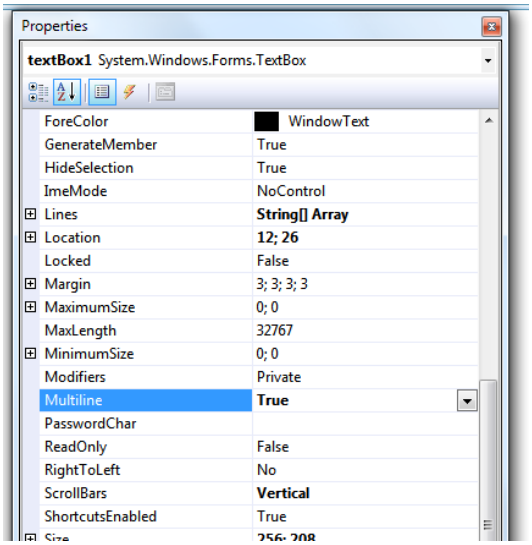
```
double  a, b = 0;
a=Convert::ToDouble(textBox1->Text);
b = a*a;
Form2^ gform2 = gcnew Form2;
//вивід другої форми
gform2->Show();
//форматний вивід числа а та б
gform2->textBox1->Text=String::Format("{0:F3} в квадраті=
{1:F3}",a, b);
```

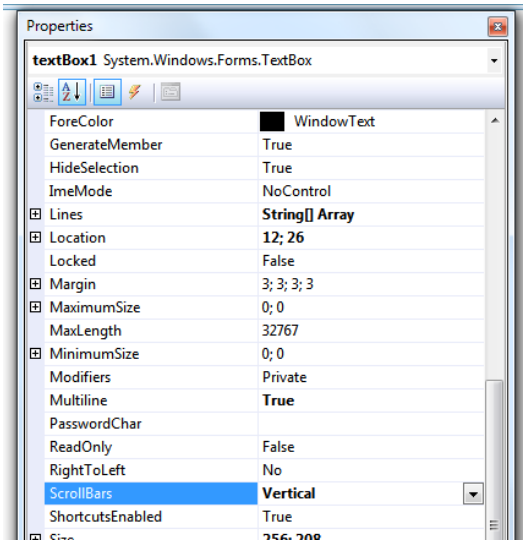
Зверніть увагу на форматний вивід обчисленого та вихідного значення чисел `a` та `b` . Тут `{0:F3}` визначає місце в тексті, де буде виведена перша змінна (`a`) та кількість знаків, що будуть виведені після коми, а `{0:F3}` – друга змінна (`b`) та формат її виводу. Результат роботи відкомпільованого проекту:



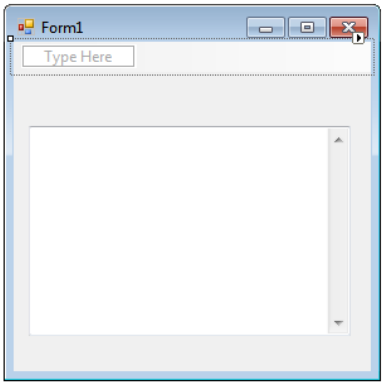
Створення меню за допомогою *MenuStrip*.

Елемент дозволяє випадаючі меню із набором можливих операцій. Для ілюстрації на формі будуть знаходитися елементи ***MenuStrip*** та ***textBox***. Для ***textBox*** задано налаштування:

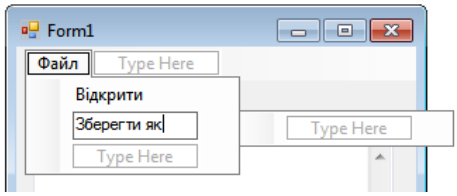




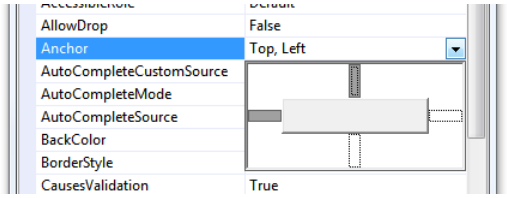
Дизайн форми наступний:



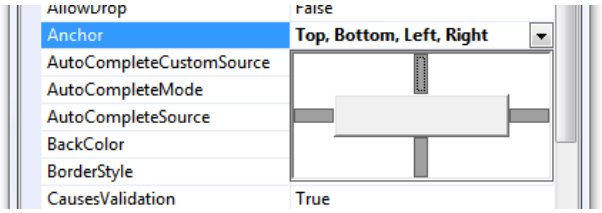
Для елементу ***MenuStrip*** формуємо наступні команди



Для прив'язки розміру елементу ***textBox*** до форми, розміри якої можуть мінятися користувачем служить властивість ***Achor***:

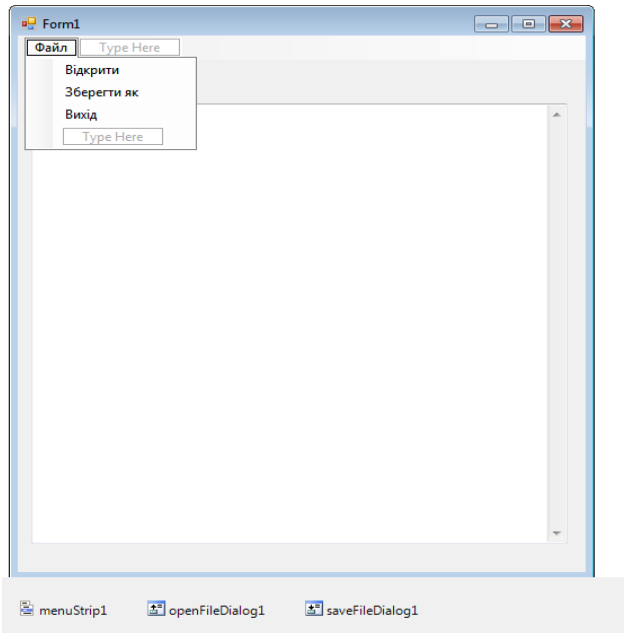


Як видно, по замовчування прив'язка є до лівої верхньої частини форми. Для пропорційної зміни розміру елементу разом із вікном необхідно виділити праву і нижню частину:



Відкриття та запис текстового файлу.

Як продовження до попереднього пункту створимо простий текстовий редактор. Доповнимо попередню форму елементами ***openFiledialog***, ***saveFiledialog*** та добавивши в ***MenuStrip*** пункт меню ***Вихід***:



Для налаштування обробки подій, пов'язаних із вибором відповідних пунктів меню введено наступні фрагменти коду:

- для функції завантаження графічного додатку ***Form1_Load***:

```
    private: System::Void Form1_Load(System::Object^
sender, System::EventArgs^  e) {
//задання заголовка додатку
this->Text = "Текстовый редактор";
//задання імені файлу
openFileDialog1->FileName = "*.txt";
//фільтр для відображення файлів при відкритті файлу
openFileDialog1->Filter = "Текстові файли
(*.txt)|*.txt|All files (*.*)|*.*";
//фільтр для відображення файлів при закритті
saveFileDialog1->Filter = "Текстові файли
(*.txt)|*.txt|All files (*.*)|*.*";
```

}

- для функції закриття графічного додатку ***Form1_Load*** для перевірки, чи зміни в файлі:

```
    //перевірка, чи були зміни у файлі
  if (textBox1->Modified == false) return;
  //Вавід повідомлення, що файл було змінено
   DialogResult=MessageBox::Show("Текст було змінено.
  \nЗберегти зміни?","Простий редактор",
  MessageBoxButtons::OKCancel,
  MessageBoxIcon::Exclamation);
  //Перевірка, яка кнопка була натичнута у MessageBox
    if(Convert::ToString(DialogResult)=="Cancel") return;
  //textBox3->Text = "Скасовано";
   else{
  //Якщо натиснуто "ОК", то викликається функція
  // Save()для зберкження текстового файлу
       {
     Save(); return;
          }
```

- для пункту меню ***Відкрити***

```
if(openFileDialog1->ShowDialog() ==
System::Windows::Forms::DialogResult::OK)
```

```
        {
            System::IO::StreamReader ^ sr = gcnew
//відкриття елемента openFiledialog
      System::IO::StreamReader(openFileDialog1->FileName);
//читання даних із файлу у textBox1
      textBox1->Text = Convert::ToString(sr->ReadToEnd());
//закриття елемента openFiledialog
            sr->Close();
    }
```

- для пункту меню ***Зберегти як:***

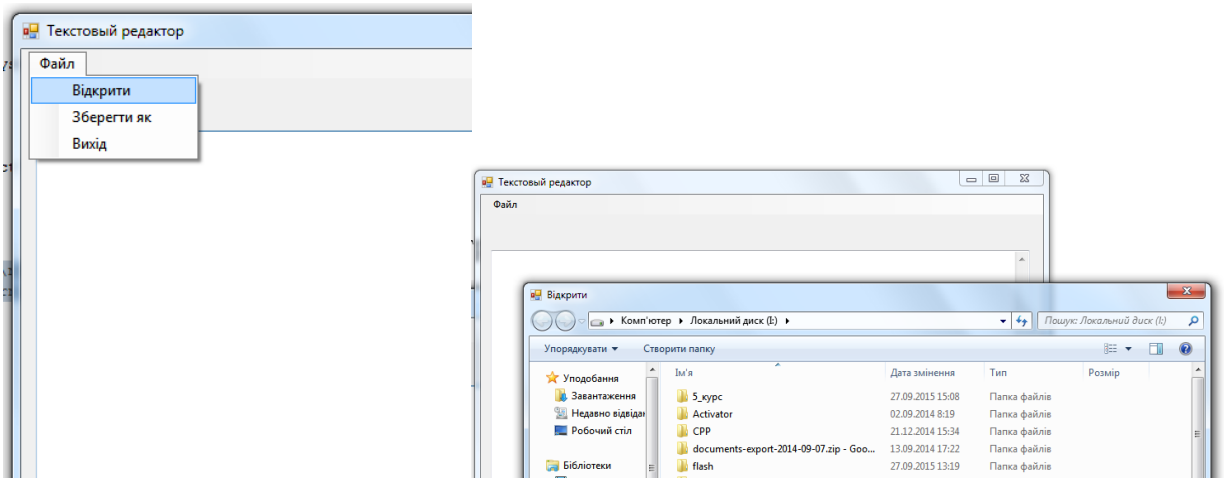
```
//відкриття елемента saveFileDialog
saveFileDialog1->FileName = openFileDialog1->FileName;
if (saveFileDialog1->ShowDialog() ==
Windows::Forms::DialogResult::OK) Save();
}
//функція для зберкження текстового файлу
void Save()
{
try
{
SaveFileDialog ^saveFileDialog1 = gcnew SaveFileDialog();
     //фільтр для відображення файлів при закритті файлу
     saveFileDialog1->Filter = "Текстові файли|*.txt"  ;
     saveFileDialog1->FilterIndex = 2 ;
     saveFileDialog1->RestoreDirectory = true ;
      if(saveFileDialog1->ShowDialog() ==
System::Windows::Forms::DialogResult::OK)
      {
//вивід даних у файл
   IO::File::WriteAllText(saveFileDialog1-
>FileName,textBox1->Text);
         }
}
catch (Exception^ Ситуация)
{
}
```

- для пункту меню ***Вихід:***

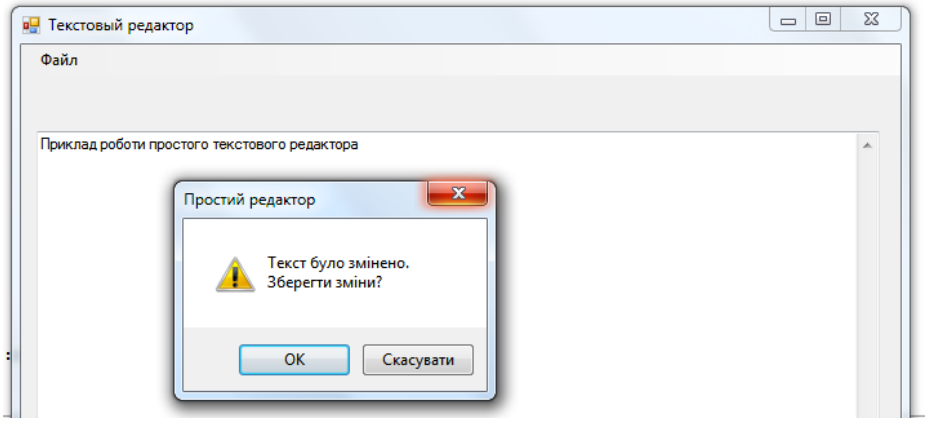
```
     Close();
```

Результати виконання програми:

відкриття файлу:



Спроба закриття документу чи виходу із віконного додатку, коли текстовий документ було редаговано:



Завдання для індивідуального виконання

Використовуючи елементи управління роботою Windows додатків, описаних у теоретичній частині до лабораторної роботи створити форму для роботи із текстовими документами. Дизайн та призначення форми розробити самостійно.

ЛІТЕРАТУРА

1. Шилдт Г. Полный справочник по C++/ Герберт Шилдт [4-е издание, Пер. с англ.]. – М. : Издательский дом «Вильямс», 2006. - 800 с.
2. Водовозов, В. М. Конструирование приложений для Windows : учебное пособие / В. М. Водовозов, А. К. Пожидаев. – СПб. : Изд-во СПбГЭТУ «ЛЭТИ», 2004. – с. 412.
3. Павловская, Т. А. C/C++. Программирование на языке высокого уровня : учебник для вузов / Т. А. Павловская. – СПб. : Питер, 2010. – с. 467.
4. Павловская, Т. А. C/C++. Структурное и объектно- ориентированное программирование : практикум / Т. А. Павловская, Ю. А. Щупак. – СПб. : Питер, 2010. – 329 с.
5. Пахомов, Б. И. C/C++ и MS Visual C++ 2010 для начинающих. – СПб. : БХВ-Петербург, 2011. – 736 с.
6. Понамарев, В. А. Программирование на C++/C# в Visual Studio. NET / В. А. Пономарев. – СПб. : БХВ-Петербург, 2004. – 562 с.
7. Хортон, Айвор Visual C++ 2005: базовый курс : пер. с англ. / Хортон Айвор. – М. : ООО «И. Д. Вильямс», 2007. – 1152 с.
8. Гордон Хогенсон "C++/CLI: язык Visual C++ для среды .NET". Издательство: Вильямс, 2007 С. 464.
9. Б. Пахомов C/C++ и MS Visual C++ 2008 для начинающих СПб.: БХВ – Петербург.-2009.- 624с.

ЗМІСТ