

## Лаб. 1. Віконний режим у MS VS2012 Windows Form Application

**Мета:** Засвоїти основні принципи побудови інтерфейсу користувача за допомогою Windows Form Application у середовищі Microsoft Visual Studio 2012.

**Завдання:** Створити віконний проєкт для вирішення модельної задачі – розв’язання квадратного рівняння за допомогою елементів WF у середовищі MS VS2012. Під час проєктування передбачити можливості неправильної роботи користувача з розробленим віконним інтерфейсом.

### 1. Алгоритм розв’язку квадратного рівняння

Із шкільного курсу алгебри відомо, що квадратне рівняння  $ax^2 + bx + c = 0$  завжди має два корені. Залежно від значення дискримінанта  $D = b^2 - 4ac$ , можливі три випадки:

- $D < 0$  – обидва корені є комплексними (уявними);
- $D = 0$  – корені є дійсні і рівні  $x = -b / (2a)$ ;
- $D > 0$  – обидва корені є дійсні, різні і визначаються за формулами:

$$x_1 = (-b + \sqrt{D}) / (2a); \quad x_2 = (-b - \sqrt{D}) / (2a).$$

Подамо алгоритм розв’язання квадратного рівняння у двох варіантах: словесно-формульному (описовому) і графічному (як блок-схему).

#### Описова форма представлення алгоритму

Алгоритм розв’язання цієї задачі полягає у виконанні послідовності кроків:

1. Вводимо у пам’ять комп’ютера вхідні дані – коефіцієнти рівняння:  $a$ ,  $b$ ,  $c$ . Для цього попередньо забезпечуємо (шляхом відповідного опису) виокремлення компілятором необхідного місця в оперативній пам’яті. Передбачаємо також виокремлення місця для запису дискримінанта рівняння  $D$  і змінних  $x$ ,  $x_1$  та  $x_2$ .

2. Якщо водночас  $a = 0$  і  $b = 0$  – викликаємо повідомлення: “Рівняння не сумісне” і закінчуємо роботу програми.

3. Якщо  $a = 0$ , але  $b \neq 0$ , обчислюємо і виводимо у відповідне поле значення єдиного кореня:  $x = -c/b$  і закінчуємо обрахунок.

4. У іншому випадку обчислюємо дискримінант:  $D = b^2 - 4ac$ .

5. Якщо  $D < 0$  – викликаємо повідомлення: “Дійсних коренів немає” і закінчуємо роботу програми.

6. У іншому випадку обчислюємо значення  $x = -b/(2a)$ .

7. Якщо  $D = 0$ , викликаємо повідомлення: “Обидва корені рівні  $x$ ”, виводимо значення  $x$  і завершуємо роботу програми.

8. В іншому випадку обчислюємо значення  $x_2 = \sqrt{D}/(2a)$ . Далі послідовно обчислюємо  $x_1 = x_2 + x$  та  $x_2 = x - x_2$ , друкуємо значення коренів  $x_1$  і  $x_2$  і завершуємо роботу програми.

## Графічна форма представлення алгоритму

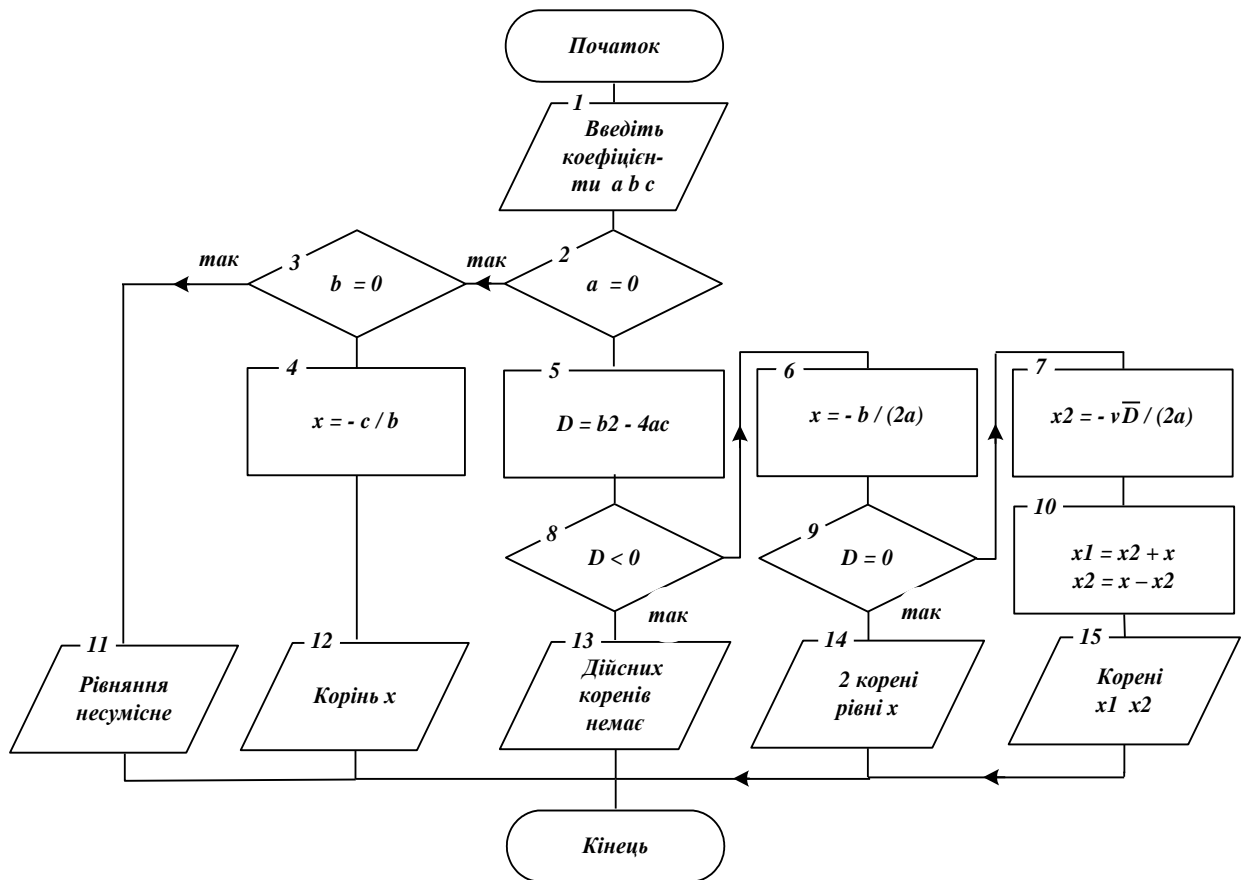
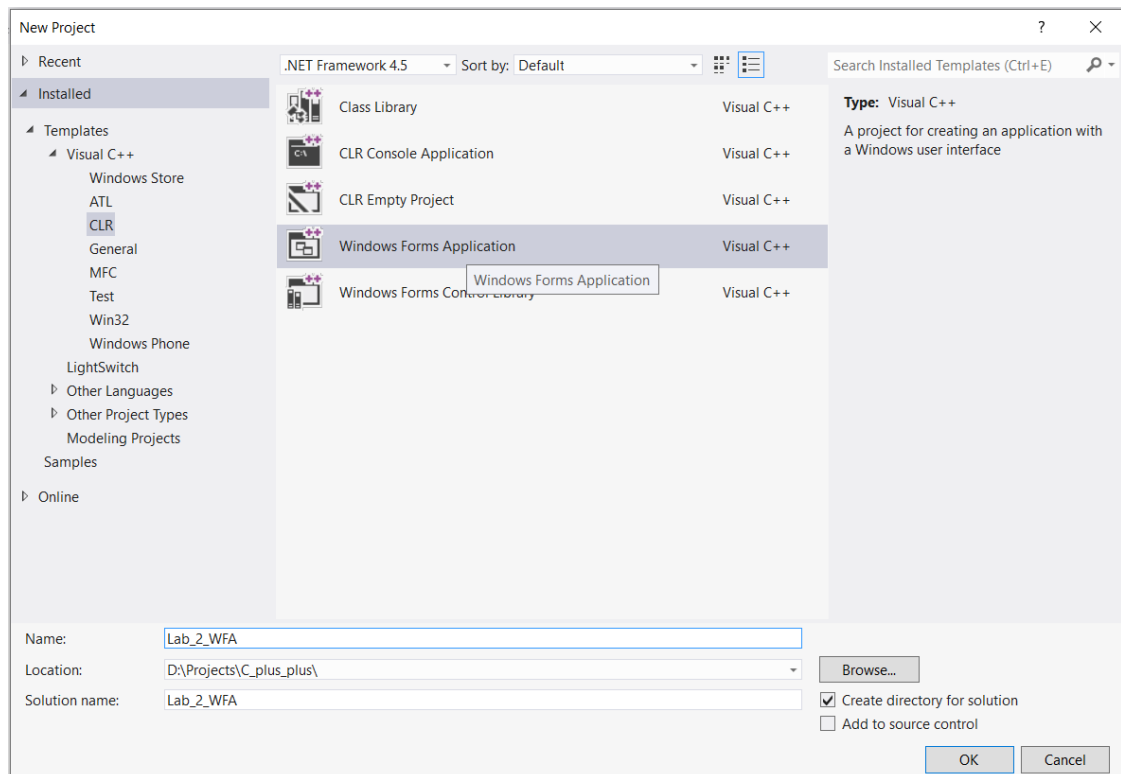


Рис. 1. Блок-схема алгоритму розв'язання рівняння  $ax^2 + bx + c = 0$

## 2. Побудова віконного інтерфейсу користувача.

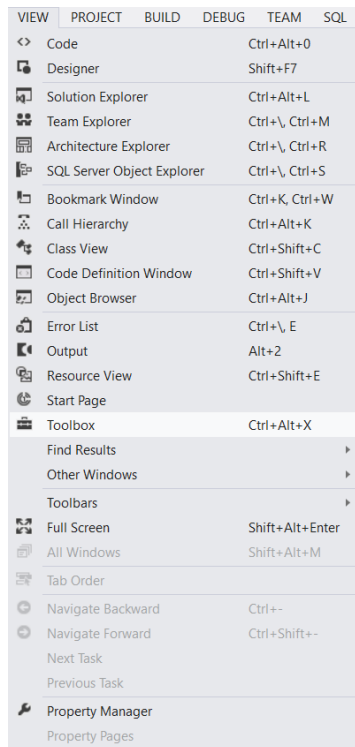
Ознайомимося з роботою у **віконного режиму** середовища Visual C++ 2012. Для цього запусимо Visual Studio, створивши попередньо новий каталог для проєкту, наприклад **Lab\_2\_WFA**.

Одержимо вікно, у якому оберемо **Windows Forms Application**, налаштуємось на новостворений каталог і натиснемо **ОК**.

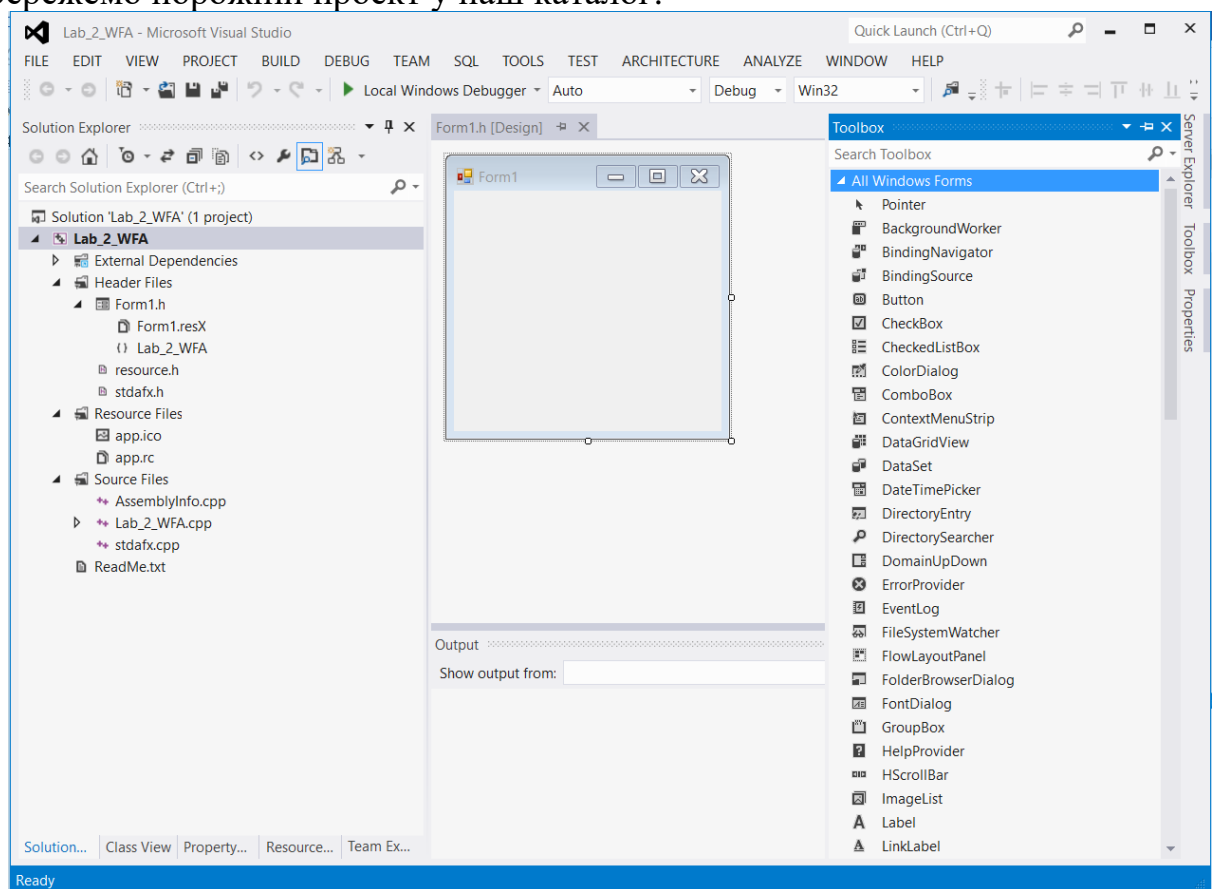


Одержимо можливість розробляти у вікні форми візуальним способом інтерфейс нашої майбутньої програми, використовуючи візуальні засоби з палітри Toolbox. Нижче представлено типовий вигляд віконного середовища Visual C++ 2012. Слід мати на увазі, що цей вигляд можна налаштовувати і тому у Вашому варіанті налаштування екран може мати дещо інший вигляд.

Якщо вікна Toolbox або якогось іншого вікна середовища Visual C++ 2012 не видно, слід скористатись текстовим меню VIEW і обрати там необхідне для відображення вікно.



Збережемо порожній проєкт у наш каталог.



У каталозі збереженого проєкту можна бачити наступний набір файлів, згенерований середовищем:

[Debug]		<Папка>
[Lab_2_WFA]		<Папка>
Lab_2_WFA	opensdf	0
Lab_2_WFA	sdf	3 080 192
Lab_2_WFA	sln	894
Lab_2_WFA.v11	suo	10 752

У каталозі **Lab\_2\_WFA** побачимо такий набір файлів:

[Debug]	
app	ico
app	rc
AssemblyInfo	cpp
Form1	h
Form1	resX
Lab_2_WFA	cpp
Lab_2_WFA	vcxproj
Lab_2_WFA.vcxpr..	filters
ReadMe	txt
resource	h
stdafx	cpp
stdafx	h

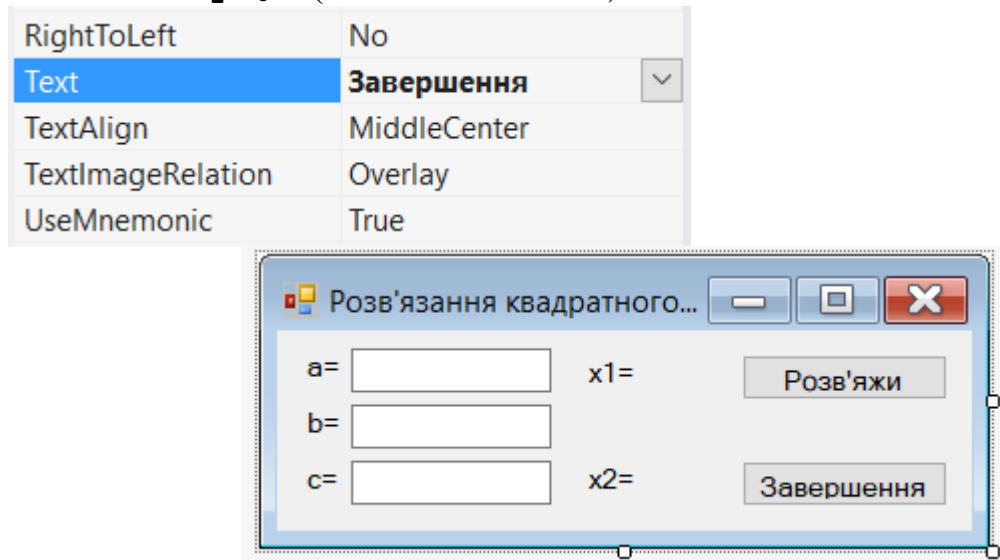
Сформуємо такий візуальний інтерфейс нашої майбутньої програми.

У властивостях форми змінимо поле **Text** на **Розв'язання квадратного рівняння**.


Font	Microsoft Sans Serif; 7,8p
ForeColor	ControlText
FormBorderStyle	Sizable
RightToLeft	No
RightToLeftLayout	False
<b>Text</b>	<b>Розв'язання квадратно</b>
UseWaitCursor	False

Коефіцієнти квадратного рівняння доцільно вводити за допомогою засобу **TextBox**. Помістивши їх на поле форми, почергово активізуємо кожен із них, правою кнопкою миші відкриваємо вікно властивостей (**Properties**) і змінюємо властивість **Name** кожного елементу на зручні для розробки. Для

відображення інформації зручно мітки **Label**. Додамо їх до дизайну нашого вікна перейменуємо їхні властивості **Name** на зручні. Властивості **Text** замінимо на: **a=**, **b=**, **c=**. Для виведення результатів використаємо також 2 мітки **Label**, змінивши їхні властивості **Name** та **Text** на відповідні до призначення. Також на поле форми помістимо два керуючі елементи **Button** із підписами **Розв'яжи** і **Завершуй** (властивості **Text**).



Готовий проект записуємо на диск кнопкою .

Запускаємо проект на виконання кнопкою .

### Використання керуючих елементів.

Поставимо курсор мишки на кнопку “**Завершення**” і двічі клацнемо лівою клавішею. Відкриється вікно для набору коду програми, у якому побачимо автоматично сформований заголовок процедури:

```
private: System::Void btnClose_Click(System::Object^ sender, System::EventArgs^ e) {
    Close(); // Вставляємо оператор Close(), який здійснює вихід з програми
}
```

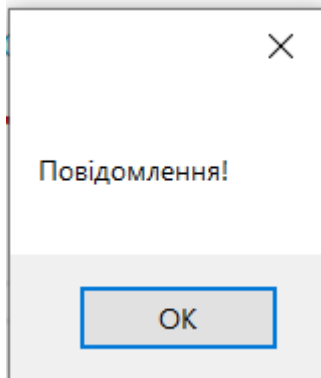
Повернемося назад на форму, відкривши мишкою закладку **Form1.h[Design]\*** та клацнемо два рази мишкою на кнопці “**Розв'яжи**”. У згенерованій процедурі треба реалізувати алгоритм розв'язку квадратного рівняння.

## Створення вікон з повідомленнями

У бібліотеці `System.Windows.Forms.dll` існує багато різноманітних вікон із повідомленнями. Для нашого проєкту використаємо найпростіший варіант. Для цього використаємо код наступного вигляду:

```
MessageBox::Show("Повідомлення!");
```

Де замість “Розв’яжи” може бути інша стрічка. Ми отримаємо наступне повідомлення при запуску:



Як бачимо, воно має мінімальну кількість елементів.

Більше про інформації за посиланням

<https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.messagebox?view=netframework-4.8>

## Опрацювання помилок

Щоб перетворити текст, введений у поле **TextBox** у число можна використати наступний стандартний метод

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

Але користувач у це поле може ввести дані, які неможливо перетворити у число. Тоді виникне помилка та програма аварійно завершиться. Щоб уникнути негараздів, треба передбачити такі випадки.

Для опрацювання помилок у мові C++ визначена наступна конструкція: Try-catch.

```
try
{
    a = Convert::ToDouble(tbVarA -> Text);
}
catch (Exception^ e)
{
    Console::WriteLine( "Exception Handler: {0}", e );
}
```

Ключове слово **try** передує блоку спроби.

Якщо спроба була невдалою, виконується блок **catch**, у якому програміст опрацьовує помилки **e**.

Докладніше про помилки можна подивитись за посиланням:

<https://docs.microsoft.com/en-us/dotnet/api/system.formatexception?view=netframework-4.8>

Додаткове завдання

1. Використовуючи властивості форми та елементів інтерфейсу створити власний дизайн вікна (змінити колір, шрифти, тощо).
2. Замість надпису гудзиків підставити піктограми (рисунки).