ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ О. ГОНЧАРА

Факультет прикладної математики

Кафедра обчислювальної математики та математичної кібернетики

ЗВІТ
ПРО ПРОХОДЖЕННЯ НАВЧАЛЬНОЇ ПРАКТИКИ: КОМП'ЮТЕРНО-ТЕХНОЛОГІЧНОЇ

Студента 2 курсу, групи ПА-22-1

Спеціальності

113 Прикладна математика

____Радченко Дмитро_____

(ім'я ПРІЗВИЩЕ)

Керівник _____асистент_____

_____Наталія БАЛЕЙКО___

Кількість балів_____

Національна шкала _____

Оцінка ECTS _____

Члени комісії: _____ Наталія КОЗАКОВА

(підпис) (прізвище та ініціали)

_____ Наталія БАЛЕЙКО

(підпис) (прізвище та ініціали)

_____ Олексій МАГАС

(підпис) (прізвище та ініціали)

Дніпро

2024

ЗМІСТ

ВСТУП

Наприкінці другого курсу навчання студентам була надана можливість закріпити отримані знання та поглибити їх, виконавши завдання з навчальної комп’ютерно-технологічної практики.

Метою виконавчої практики було ознайомлення з візуальним програмуванням. Завдання роботи складалося з наступних пунктів:

1. Вивчити базові можливості візуального програмування.
2. Вивчити базові можливості програмування графіки.
3. Зобразити блок-схему алгоритму для виведення на екран графіка функції.
4. Написати програму для виведення на екран графіка функції, а також програму для підготовки даних.

Завдання було виконано в IDE (Integrated Development Environment) Інтегрованій Середі Розробки від компанії Microsoft: Visual Studio 2019. Для його реалізації були використанні мови програмування C++ та C#.

Під час навчальної практики дотримувалися всіх вимог щодо охорони праці, згідно з чинним законодавством та правил, які забезпечують безпеку працівників.

ОСНОВНА ЧАСТИНА

Постановка задачі

Розробити програму, яка буде обчислювати значення для функції, що задана в індивідуальному варіанті. Після обчислення результат виводити у текстовий файл. Програму слід розробити використовуючи мову програмування C++ у середовищі розробки Visual Studio.

Варіант 17

$$F(x)=\begin{cases}\sqrt{e^{\sin^3 x}}+4\ln 5x+\dfrac{1}{12}, & 1<x\le 3;\\ x^{30}, & -2<x<-1;\\ x^{-10}, & \textit{в іншому випадку.}\end{cases}$$

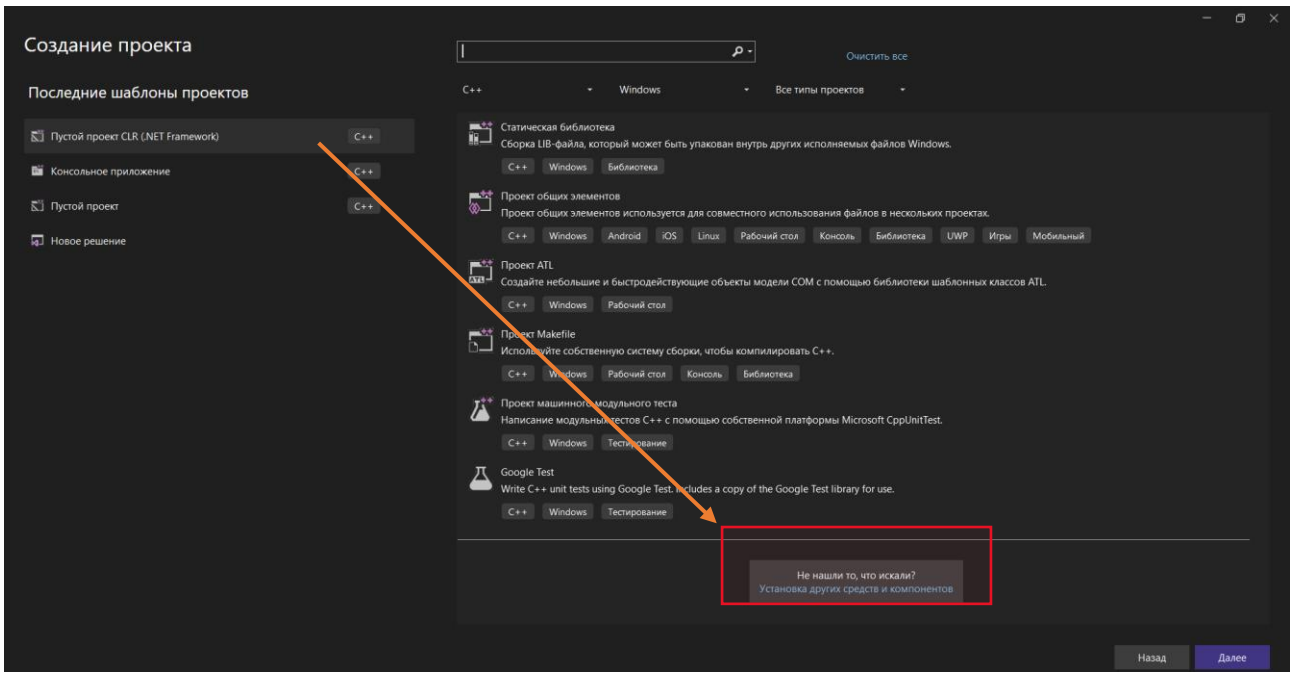
Рисунок 1 - Функція індивідуального варіанту

1 Хід розв’язання

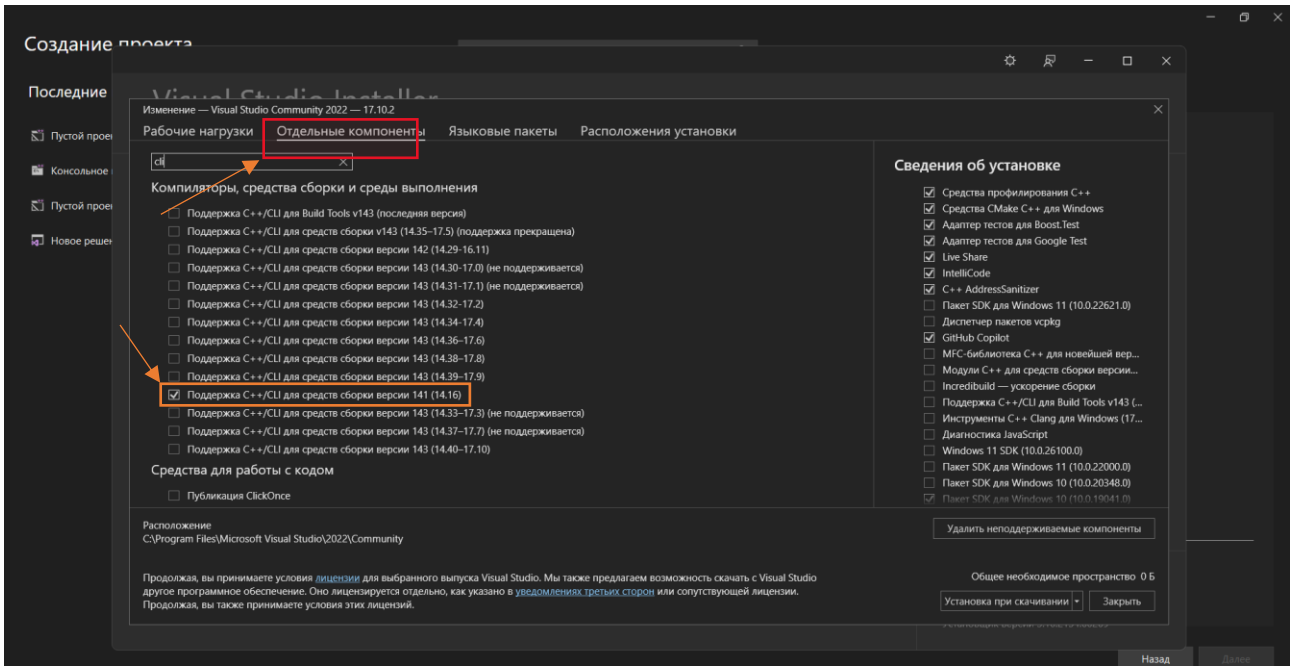
Для роз’вязання поставленої задачі будемо використовувати програму Microsoft VS із встановленним пакетом WinForms для створення візуалізації. Функцію із індивідуального варіанту буде написано мовою c++.

Установка WinForms

Обираємо пункт «створити проєкт» далі переходимо до розділу «встановлення інших середовищ і компонентів»



Знаходимо меню «окремі компоненти». У пошуку прописуємо «cli/c++» та обираємо:



Натискаємо кнопку «Змінити» та чекаємо на установку. Готово! Можна починати створювати застосунок!

Створення візуальної частини проєкту.

Створюємо новий проєкт із розширенням CLR. Переходимо у cpp файл проєкту і пишемо там наступний код

```
#include "MyForm.h"
#include "formula.cpp" // модуль із розробленою функцією з варіанту

using namespace System;
using namespace System::Windows::Forms;

[STAThreadAttribute]

int main(array<String^>^ args) { // ^ - автоматичне очищення пам'яті

    Application::SetCompatibleTextRenderingDefault(false); // коректна обробка тексту
    Application::EnableVisualStyles();// підключаємо різні візуальні стилі
    graph::MyForm form; // звертаємось до простору імен (graph) а потім до класу (MyForm) та створюємо о
                        //та створюємо об'єкт (form)
    Application::Run(% form); // об'єкт (form) передаємо за ссилкою (%)
}
```

Цей код нам потрібен для створення об’єкту типу My Form (My Form у свою чергу є класом, у якому будуть зберігатись основні об’єкти та властивості). Далі переходимо до вікна із назвою «Конструктор». У ньому будемо створювати основні елементи інтерфейсу, які буде бачити користувач, це:

Menu – дозволяє додавати різні функції як список. У нашому меню ми створимо дві групи «Menu» де будуть знаходитися функції «build», побудова графіку, та «clear», для його очищення, «Exit» - вихід із програми

Group Box – вони слугують для зручного розділення інтерфейсу форму

Text Box – потрібні для заповнення формули. У нашому випадку користувач буде вводити проміжок для побудови графіка та крок.

Label – написи. Вони будуть допомагати користувачеві правильно ввести усі необхідні дані.

Chart – вікно з графіком. На ньому будуть відбуватися зміни графіку, залежно від значень, які увів користувач.

Програмування форми

Тепер маємо запрограмувати усі необхідні компоненти, з якими буде працювати користувач.

Для того, щоб перейти до об’єкту інтерфейса класу My Form маємо натуснути на нього лівою кнопкою миші двічі.

Компоненти, які ми будемо запрограмовувати це текст бокси, меню (з усіма його функціями), вихід, перевірку на правильність введених параметрів.

Але спочатку у файлі «My Form.h» маємо дещо змінити, а саме:

Маємо прибрати фігурні дужки із обведених на скріншоті методів(це потрібно для того, аби у файлі з розширенням cpp ми могли записати своє рішення цих методів);

Додамо метод під назвою «DefaultParams()». Цей метод буде перевіряти, чи усе ввів користувач коректно.

```
private: System::Void menuStrip2_ItemClicked(System::Object^ sender, System::Windows::Forms::ToolStripItemClickedEventArgs^ e) {
}
private: System::Void builtToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void clearToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void menuToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e) {


}
private: System::Void exitToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e);
private: System::Void label5_Click(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void label4_Click(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void textBox_A_TextChanged(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void textBox_B_TextChanged(System::Object^ sender, System::EventArgs^ e) {
}
    private: void DefaultParams();
};
```

Реалізація цих методів буде знаходитись у файлі «My Form.cpp»

Почнемо з методу builtToolStripMenuItem_Click(). Цей метод оброблює кнопку «побудувати» у пункті «меню».

Спочатку перевіряємо, чи ввів користувач деякі із параметрів. Якщо хоча б один параметр не є введеним, тоді програма викличе метод «встановити параметр за замовченням». Якщо ж усі параметри введені правильно, тоді програма присвоїть змінним h a b значення із текст боксів. Також використовємо функцію «Коніерувати» аби програма підрахувла значення функції правильно.

```
if (textBox_h->Text == "" || textBox_A->Text == "" || textBox_B->Text == "") {// перевіряємо, чи задан хоч один з переметрів
    MessageBox::Show("Parameters set by default!", "Warning!");// якщо ні, тоді попереджаємо про це користувача
    DefaultParams();//виставляємо параметри за замовченням
}
else {
    h = Convert::ToDouble(textBox_h->Text);
    a = Convert::ToDouble(textBox_A->Text);
    b = Convert::ToDouble(textBox_B->Text);
}
```

Код методу «встановити параметр за замовченням»

```
void graph::MyForm::DefaultParams() //параметри за замовчуванням
{
    a = 1;
    b = 2;
    h = 0.1;
}
```

Після того, як програма присвоїть значення змінним h a b значення із текст боксів

Буде виконаний наступний код

```
this->chart->Series[0]->Points->Clear();
x = a;
while (x <= b) {
    // замінити на функцію із файлу formula.cpp
    //y = Math::Sin(x);
     y = fx(x);
    //
    this->chart->Series[0]->Points->AddXY(x,y);
    x += h;
}
```

Тобто програма намалює графік функції на елементі інтерфейсу «Chart». Функція F(x) буде написана у окремому cpp файлі, який ми потім під'єднаємо до цієї програми.

Наступне, що ми зробимо це запрограмуємо функцію «очистити графік»

Тут все дуже просто. Маємо звернутися до елемента інтерфейсу chart та викликати функцію «очистити». Як наслідок, програма виведе порожній графік.

```
System::Void graph::MyForm::clearToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e)//конопка очичстити графік
{
    this->chart->Series[0]->Points->Clear();// очищаємо графік
}
```

Останнє перед написанням функції F(x) буде програмування кнопки вихід. Аби користувачу було зручніше ми створимо блок-повідомлення, який запистає користувача чи впевнений він, що хоче завершити програму. Якщо кристувач тисне «так» то програма закривається, а якщо «ні» - продовжує роботу. Для цього викликаємо клас MassegeBox метод Show далі MassageBoxButtons::YesNo, тобто встановлюємо кнопки «так» «ні» а далі клас і результат «так» DialogResult::Yes. Для правильного закриття форми скористаємось класом Application та методом Exit().

```
System::Void graph::MyForm::exitToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e)//конопка вихід
{
    if (MessageBox::Show("Exit program?", "Warning!", MessageBoxButtons::YesNo) == Windows::Forms::DialogResult::Yes) {
        Application::Exit();
    }
}
```

Написання функції з індивідуального завдання

У папці зі шляхом C:\Users\PC\OneDrive\Рабочий стол\Практика\програма\graph\graph сьворюємо cpp файл під назвою «formula.cpp» у якому ми напишемо код, відповідний до функції з індивідуального завдання. Для написанн коду використаємо таку бібліотеку як Math.h яка включає у себе усі необзідні нам математичні функції. Далі пишемо код функції

```
#include <math.h>
using namespace std;

double fx(double x) {

        if (1 < x && x <= 3) {
            return sqrt(exp(sin(x) * sin(x) * sin(x))) + 4 * log(5) * x + (1.0 / 12.0);
        }
        else if (-2 < x && x < -1) {
            return pow(x, 30);
        }
        else {
            return pow(x, -10);
        }
}
```

Після написання коду під’єднуємо наш модуль із функцією до модулю «My Form.cpp» за допопмогою дерективи #include і назву модуля беремо у лапки “”

```
#include "MyForm.h"
#include "formula.cpp" // модуль із розробленою функцією з варіанту
```

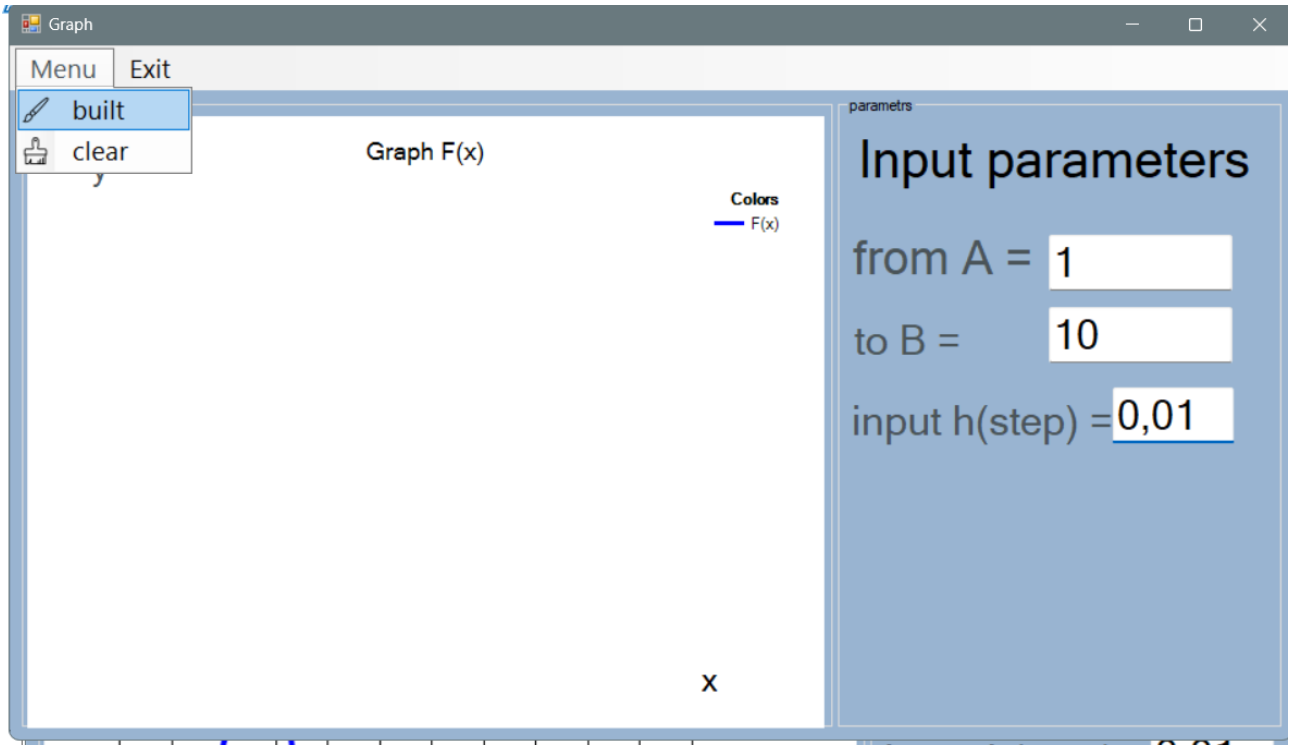
У коді, де відбувається побудова програми, викликаємо функцію fx(x)

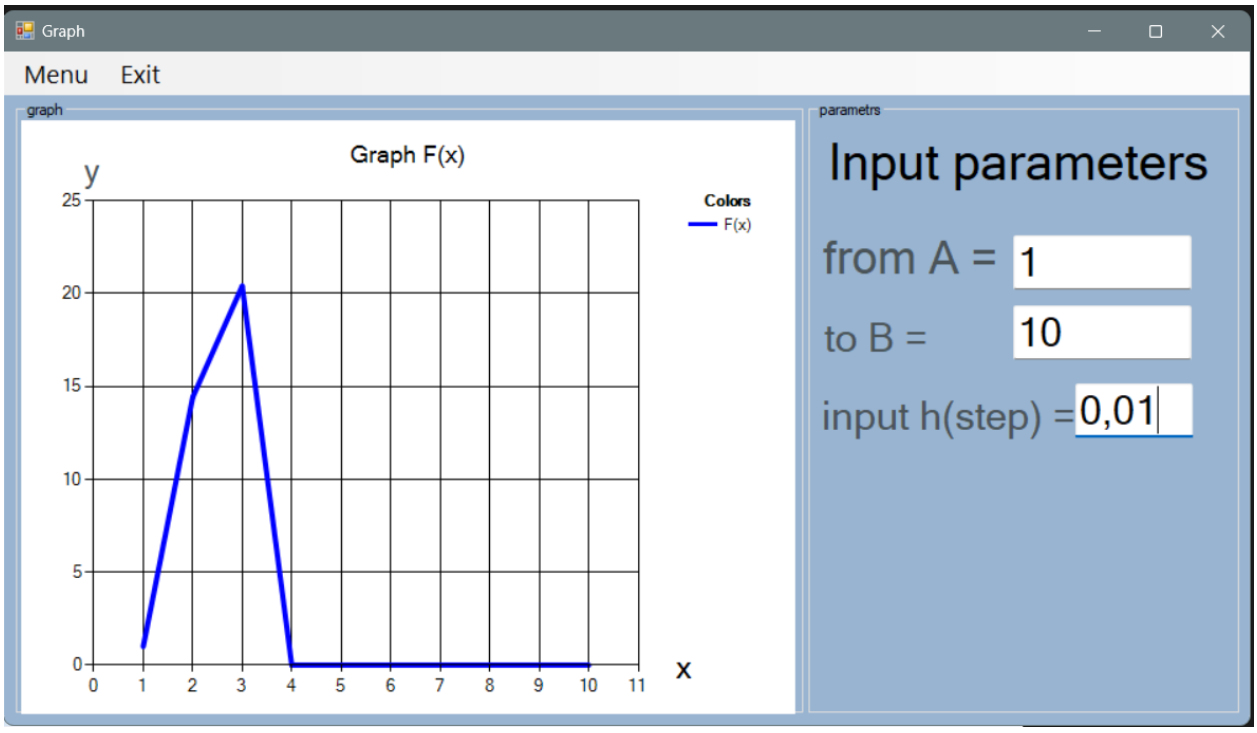
```
System::Void graph::MyForm::builtToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e) // кнопка побудувати г
{
    if (textBox_h->Text == "" || textBox_A->Text == "" || textBox_B->Text == "") {// перевіряємо, чи задан хоч один з перемет
        MessageBox::Show("Parameters set by default!", "Warning!");// якщо ні, тоді попереджаємо про це користувача
        DefaultParams();//виставляємо параметри за замовченням
    }
    else {
        h = Convert::ToDouble(textBox_h->Text);
        a = Convert::ToDouble(textBox_A->Text);
        b = Convert::ToDouble(textBox_B->Text);
    }

    this->chart->Series[0]->Points->Clear();
    x = a;
    while (x <= b) {
        // замінити на функцію із файлу formula.cpp
        //y = Math::Sin(x);
         y = fx(x);
        //
        this->chart->Series[0]->Points->AddXY(x,y);
        x += h;
    }
```

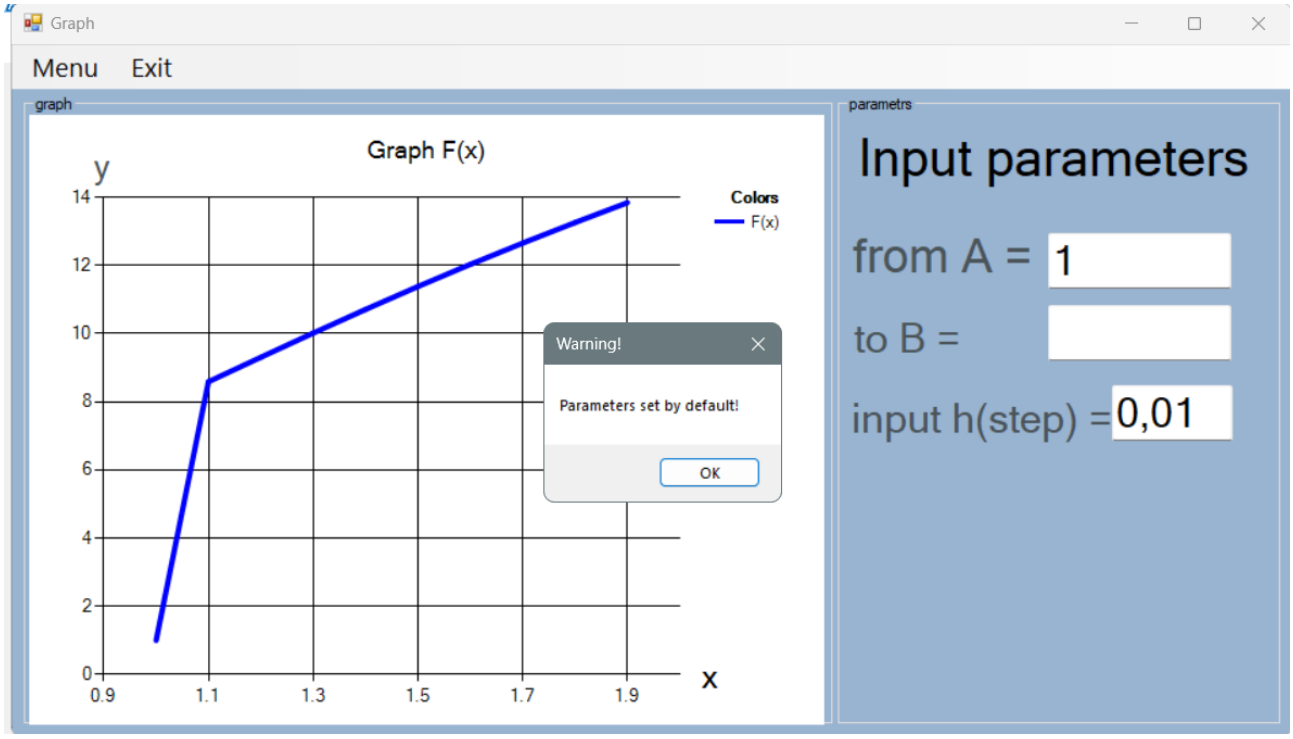
2 Тест програми

Під час запуску програми користувач побачить наступний екран, де користувач може ввести проміжок «від А» і «до Б», це проміжки у яких програма буде будувати графік функції. Окрім вводу параметрів користувач має натиснути кнопку «побудувати» аби програма запрацювала.

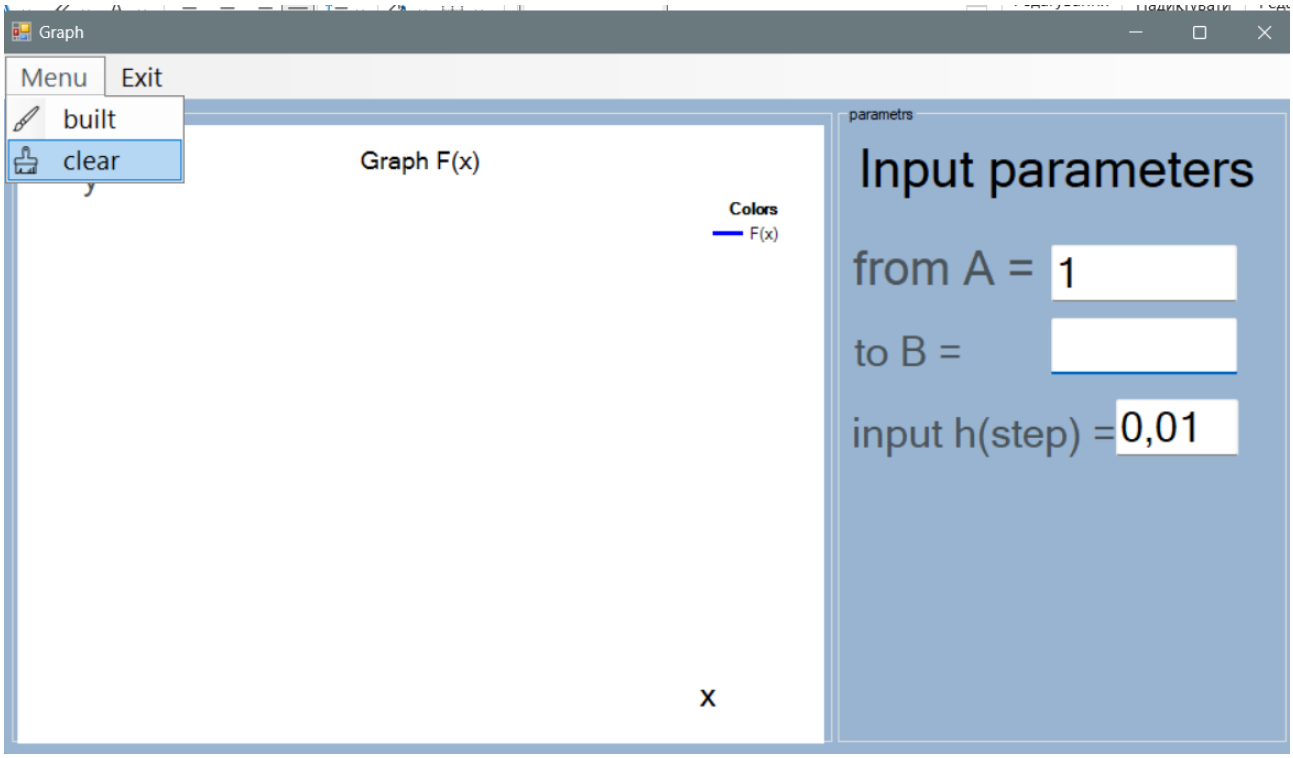




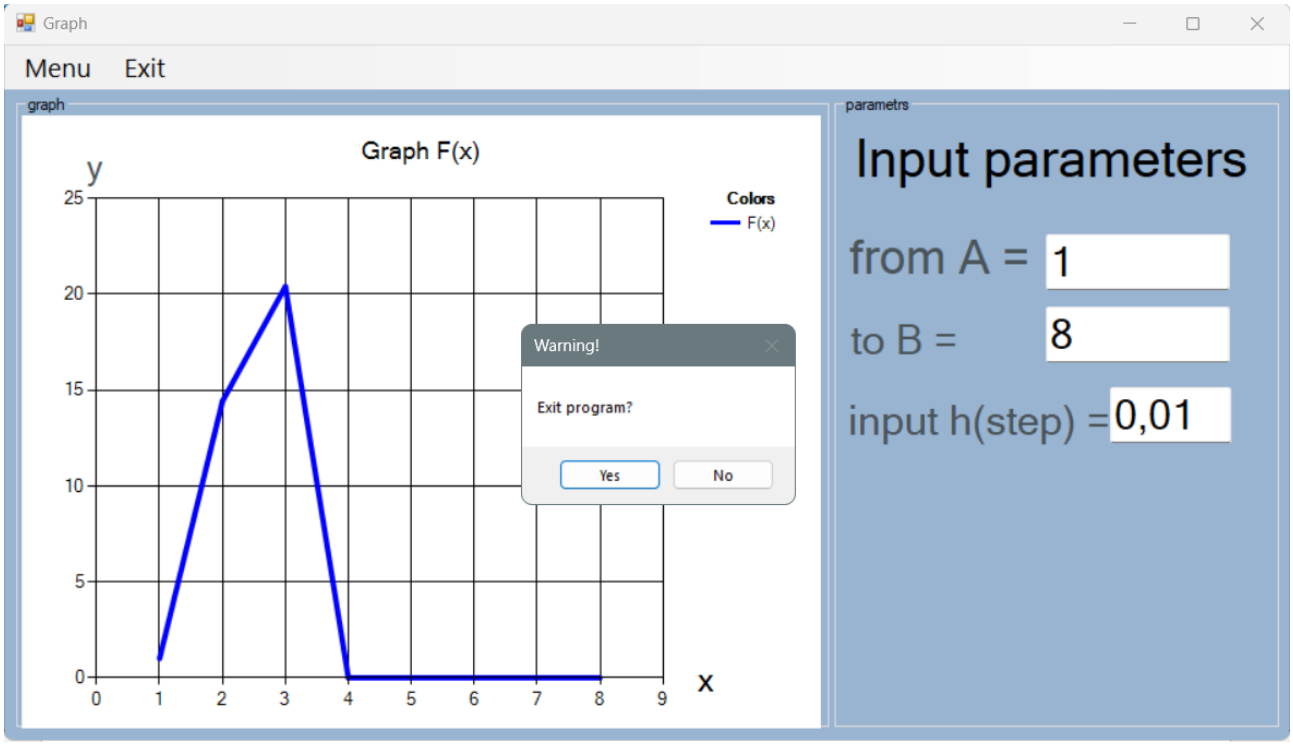
Якщо користувач не введе один або більше параметрів для побудови графіка з’являється вікно, яке повідомляє користувачу про те, що праматри встановлені за замовченням.



У меню користувач також може використати функцію «очичтити» і тоді програма зітре попередній побудований графік.



Коли користувач захоче вийти із програми, то натиснувши кнопку «вийти» він отримає повідомлення, яке запитає чи дійсно він хоче вийти з програми чи ні.



ВИСНОВКИ

За час виконання навчальної практики було розроблено дві програми: перша - для підготовки даних, друга – для виведення графіка на екран та можливості роботи з ним. Таке завдання допомогло набути навичок розробки графічних програмних за стосунків, що є корисним для майбутніх робіт.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Graphics Клас [Електронний ресурс]: Режим доступу: https://learn.microsoft.com/ru-ru/dotnet/api/system.drawing.graph cs?redirectedfrom=MSDN&view=dotnet plat-ext-7.0 – Заголовок з екрану.
2. Панель вкладок TabControl и SplitContainer [Електронний ресурс]: Режим доступу: https://metanit.com/sharp/windowsforms/3.5.php.
3. Окно сообщения MessageBox [Електронний ресурс]: Режим доступу: https://metanit.com/sharp/windowsforms/4.19.php - Заголовок з екрану.
4. Чтение и запись текстовых файлов. StreamReader и StreamWriter[Електронний ресурс]: Режим доступу: C# и .NET | Чтение и запись текстовых файлов. StreamReader и StreamWriter (metanit.com) - Заголовок з екрану.
5. OpenFileDialog Class [Електронний ресурс]: Режим доступу: https://learn.microsoft.com/enus/dotnet/api/system.windows.forms.openf iledialog?view=windowsdesktop-7.0 – Заголовок з екрану.
6. Исключения и обработка исключений [Електронний ресурс]: Режим доступу:https://learn.microsoft.com/ruru/dotnet/csharp/fundamentals/ex ceptions/ - Заголовок з екрану.
7. Преобразование типов и класс Convert [Електронний ресурс]: Режим доступу: https://metanit.com/sharp/tutorial/20.4.php - Заголовок з екрану.
8. Out Parameter With Examples in C# [Електронний ресурс]: Режим доступу: https://www.geeksforgeeks.org/out-parameter-with-examples-in-c-sharp/ - Заголовок з екрану.
9. Using keyboard events (Windows Forms .NET) [Електронний ресурс]: Режим доступу: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/input-keyboard/events?view=netdesktop-7.0 – Заголовок з екрану.
10. Keys Enum [Електронний ресурс]: Режим доступу: https://learn.microsoft.com/enus/dotnet/api/system.windows.forms.keys?view=windowsdesktop-7.0 – Заголовок з екрану.
11. KeyPress event in C# [Електронний ресурс]: Режим доступу: http://csharp.net-informations.com/gui/key-press-cs.htm - Заголовок з екрану.

12. TextBox Class [Електронний ресурс]: Режим доступу: https://learn.microsoft.com/enus/dotnet/api/system.windows.forms.textbox?view=windowsdesktop-7.0 – Заголовок з екрану.
13. Текстовое поле TextBox [Електронний ресурс]: Режим доступу: https://metanit.com/sharp/windowsforms/4.3.php - Заголовок з екрану.
14. Элементы GroupBox, Panel и FlowLayoutPanel [Електронний ресурс]: Режим доступу: https://metanit.com/sharp/windowsforms/3.2.php - Заголовок з екрану.
15. FlowLayoutPanel Class [Електронний ресурс]: Режим доступу: https://learn.microsoft.com/enus/dotnet/api/system.windows.forms.flowlayoutpanel?view=windowsdesktop-7.0 - Заголовок з екрану.
16. Потоковые классы и Строки [Електронний ресурс]: Режим доступу: https://ravesli.com/urok-210-potokovye-klassy-i-stroki/ - Заголовок з екрану.
17. std::replace, std::replace_if [Електронний ресурс]: Режим доступу: https://en.cppreference.com/w/cpp/algorithm/replace - Заголовок з екрану.
18. Label Class [Електронний ресурс]: Режим доступу: https://learn.microsoft.com/enus/dotnet/api/system.windows.forms.label?view=windowsdesktop-7.0 – Заголовок з екрану.
19. Компонент Label. Програмне створення елементу управління Label. Клас MessageBox. Зчислення DialogResult [Електронний ресурс]: Режим доступу: https://www.bestprog.net/uk/2022/07/21/c-windows-forms-the-label-component-programmatically-create-a-label-control-messagebox-class-ua/ - Заголовок з екран

ДОДАТОК

Переглянути код всього проєкту можна на веб-сайті github за посиланням: https://github.com/RadchenkoDmytro04/practice