**Д****НІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**ІМЕНІ ОЛЕСЯ ГОНЧАРА**

**ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ КАФЕДРА КОМП’ЮТЕРНИХ ТЕХНОЛОГІЙ**

**ЗВІТ**

**З НАВЧАЛЬНОЇ ПРАКТИКИ**

**КОМП’ЮТЕРНО - ТЕХНОЛОГІЧНОЇ**

Освітньо-професійна програма

Комп’ютерне моделювання та технології програмування

Спеціальність 113 Прикладна математика

Галузь знань 11 *Математика* і статистика

Рівень вищої освіти перший (бакалаврський)

Студента 2 курсу групи ПА-22-2

\_Овдієнко А.В,\_\_\_

(прізвище та ініціали)

Керівник \_Сірик С.Ф.\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Кількість балів \_\_\_\_\_\_\_\_\_\_\_

Національна шкала \_\_\_\_\_\_\_

Члени комісії:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Зайцев В.Г.

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Золотько К.Є.

(підпис) (прізвище та ініціали)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Сірик С.Ф.

(підпис) (прізвище та ініціали)

м. Дніпро, 2024 р.

**Зміст**

[Вступ 4](#_Toc169799941)

[Постановка задачі 5](#_Toc169799942)

[1 Реалізація графічного інтерфейсу програми 6](#_Toc169799943)

[1.1 Додавання границь функції 7](#_Toc169799944)

[1.2 Додавання кількості точок графіку 8](#_Toc169799945)

[1.3 Кнопка генерації та панель вибору графіка 9](#_Toc169799946)

[1.4 Переміщення та стиснення графіків 11](#_Toc169799947)

[1.5 Шапка – штучна панель інструментів 12](#_Toc169799948)

[2 Опис програмного забезпечення 15](#_Toc169799949)

[2.1 Мова програмування C++ 15](#_Toc169799950)

[2.2 Опис розв’язку завдання 16](#_Toc169799951)

[2.2.1. Вид графіку 16](#_Toc169799952)

[2.2.2. Вивід графіків на екран 16](#_Toc169799953)

[2.2.3. Вивід та ввід у файл точок 17](#_Toc169799954)

[2.2.4. Масштабування графіку 18](#_Toc169799955)

[2.2.5. Паралельний переніс графіку 18](#_Toc169799956)

[2.2.6. Виведення нового вікна на екран 18](#_Toc169799957)

[2.2.7. Формула для розрахунку кроку 18](#_Toc169799958)

[3 Аналіз отриманих результатів 19](#_Toc169799959)

[Висновки 20](#_Toc169799960)

[Список використаних джерел 21](#_Toc169799961)

[Рекомендовані джерела 21](#_Toc169799962)

[Основні 21](#_Toc169799963)

[Додаткові 21](#_Toc169799964)

[Інформаційні ресурси 21](#_Toc169799965)

[Додаток А 22](#_Toc169799966)

[Додаток Б 36](#_Toc169799967)

[Додаток В 39](#_Toc169799968)

[Додаток Г 41](#_Toc169799969)

[Додаток Ґ 42](#_Toc169799970)

[Додаток Д 43](#_Toc169799971)

[Додаток Е 45](#_Toc169799972)

[Додаток Є 47](#_Toc169799973)

[Додаток Ж 49](#_Toc169799974)

[Додаток З 51](#_Toc169799975)

[Додаток И 56](#_Toc169799976)

[Додаток І 57](#_Toc169799977)

[Додаток Ї 58](#_Toc169799978)

[Додаток Й 59](#_Toc169799979)

# Вступ

Проходження навчальної комп’ютерно - технологічної практики є невід’ємною складовою частиною під час навчального процесу – оскільки вона являється ефективним засобом для закріплення отриманих під час навчання знань, умінь і навичок, а також практика дозволяє студенту як майбутньому спеціалісту освоїти додаткові знання та отримати базовий досвід роботи з різними середовищами програмування.

При проходженні комп’ютерно – технологічної практики була поставлена задача - побудови графіку функції за індивідуальним завданням.

Для реалізації даного завдання потрібно:

* реалізувати побудову графіків функцій;
* надати оформлення: вісі (розмітка вісей, підписи вісей);
* вміти реалізувати побудову декількох графіків одночасно в одній системі координат: функції індивідуального завдання; функції sin(x); функції cos(x); функції, заданої рандомними числами; функції, заданої в текстовому файлі дискретно;
* зробити паралельний перенос графіку відносно вісей абсцис / ординат;
* зробити стиснення / розтягнення графіку функції.

Для виконання індивідуального завдання необхідно вдосконалити знання з мови програмування C++ у середовищі Visual Studio Code. Закріпити навички з курсу програмування, математичного аналізу, отримані під час навчання. Ознайомитися з базовою літературою, яку надано в начальній практиці.

# Постановка задачі

**Примітка:** функціональні можливості повинні бути реалізовані в тексті програми (засобами мови) і в інтерфейсі користувача (засобами візуальних компонент)

За допомогою графічних методів візуального середовища створити графіки функцій відповідно до індивідуального завдання №15

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 (номер варіанта Індивідуального завдання збігається з номером в студентському журналі);
5. Підготувати звіт про виконання завдання для навчальної практики: комп'ютерно – технологічної.

Технічні вимоги:

1. Вигляд (побудова) графіка;
2. вісі (розмітка вісей, підписи вісей, оформлення);
3. можливість виведення декількох графіків;
4. вивести графік функції, заданої в текстовому файлі дискретно, тобто значеннями функції в точках масштабування графіку;
5. переміщення графіка (паралельний перенос);
6. стиснення / розтягнення графіка функції щодо вісі абсцис / ординат.

# Реалізація графічного інтерфейсу програми

Інтерфейс(Рисунок 1.1) складається з прямокутників, які поступово перекривають один одного, відповідних текстових конструкторів, полів для вводу, ліній (віссі абцис та ординат),а також заднього фону – всі ці елементи розташовані у вікні sfml(Рисунок 1.1). Розмітка виконана так, щоб користувачеві було зручно працювати з програмою: одна частина призначена для відображення графіків (Рисунок 1.2), інша для головного меню (Рисунок 1.3), а також шапка(Рисунок 1.4) - для штучної панелі інструментів.

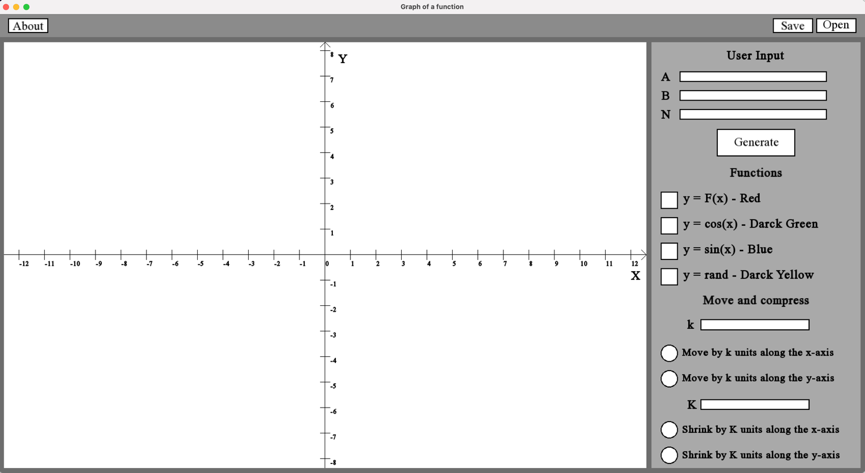


Рисунок 1.1 – Головне вікно проорами.

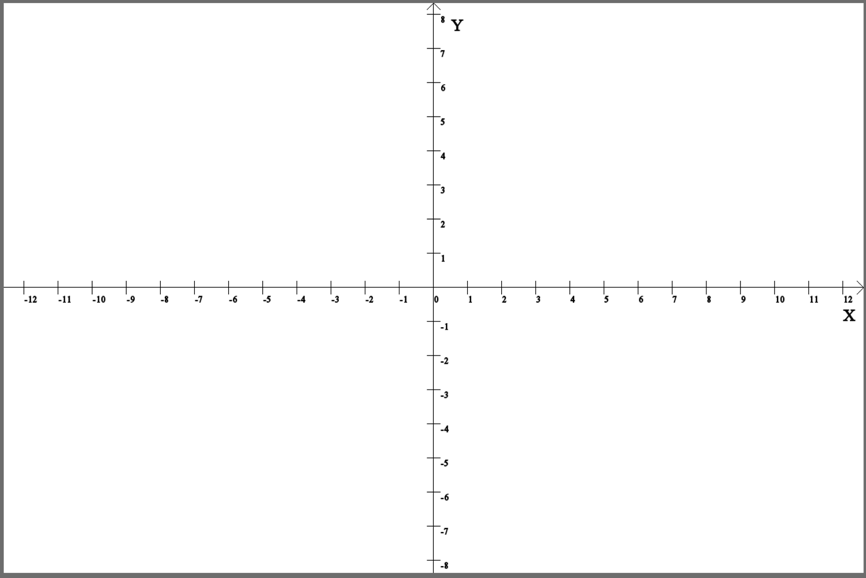


Рисунок 1.2 - Область для відображення графіків.

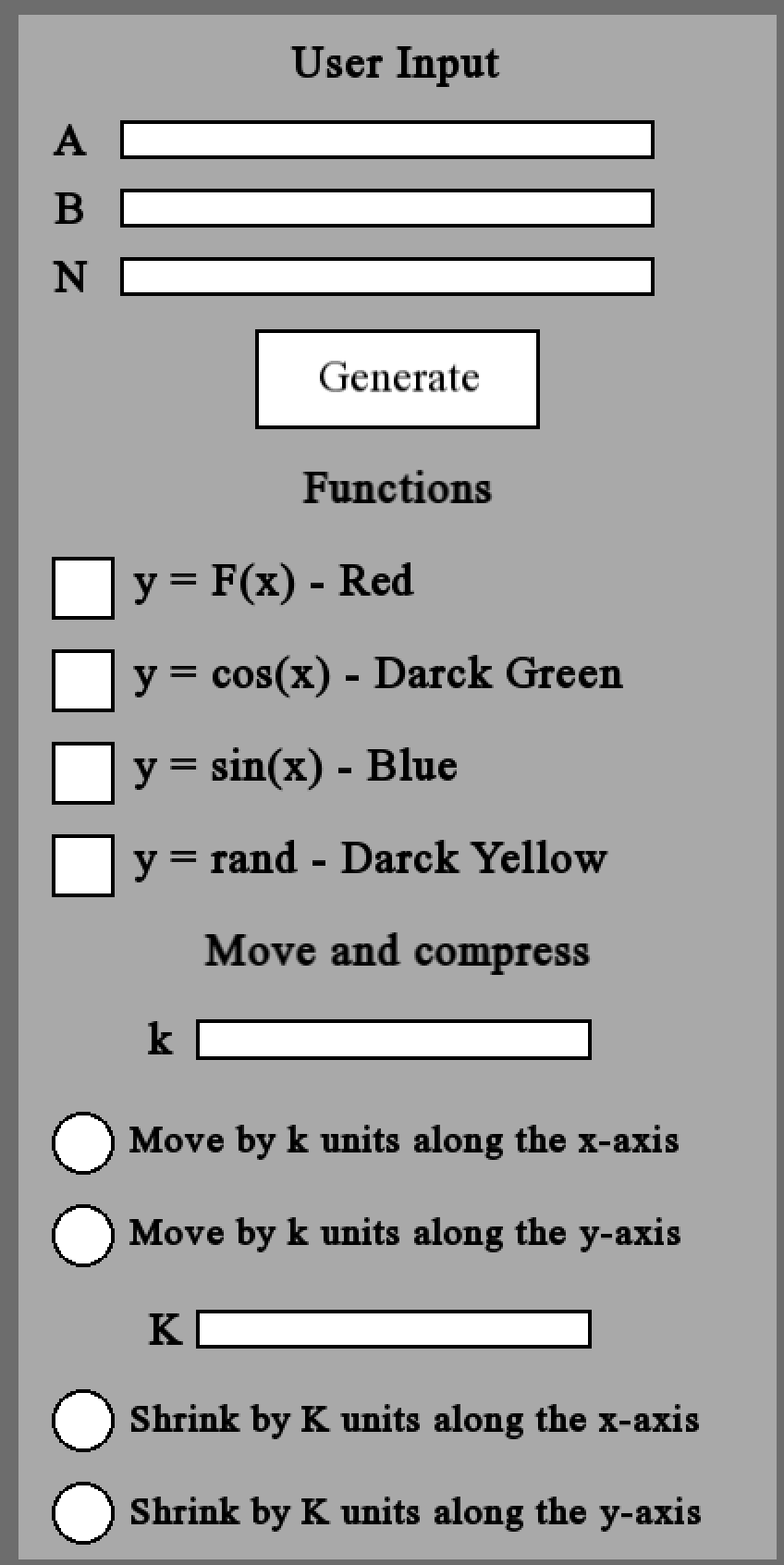


Рисунок 1.3 – Область головного меню для користувача.



Рисунок 1.4 – Область штучної панелі інструментів.

Звичайно в процесі роботи можна буде змінити позицію тексту, прямокутників, полів вводу, ліній за допомогою зміни позиції крайньої лівої верхньої точки обʼєкта.

## Додавання границь функції

Для додавання границь функції було використано два обʼєкта A(Рисунок 1.5) і B(Рисунок 1.5) , які мали: текстовий тип, який відповідає за підказки, що саме ми вводимо; прямокутник, щоб виділити область де будемо вводити текст; стрінг, для вводу текста, щоб перевіряти на коректність вводу символи, і звичайно щоб можна було вводити лише числа в правильній формі, тобто мінус лише першим символом, крапку лише один раз; змінну дробного типа, у якій буде зберігатися наше число для подальшого використання; змінну булівського типа, яка відповідає за натискання на проміжуток поля вводу; і звичайно текстове поле, яке буде відображати наш ввід. Обидва обʼєкта (Рисунок 1.5) в підказка так і підписані: A, B – для зручності.

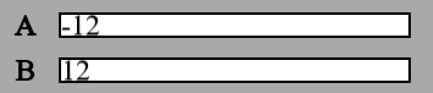


Рисунок 1.5 – Границі функцій.

Якщо користувач введе початкове значення інтервала, яке більше за кінцеве, то програма автоматично поміняє місцями ці два значення та всі їхні допоміжні обʼєкти, а також виведе повідомлення(Рисунок 1.6) на єкран у вигляді ще одного вікна(Рисунок 1.6), яка має домівнювання над основним.

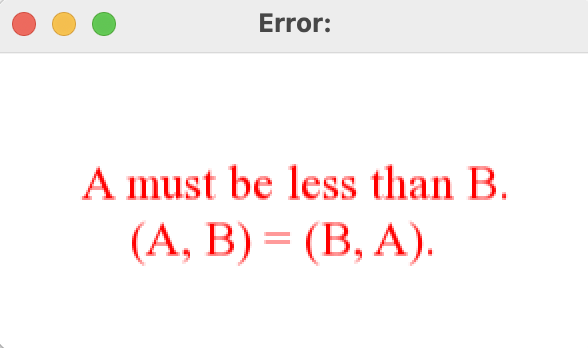


Рисунок 1.6 – Повідомлення про некоректні границі функції.

Для повідомлень типу спливаючих вікон, які продемонстровано на (Рисунок 1.6) для закриття слід нажати на кнопку закриття зліва вгорі, яка є в кожної програми.

## Додавання кількості точок графіку

Графік будується по точкам на проміжку, який задав користувач. Так само, як і проміжки, він має ввести й кількість точок із яких буде складатися графік. Тобто є ще один обʼєкт N(Рисунок 1.7), який потрібен лише для кількісті точок. Також має: текстовий тип, який відповідає за підказки, що саме ми вводимо; прямокутник, щоб виділити область де будемо вводити текст; стрінг для вводу текста, щоб перевіряти на коректність вводу символи, і звичайно щоб можна було вводити лише натуральні числа не більше ста тисяч; змінну цілочисельного типа, у якій буде зберігатися наше число для подальшого використання; змінну булівського типа, яка відповідає за натискання на проміжуток поля вводу; і звичайно текстове поле, яке буде відображати наш ввід.



Рисунок 1.7 – Кількість точок функції на певному проміжку.

І якщо ми не введемо проміжки A(Рисунок 1.5), B(Рисунок 1.5) чи кількість точок N(Рисунок 1.7), або введемо занадто велике N(Рисунок 1.7), то отримаємо повідомлення(Рисунок 1.8) з вказівками про помилку вводу.



Рисунок 1.8 – Повідомлення про помилку вводу границь чи кількості точок.

## Кнопка генерації та панель вибору графіка

Після введення проміжків(Рисунок 1.5) функції та кількості точок(Рисунок 1.7) графіків слід натиснути на кнопку генерації(Рисунок 1.9), щоб графіки сворилися.



Рисунок 1.9 – Кнопка генерації графіків.

Тепер вже можна обирати графіки(Рисунок 1.10) функцій для відображення.

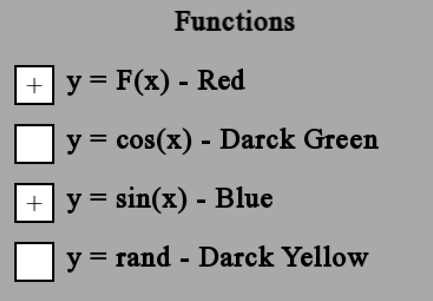


Рисунок 1.10 – Функції, які можна вибрати для відображення графіків.

Зазначимо, що були створені додаткові функції sin(x) та cos(x) для демонстрації правильної побудови. Поруч із функцією написан колір – це колір графіку. Також можна помітити рандомну функцію, якщо ми захочемо її відобразити, то відкривається вікно(Рисунок 1.11) для вибору границь для рандомної функції.

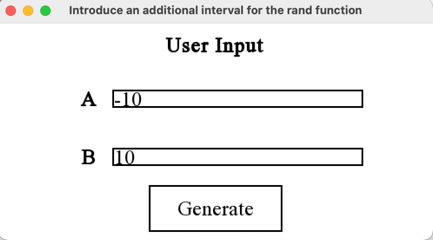


Рисунок 1.11 – Вікно для вибору границь рандомної функції.

Якщо користувач введе початкове значення інтервала, яке більше за кінцеве, то програма автоматично поміняє місцями ці два значення та всі їхні допоміжні обʼєкти, а також виведе повідомлення(Рисунок 1.6) на єкран у вигляді ще одного вікна(Рисунок 1.6), яка має домівнювання над основним. І слід натиснути на кнопку генерації(Рисунок 1.12) у цьому вікні(Рисунок 1.11).

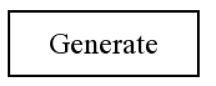


Рисунок 1.12 – Кнопка генерації рандомної функції.

І побудується графік з рандомними функціями у кожній точці, а кількість точок(Рисунок 1.7) візьметься з головного меню(Рисунок 1.3).

І щойно згенерується рандомний графік – на області вибору функцій для відображення зʼявиться проміжок(Рисунок 1.14) для рандомної функції на виборі функцій(Рисунок 1.13).

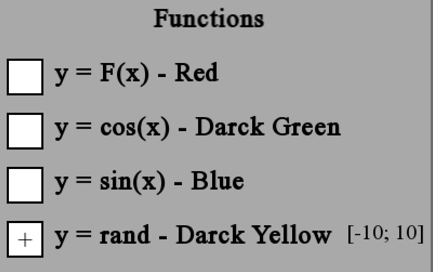


Рисунок 1.13 – Виділений графік функції із заданими проміжками.

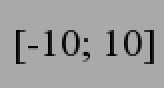


Рисунок 1.14 – Заданий проміжок рандомної функції.

І якщо користувач вирішив змінити кількість точок(Рисунок 1.7), але не натиснув на кнопку генерації(Рисунок 1.9) – то вивідеться повідомлення про помилку(Рисунок 1.8), що слід також натиснути на кнопку генерації(Рисунок 1.9).

## Переміщення та стиснення графіків

Для переміщення та стиснення графіків використовувалися поля для введення руху та стиснення графіка.

Поле для вводу переміщення графіків(Рисунок 1.15): текстовий тип, який відповідає за підказки, що саме ми вводимо; прямокутник, щоб виділити область де будемо вводити текст; стрінг, для вводу текста, щоб перевіряти на коректність вводу символи, і звичайно щоб можна було вводити лише числа в правильній формі, тобто мінус лише першим символом, крапку лише один раз; змінну дробного типа, у якій буде зберігатися наше число для подальшого використання; змінну булівського типа, яка відповідає за натискання на проміжуток поля вводу; і звичайно текстове поле, яке буде відображати наш ввід.

Поле для вводу стискання графіків(Рисунок 1.16): текстовий тип, який відповідає за підказки, що саме ми вводимо; прямокутник, щоб виділити область де будемо вводити текст; стрінг, для вводу текста, щоб перевіряти на коректність вводу символи, і звичайно щоб можна було вводити лише числа в правильній формі, тобто лише позитивні числа, крапку лише один раз; змінну дробного типа, у якій буде зберігатися наше число для подальшого використання; змінну булівського типа, яка відповідає за натискання на проміжуток поля вводу; і звичайно текстове поле, яке буде відображати наш ввід.

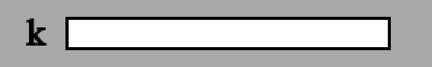


Рисунок 1.15 – Поле вводу для переміщення графіків.



Рисунок 1.16 – Поле вводу для стискання графіків.

Також є кнопки для застосування переміщення по осям(Рисунок 1.17), натиснувши на яких береться число із поля для переміщення(Рисунок 1.15) і переміщуються вибрані графіки(Рисунок 1.10).

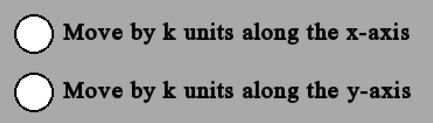


Рисунок 1.17 – Кнопки з підказками для переміщення функцій по осям.

Також є кнопки для застосування стиснення графіків по осям(Рисунок 1.18), натиснувши на яких береться число із поля для стиснення(Рисунок 1.16) і свужуються вибрані графіки(Рисунок 1.10).

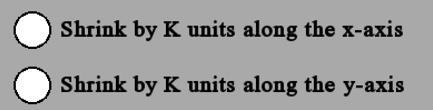


Рисунок 1.18 – Кнопки з підказками для стиснення функцій по осям.

І якщо ми нічого не введемо і cпробуємо перемістити графіки – програма виведе повідомлення(Рисунок 1.19), де говорить про дії, які немають сенсу: додавання нуля, множення на нуль.

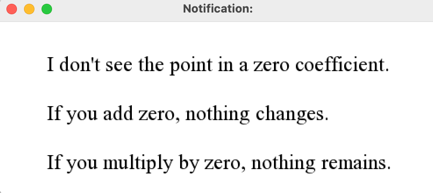


Рисунок 1.19 – Повідомлення про некоректне переміщення чи стискання.

## Шапка – штучна панель інструментів

Штучна панель інструментів(Рисунок 1.4) представляє собою три однакові кнопки, але різні за назвами: “About”, “Save”, “Open”.

Якщо людина натисне на кнопку «About», то відобразиться вікно (Рисунок 1.20), де описана коротка інформація про розробника та програму.

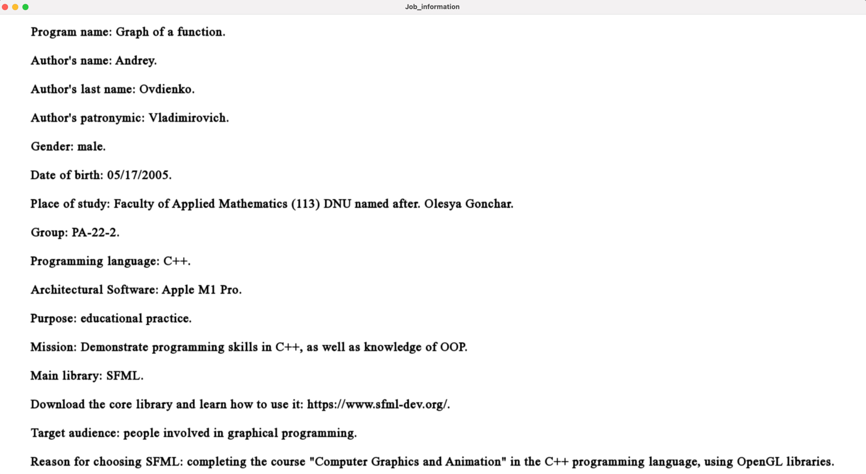


Рисунок 1.20 – Інформація про розробника та програму.

Якщо людина натисне на кнопку «Save», то відобразиться вікно(Рисунок 1.21), де можна написати імʼя файлу, чи повний шлях, зберігти у тестовому файлі в форматі “.txt” - натиснувши на кнопку “Save” у цьому вікні(Рисунок 1.21).

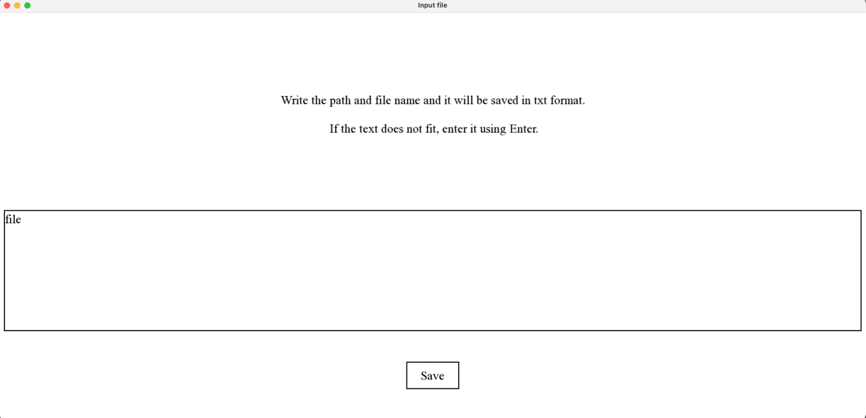


Рисунок 1.21 – Вікно для вводу назви файлу, у який все збережеться.

Після збереження ввивідиться вікно(Рисунок 1.22), що файл збережено та його повна назва. Щоб закрити вікно – натиснути на кнопку “OK” у цьому вікні(Рисунок 1.22). Тобто ми зберігли всі графіки, в один файл.



Рисунок 1.22 – Вікно про збереження данних у відповідний файл.

Майже те саме для “Open”. У нас є відповідне вікно(Рисунок 1.23), куди можна ввести назву файлу, який буде відкритий

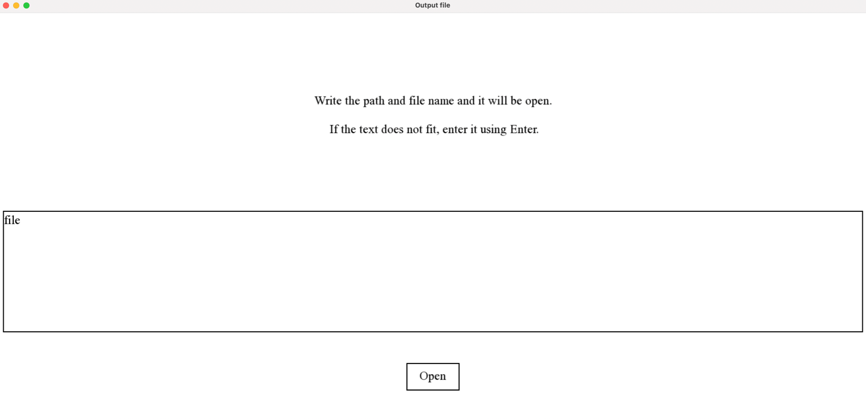


Рисунок 1.23 – Вікно для вводу файлу, який ми хочемо відкрити.

Якщо такий файл є і він відкрився – буде відповідне повідомлення(Рисунок 1.24).



Рисунок 1.24 – Вікно про відкриття файлу, де знаходяться данні для програми.

Якщо такого файлу немає то буде альтернативне повідомлення(Рисунок 1.25).



Рисунок 1.25 – Помилка відкриття файлу.

# Опис програмного забезпечення

## Мова програмування C++

Для створення графічного інтерфейсу використаємо бібліотеку sfml, яку можна завантажити із офіційного сайту [SFML](https://www.sfml-dev.org/) (Рисунок 2.1).



Рисунок 2.1 - Головна сторінка офіційного сайту SFML.

Зверху можна побачити розділи де можна навчитися працювати з цією бібліотекою, завантажити її, громада та розробники. Ми підготували відповідну бібліотеку. Тепер слід налаштувати Visual Studio Code для роботи С++ і звісно встановити його. Зручне середовище для написання коду, а також маже єдине можливе для C++ - Visual Studio Code. Щоб завантажити цю середу розробки мені допоміг сайт Microsoft(Рисунок 2.2) [Visual Studio](https://visualstudio.microsoft.com/downloads/?utm_medium=post-banner&utm_source=microsoft.com&utm_campaign=channel+banner&utm_content=launch+vs2022&icid=mscom_marcom_dlc_pfdev). Де я міг обрати своою операційну систему та бажану мову програмування.

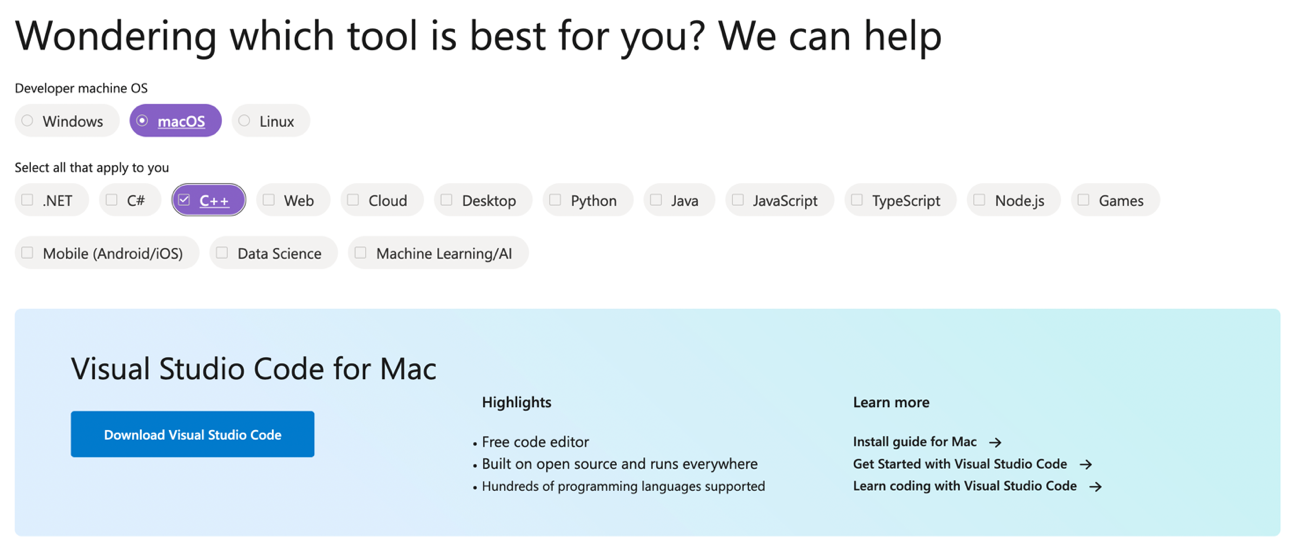


Рисунок 2.2 - Порада Microsoft обрати та встановити середу розробки Visual Studio Code.

Слід завантажити додаткові розширення С/С++ від Microsoft(Рисунок 2.3).

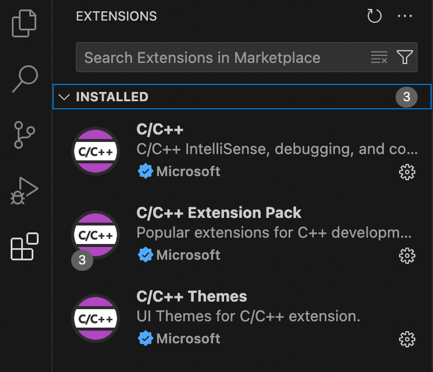


Рисунок 2.3 - – Розширення для роботи на С++ у Visual Studio Code від Microsoft.

Для налаштування коду, тобто щоб компілювалася програма – слід сворити скрипт(Рисунок 2.4) файл формату “.bat”, де ми будемо викликати нашу бібліотеку sfml, створювати та відкривати виконувальний файл.

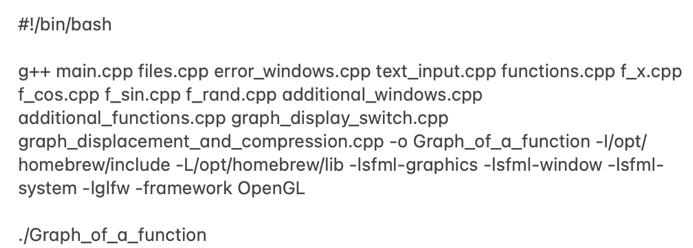


Рисунок 2.4 – Скрипт компіляції та відкриття програми на SFML із додатковими файлами.

## Опис розв’язку завдання

### Вид графіку

Для зовнішнього вигляду графіка я взяв точки типу «CircleShape» радіусом два пікселі, бо це простий спосіб відображення графіка.

### Вивід графіків на екран

Беремо наш інтервал(Рисунок 1.5), який вказав користувач; і кількість точок(Рисунок 1.7), яка також встановлюється користувачем. Якщо у нас одна точка, то зрозуміло, що її треба брати в середині проміжка(Рисунок 1.5). Якщо ми маємо 2 або більше точок, то ми можемо припустити, що якщо ми маємо n кількість відрізків на інтервалі, то ми маємо (n+1) вузлову точку разом із крайовими. Тому наше завдання зводиться до розрахунку відстані цього кроку, тобто кроку між сусідніми вузловими точками, підкреслюю, що для рівномірного відображення крок між точками ставився однаковим. Усі точки зберігаються у масивах xi, yi, які в свою чергу зберігаються у конретному класі функції.

### Вивід та ввід у файл точок

Щоб записати дані у файл я використав додаткову юіюліотеку “fsream” для роботи з файлами. Коли я отримував від користувача назву файлу, то приписував до нього розширення “.txt”, щоб працювати із текстовим файлом. Далі створювався цей файл та відкривався. Першим числом записується кількість точок для графіків F(x), sin(x), cos(x) - усі вони мають однакову кількість точок. Якщо кількість точок нульова, тобто їх немає – пишемо 0, якщо ж там є якесь число, то ми прописуємо проміжки функцій(Рисунок 1.5) - вони однакові, далі записуємо через пробіл точки х та у для функції F(x), так само потім для cos(x) і sin(x). Далі йде перевірка, якщо рандомний графік був створенний, то має свою кількість точок. Якщо точок немає – пишемо нуль та закриваємо файл. Якщо там щось є – знов пишемо розмір а потім наш допоміжний інтервал для рандомної функції(Рисунок 1.11) і так само, як і точки для F(x) запишемо точки для рандомної функції. Закриємо файл та виведемо повідомлення про збереження(Рисунок 1.22).

Щоб дістати дані із фалу, беремо в користувача назву файлу та додаємо розширення “.txt”. Якщо не вдалося відкрити, то виводимо відповідне повідомлення(Рисунок 1.25). Якщо вдалося, то перевірка на нуль. Якщо нуль – значить функцій: F(x), cos(x), sin(x) – немає, йдемо далі. Якщо ж там все ж таки щось є, то надаємо нашим масивам точок той розмір, який є у файлі. Далі передаємо інтервал(Рисунок 1.5) до полів вводу(Рисунок 1.5), і зчитуємо х і у точки із функцій: F(x), cos(x), sin(x). За тим йде перевірка на наявність розміра у рандомної функції. Якщо кількість точок рандомної функції нульова – її немає. Якщо у функції щось є - надаємо нашим масивам точок той розмір, який є у файлі. Далі передаємо інтервал(Рисунок 1.11) до полів вводу(Рисунок 1.11), і зчитуємо х і у точки із рандомної функції. Закриваємо файл та виводимо вікно(Рисунок 1.24), що дані взяті з файлу.

### Масштабування графіку

Для маштабування будь-якого графіка бралися коефіцієнти K(Рисунок 1.16), якщо він нульовий, то виводилося повідомлення(Рисунок 1.19) про дивні дії. Якщо не нульовий, то в нас є відповідні точки (xi, yi), точка початку координат (x0,y0) та наш коефіцієнт стискання. Нагадаємо, що розмір точок, з який складається графік, є радіусом 2 байта. Віднімаємо початкові координати від кожної точки, до кожної точки додаємо 2, кожну точку помножуємо на коефіцієнт, знов додаємо початкові координати та віднімаємо 2. Це зроблено так, бо в компʼютері точка (0,0) починається із вірхнього лівого кута.

### Паралельний переніс графіку

Для паралельного переносу будь-якого графіка бралися коефіцієнти k(Рисунок 1.15), якщо він нульовий, то виводилося повідомлення(Рисунок 1.19) про дивні дії. Якщо не нульовий, то в нас є відповідні точки (xi, yi) та наш коефіцієнт зсуву. Додаємо до кожної точки наш коефіцієнт.

### Виведення нового вікна на екран

Для виводу нового вікна використовувалося те, що можна створювати ще вікна sfml. Тобто додаткові вікна створювалися так само, як і головне вікно(Рисунок 1.1), але там були свої розміри назви, кнопки, тексти, поля вводу. Всі вікна закриваються за допомогою команди “window.close()” при натисненні на певну кнопку або крестик.

### Формула для розрахунку кроку

Якщо у нас є одна точка для відображення графіка, то це має бути середнє арифметичне нашого проміжку(Рисунок 1.5). Якщо у нас більше однієї точки, то ми маємо крок, тобто початковий x у точці A(Рисунок 1.5), і рівновіддалені точки чкількості N(Рисунок 1.7) до точки B(Рисунок 1.5). Для цього розрахуємо крок: від кінцевої точки віднімаємо початкову і ділимо на кількість відрізків. Враховуємо, що якщо точок n, то кількість рівномірних відрізків дорівнює (n-1).

# Аналіз отриманих результатів

Перевіримо правильність відображення графіка індивідуального завдання. Для цього побудуємо його на проміжку [-12,12] у мої програмі написаній на С++(Рисунок 3.1) та у [WOLFRAM CLOUD](https://www.wolframcloud.com/)(Рисунок 3.2).

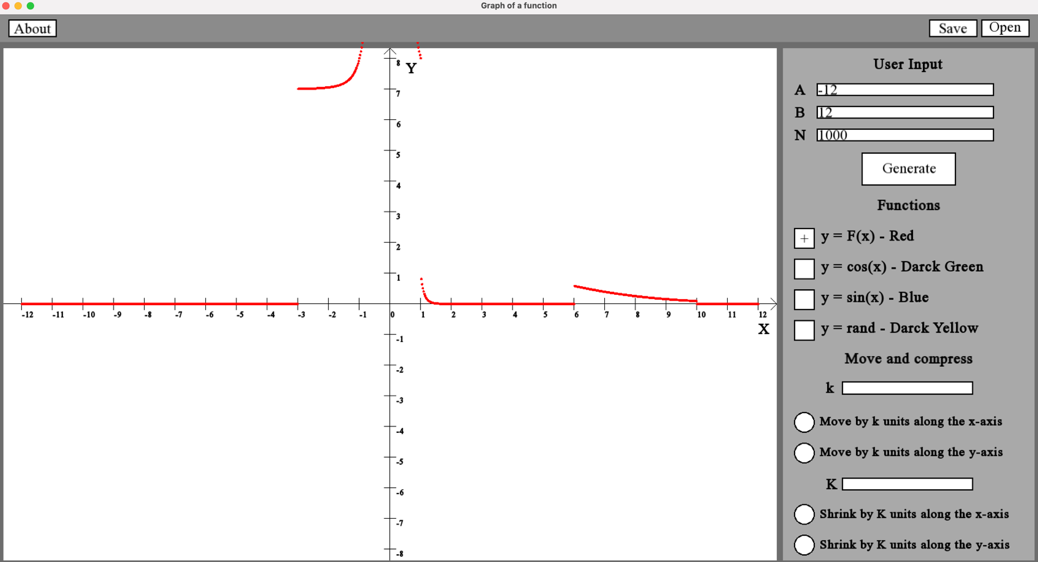


Рисунок 3.1 – Графік індивідуальної функції у зробленій програмі.

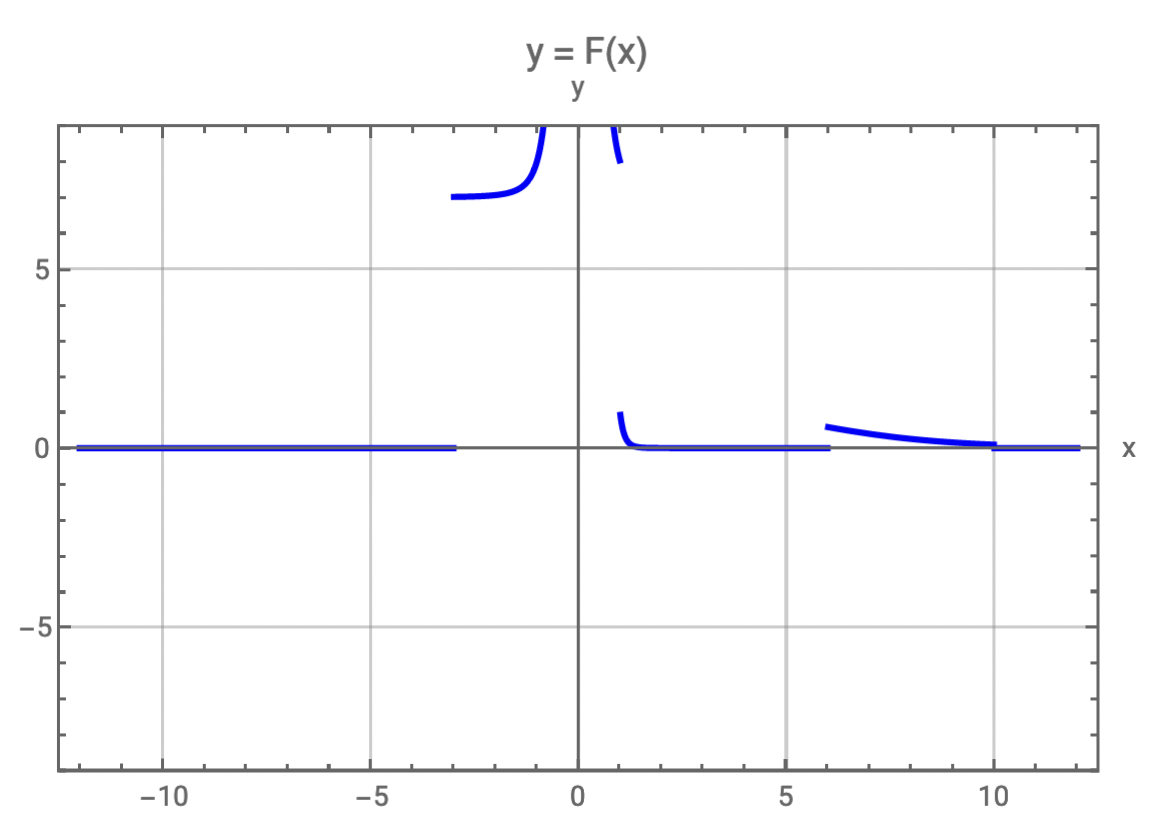


Рисунок 3.2 – Графік індивідуальної функції у WOLFRAM CLOUD.

Можна побачити, що графіки на С++(Рисунок 3.1) та у [WOLFRAM CLOUD](https://www.wolframcloud.com/)(Рисунок 3.2) нічим не відрізняються, крім кольору відмальовки, тобто я зробив правильну функцію індивідуального варіанта та коректно її відобразив.

# Висновки

Під час стажування з комп’ютерних технологій я навчився візуальному програмуванню на C++ за допомогою графічної бібліотеки SFML. Тепер я можу створити програму з різними кольорами та стилями. Я знаю, як створювати різні діаграми розсіювання з певними інтервалами та відтінками кольорів.

Робота складалася з побудови плану дій, створення графічного інтерфейсу, створення точок і надання їм координат для відображення, розрахованих на основі проміжку та кількості точок, які надав користувач, обробки паралельного переносу, стиснення, працювання з файлами збереження/відкриття даних, створення інформації про програму.

Роботи була великою, тому додано окремі програмні модулі, які складаються з: заголовного файлу, сурс файлу. Додаток А відповідає за основний програмний код, до якого підключаються всі інші модулі. У Додатку Б зберігаються вікна, які відкриваються для введення імені файлу для збереження/відкриття даних. Додаток В містить вікна помилок, коли, наприклад, не введено межу функції або кількість точок, або відсутні коефіцієнти, неправильно записані початкове і кінцеве значення інтервалу. Додаток Г відповідає за можливість вводити лише числа в полях для інтервалів, кількість точок і коефіцієнтів, як правило, з можливістю видалення. Додаток Ґ містить базовий клас функцій, який буде розроблено пізніше, тобто батьківський клас. Додаток Д — це похідний клас від класу в Додатку Ґ, який відповідає за обробку та надання значень x і y для індивідуальної функції. Додатки Е, Є, Ж роблять те саме, що й додаток Д, але для функцій: cos(x), sin(x), випадкова функція. Додаток З відповідає за додаткові вікна: встановлення випадкових інтервалів розкладу, налаштування вікон інформації про програму та користувача, збереження даних у файл та їх отримання. Додаток И містить функцію, яка видаляє всі кінцеві нулі з рядка змінної, якщо немає інших символів. Додаток І відповідає за кнопки для відображення графіків зі знаком плюс, коли графік відображається. Додаток Ї відповідає за кнопки переміщення та ступення стиснення.

# Список використаних джерел

## Рекомендовані джерела

### Основні

1. Гук Н.А., Єгошкін Д.І., Сірик С.Ф. Алгоритм класифікації на базі нечіткої логіки з розширюваною кількістю виводів // Зб. наук. праць «Питання прикладної математики і математичного моделювання». – Дніпро. – 2018. – Вип. 18. – С. 33-41.
2. Приклад коду програмування графіки в Microsoft Visual Studio [http://khizha.dp.ua/library/graphics\_in\_WindowsForms/graphics in WF.rar](http://khizha.dp.ua/library/graphics_in_WindowsForms/graphics%20in%20WF.rar)
3. Козак Л. І., Костюк І. В., Стасевич С. П. Основи програмування: навчальний посібник – Львів:«Новий Світ-2000», 2020. – 328с.

### Додаткові

1. Юрій Грицюк, Тарас Рак. ПРОГРАМУВАННЯ МОВОЮ С++, Львів Вид-во ЛДУ БЖД 2011,- 288 с
2. Трофименко О.Г., Прокоп Ю.В., Швайко І.Г., Буката Л.М., Косирева Л.А.. Основи програмування. Теорія, практика: підручник. Одеса, Фенікс 2010, 534 с.
3. Ковалюк Т. В.К 56 Алгоритмізація та програмування: Підручник. — Львів: «Магнолія 2006», 2013. — 400 с.
4. Нестеренко В., Краєвська О., Чермних І., Адашевська І., Сілічев А. Основи інженерної графіки з елементами професійного конструювання, Київ: Кондор. -2020. -220с.

## Інформаційні ресурси

1. Робота в графіці у платформі .NET Framework на мові C++ <http://msdn.microsoft.com/ru-ru/library/9be4455y%28v=vs.90%29.aspx>
2. Робота з графікою у бібліотеці sfml на мові С++ <https://www.sfml-dev.org/tutorials/2.6/>

# Додаток А

Код основої програми

// Підключення додаткових файлів

#include "Files.h"

#include "Error\_windows.h"

#include "Text\_input.h"

#include "Functions.h"

#include "F\_x.h"

#include "F\_cos.h"

#include "F\_sin.h"

#include "F\_rand.h"

#include "Additional\_windows.h"

#include "Additional\_functions.h"

#include "Graph\_display\_switch.h"

#include "Graph\_displacement\_and\_compression.h"

// Підключення стандартної бібліотеки для роботи з потоками

#include <fstream>

// Підключення стандартноі бібліотеки для вводу/виводу та простору імен std

#include <iostream>

using namespace std;

// Підключення бібліотеки sfml для вграфіки та простору імен sf

#include <SFML/Graphics.hpp>

using namespace sf;

int main(){

// Встановлення розмірів основного вікна, назви та розрахування ширини для відображення графіків та меню користувача

int WIDTH = 1700, HEIGHT = 900;

char NAME[] = "Graph of a function";

float USER\_INPUT\_WIDTH = WIDTH \* (1./ 4) - 15;

float FUNCTION\_GRAPH\_WIDTH = WIDTH \* (3./4) - 15;

// Створення вікна

RenderWindow window(VideoMode(WIDTH, HEIGHT), NAME);

// Створення області для шапки програми - область штуччної панелі інструментів

RectangleShape head\_background(Vector2f(WIDTH, 45));

head\_background.setPosition(0, 0);

head\_background.setFillColor(Color(139, 139, 139));

// Створення області для відображення графіка

RectangleShape graph\_background(Vector2f(FUNCTION\_GRAPH\_WIDTH, HEIGHT - 65));

graph\_background.setPosition(10, 55);

graph\_background.setFillColor(Color::White);

// Створення області для шменю користувача

RectangleShape user\_background(Vector2f(USER\_INPUT\_WIDTH, HEIGHT - 65));

user\_background.setPosition(FUNCTION\_GRAPH\_WIDTH + 20, 55);

user\_background.setFillColor(Color(169, 169, 169));

// Створення шрифта

Font font;

// Встановлення шрифта з файла та перевірка відкриття

font.loadFromFile("/Users/admin/Documents/My documends/Університет/ДНУ/2 Курс/2 Семестр/Навчальна практика/Times\_New\_Roman-font.ttf");

if (!font.loadFromFile("/Users/admin/Documents/My documends/Університет/ДНУ/2 Курс/2 Семестр/Навчальна практика/Times\_New\_Roman-font.ttf"))

{

cerr << "\nError opening file Times\_New\_Roman-font.ttf.\n";

return 1;

}

// Створення підказок, що має ввести користувач

Text user\_hint("User Input", font, 24);

user\_hint.setFillColor(Color::Black);

user\_hint.setStyle(Text::Bold);

user\_hint.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + (USER\_INPUT\_WIDTH - user\_hint.getLocalBounds().width)/2, 65);

Text user\_hint\_A("A", font, 24);

user\_hint\_A.setFillColor(Color::Black);

user\_hint\_A.setStyle(Text::Bold);

user\_hint\_A.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + user\_hint\_A.getLocalBounds().width, 85 + user\_hint.getLocalBounds().height);

Text user\_hint\_B("B", font, 24);

user\_hint\_B.setFillColor(Color::Black);

user\_hint\_B.setStyle(Text::Bold);

user\_hint\_B.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + user\_hint\_A.getLocalBounds().width, 105 + user\_hint.getLocalBounds().height + user\_hint\_A.getLocalBounds().height);

Text user\_hint\_N("N", font, 24);

user\_hint\_N.setFillColor(Color::Black);

user\_hint\_N.setStyle(Text::Bold);

user\_hint\_N.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + user\_hint\_A.getLocalBounds().width, 125 + user\_hint.getLocalBounds().height + user\_hint\_A.getLocalBounds().height + user\_hint\_B.getLocalBounds().height);

// Створення білих прямокутників із чорною оболочкою, для візуального розуміння де буде текст та куди клікати, щоб його написати

RectangleShape user\_A\_wright(Vector2f(15 \* user\_hint\_A.getLocalBounds().width, user\_hint\_A.getLocalBounds().height));

user\_A\_wright.setFillColor(Color::Transparent);

user\_A\_wright.setFillColor(Color::White);

user\_A\_wright.setOutlineColor(Color::Black);

user\_A\_wright.setOutlineThickness(2);

user\_A\_wright.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + 3 \* user\_hint\_A.getLocalBounds().width, 92 + user\_hint.getLocalBounds().height);

RectangleShape user\_B\_wright(Vector2f(15 \* user\_hint\_A.getLocalBounds().width, user\_hint\_A.getLocalBounds().height));

user\_B\_wright.setFillColor(Color::Transparent);

user\_B\_wright.setFillColor(Color::White);

user\_B\_wright.setOutlineColor(Color::Black);

user\_B\_wright.setOutlineThickness(2);

user\_B\_wright.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + 3 \* user\_hint\_A.getLocalBounds().width, 112 + user\_hint.getLocalBounds().height + user\_hint\_A.getLocalBounds().height);

RectangleShape user\_N\_wright(Vector2f(15 \* user\_hint\_A.getLocalBounds().width, user\_hint\_A.getLocalBounds().height));

user\_N\_wright.setFillColor(Color::Transparent);

user\_N\_wright.setFillColor(Color::White);

user\_N\_wright.setOutlineColor(Color::Black);

user\_N\_wright.setOutlineThickness(2);

user\_N\_wright.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + 3 \* user\_hint\_A.getLocalBounds().width, 132 + user\_hint.getLocalBounds().height + user\_hint\_A.getLocalBounds().height + user\_hint\_B.getLocalBounds().height);

// Створення обʼєктів для відображення введеного текста

Text text\_display\_written\_A("", font, 24);

text\_display\_written\_A.setFillColor(Color::Black);

text\_display\_written\_A.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + 3 \* user\_hint\_A.getLocalBounds().width, 84.5 + user\_hint.getLocalBounds().height);

Text text\_display\_written\_B("", font, 24);

text\_display\_written\_B.setFillColor(Color::Black);

text\_display\_written\_B.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + 3 \* user\_hint\_A.getLocalBounds().width, 104.5 + user\_hint.getLocalBounds().height + user\_hint\_A.getLocalBounds().height);

Text text\_display\_written\_N("", font, 24);

text\_display\_written\_N.setFillColor(Color::Black);

text\_display\_written\_N.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + 3 \* user\_hint\_A.getLocalBounds().width, 124.5 + user\_hint.getLocalBounds().height + user\_hint\_A.getLocalBounds().height + user\_hint\_B.getLocalBounds().height);

// Змінні для розрахункових дій

float number\_A = 0.0;

float number\_B = 0.0;

float number\_A2 = 0.0;

float number\_B2 = 0.0;

int number\_N = 0.0;

float number\_k = 0.0;

float number\_K = 0.0;

// Змінні для перевірки натискання

bool click\_A = false;

bool click\_B = false;

bool click\_A2 = false;

bool click\_B2 = false;

bool click\_N = false;

bool click\_k = false;

bool click\_K = false;

bool click\_save\_button = false;

bool click\_open\_button = false;

bool click\_generate\_button = false;

bool click\_about\_button = false;

// змінні для перевірки вводу

string text\_assistant\_A;

string text\_assistant\_B;

string text\_assistant\_N;

string text\_assistant\_k;

string text\_assistant\_K;

string text\_assistant\_A2;

string text\_assistant\_B2;

// встановлення точки цунтру координатних осей - (0;0)

int x0 = 10 + graph\_background.getLocalBounds().width/2;

int y0 = 55 + graph\_background.getLocalBounds().height/2;

// Задання кількості цифр для відображення на осях від нуля до певного числа

int number\_of\_segments\_x = 13;

int number\_of\_segments\_y = 9;

int move = 50;

int index;

// Створення квадратних кнопок для надання можливості відображення графіків

square\_buttons\_with\_selection\_of\_displayed\_functions f\_x(font, 0), f\_cos\_x(font, 1), f\_sin\_x(font, 2), f\_rand\_x(font, 3);

// Створення функцій з відповідними початковими координатами

f\_F F\_x\_(x0, y0);

f\_cos F\_cos\_(x0, y0);

f\_sin F\_sin\_(x0, y0);

f\_rand F\_rand\_(x0, y0);

// Створення круглихх кнопок для надання можливості зсуву та маштабування графіків

round\_buttons\_for\_moving\_and\_compressing\_graphs move\_x(font, 6), move\_y(font, 7), compress\_x(font, 9), compress\_y(font, 10);

// Створення віссі абцис

VertexArray Ox(Lines, 2);

Ox[0].position = Vector2f(10, y0);

Ox[0].color = Color::Black;

Ox[1].position = Vector2f(10 + FUNCTION\_GRAPH\_WIDTH, y0);

Ox[1].color = Color::Black;

// Створення віссі ординат

VertexArray Oy(Lines, 2);

Oy[0].position = Vector2f(x0, 55);

Oy[0].color = Color::Black;

Oy[1].position = Vector2f(x0, 55 + HEIGHT - 65);

Oy[1].color = Color::Black;

// Створення правої частини стрілки напряму віссі абцис

VertexArray right\_side\_of\_the\_arrow\_x(Lines, 2);

right\_side\_of\_the\_arrow\_x[0].position = Vector2f(x0+x0-10, y0);

right\_side\_of\_the\_arrow\_x[0].color = Color::Black;

right\_side\_of\_the\_arrow\_x[1].position = Vector2f(x0+x0-20, y0+10);

right\_side\_of\_the\_arrow\_x[1].color = Color::Black;

// Створення лівої частини стрілки напряму віссі абцис

VertexArray left\_side\_of\_the\_arrow\_x(Lines, 2);

left\_side\_of\_the\_arrow\_x[0].position = Vector2f(x0+x0-10, y0);

left\_side\_of\_the\_arrow\_x[0].color = Color::Black;

left\_side\_of\_the\_arrow\_x[1].position = Vector2f(x0+x0-20, y0-10);

left\_side\_of\_the\_arrow\_x[1].color = Color::Black;

// Створення правої частини стрілки напряму віссі ординат

VertexArray right\_side\_of\_the\_arrow\_y(Lines, 2);

right\_side\_of\_the\_arrow\_y[0].position = Vector2f(x0, 55);

right\_side\_of\_the\_arrow\_y[0].color = Color::Black;

right\_side\_of\_the\_arrow\_y[1].position = Vector2f(x0+10, 65);

right\_side\_of\_the\_arrow\_y[1].color = Color::Black;

// Створення лівої частини стрілки напряму віссі ординат

VertexArray left\_side\_of\_the\_arrow\_y(Lines, 2);

left\_side\_of\_the\_arrow\_y[0].position = Vector2f(x0, 55);

left\_side\_of\_the\_arrow\_y[0].color = Color::Black;

left\_side\_of\_the\_arrow\_y[1].position = Vector2f(x0-10, 65);

left\_side\_of\_the\_arrow\_y[1].color = Color::Black;

// Створення одиничних відрізків на позитивній області віссі абцис

VertexArray positive\_segments\_x(Lines, number\_of\_segments\_x \* 2);

for (index = 0; index < number\_of\_segments\_x; index++) {

positive\_segments\_x[index \* 2].color = Color::Black;

positive\_segments\_x[index \* 2].position = Vector2f(x0 + move \* index, y0 - 10);

positive\_segments\_x[index \* 2 + 1].color = Color::Black;

positive\_segments\_x[index \* 2 + 1].position = Vector2f(x0 + move \* index, y0 + 10);

}

// Створення одиничних відрізків на негативній області віссі абцис

VertexArray negative\_segments\_x(Lines, number\_of\_segments\_x \* 2);

for (index = 0; index < number\_of\_segments\_x; index++) {

negative\_segments\_x[index \* 2].color = Color::Black;

negative\_segments\_x[index \* 2].position = Vector2f(x0 - move \* index, y0 - 10);

negative\_segments\_x[index \* 2 + 1].color = Color::Black;

negative\_segments\_x[index \* 2 + 1].position = Vector2f(x0 - move \* index, y0 + 10);

}

// Створення одиничних відрізків на позитивній області віссі ординат

VertexArray positive\_segments\_y(Lines, number\_of\_segments\_y \* 2);

for (index = 0; index < number\_of\_segments\_y; index++) {

positive\_segments\_y[index \* 2].color = Color::Black;

positive\_segments\_y[index \* 2].position = Vector2f(x0 - 10, y0 + move \* index);

positive\_segments\_y[index \* 2 + 1].color = Color::Black;

positive\_segments\_y[index \* 2 + 1].position = Vector2f(x0 + 10, y0 + move \* index);

}

// Створення одиничних відрізків на негативній області віссі ординат

VertexArray negative\_segments\_y(Lines, number\_of\_segments\_y \* 2);

for (index = 0; index < number\_of\_segments\_y; index++) {

negative\_segments\_y[index \* 2].color = Color::Black;

negative\_segments\_y[index \* 2].position = Vector2f(x0 - 10, y0 - move \* index);

negative\_segments\_y[index \* 2 + 1].color = Color::Black;

negative\_segments\_y[index \* 2 + 1].position = Vector2f(x0 + 10, y0 - move \* index);

}

// Створення тексту для підпису вісей

Text x\_image("X", font, 24);

x\_image.setFillColor(Color::Black);

x\_image.setStyle(Text::Bold);

x\_image.setPosition(x0 + move\*12, y0 + move/2);

Text y\_image("Y", font, 24);

y\_image.setFillColor(Color::Black);

y\_image.setStyle(Text::Bold);

y\_image.setPosition(x0 + move/2, y0 - move\*8);

// Створення масиву із числами для відображення відстані одиничних відрізків на віссі абцис

Text number\_x[2\*number\_of\_segments\_x-1];

for (index = 0; index < 2\*number\_of\_segments\_x-1; index++) {

number\_x[index].setFont(font);

number\_x[index].setCharacterSize(12);

number\_x[index].setFillColor(Color::Black);

number\_x[index].setStyle(Text::Bold);

number\_x[index].setPosition(x0 - move \* 12 + move \* index, y0 + 10);

number\_x[index].setString(to\_string(-12 + index));

}

// Створення масиву із числами для відображення відстані одиничних відрізків на віссі ординат

Text number\_y[2\*number\_of\_segments\_y-1];

for (index = 0; index < 2\*number\_of\_segments\_y-1; index++) {

number\_y[index].setFont(font);

number\_y[index].setCharacterSize(12);

number\_y[index].setFillColor(Color::Black);

number\_y[index].setStyle(Text::Bold);

number\_y[index].setPosition(x0 + 10, y0 + move \* 8 - move \* index);

number\_y[index].setString(to\_string(-8 + index));

}

// Створення кнопок: відкриття файла, закриття файла, про програму та користуваяа, генерація

RectangleShape open\_button(Vector2f(75, 25));

open\_button.setPosition(WIDTH - 20 - open\_button.getLocalBounds().width,(head\_background.getLocalBounds().height - open\_button.getLocalBounds().height)/2);

open\_button.setFillColor(Color::Transparent);

open\_button.setFillColor(Color::White);

open\_button.setOutlineColor(Color::Black);

open\_button.setOutlineThickness(2);

open\_button.setFillColor(Color::White);

RectangleShape save\_button(Vector2f(75, 25));

save\_button.setPosition(WIDTH - 30 - open\_button.getLocalBounds().width - save\_button.getLocalBounds().width,(head\_background.getLocalBounds().height - save\_button.getLocalBounds().height)/2);

save\_button.setFillColor(Color::Transparent);

save\_button.setFillColor(Color::White);

save\_button.setOutlineColor(Color::Black);

save\_button.setOutlineThickness(2);

save\_button.setFillColor(Color::White);

RectangleShape about\_button(Vector2f(75, 25));

about\_button.setPosition(20,(head\_background.getLocalBounds().height - about\_button.getLocalBounds().height)/2);

about\_button.setFillColor(Color::Transparent);

about\_button.setFillColor(Color::White);

about\_button.setOutlineColor(Color::Black);

about\_button.setOutlineThickness(2);

about\_button.setFillColor(Color::White);

RectangleShape generate\_button(Vector2f(150, 50));

generate\_button.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + (USER\_INPUT\_WIDTH - generate\_button.getLocalBounds().width)/2, 154.5 + user\_hint.getLocalBounds().height + user\_hint\_A.getLocalBounds().height + user\_hint\_B.getLocalBounds().height + user\_hint\_N.getLocalBounds().height);

generate\_button.setFillColor(Color::Transparent);

generate\_button.setFillColor(Color::White);

generate\_button.setOutlineColor(Color::Black);

generate\_button.setOutlineThickness(2);

generate\_button.setFillColor(Color::White);

// Створення назви кнопок: відкриття файла, закриття файла, про програму та користуваяа, генерація

Text open\_button\_text("Open", font, 24);

open\_button\_text.setFillColor(Color::Black);

open\_button\_text.setOrigin(open\_button\_text.getLocalBounds().left + open\_button\_text.getLocalBounds().width / 2, open\_button\_text.getLocalBounds().top + open\_button\_text.getLocalBounds().height / 2);

open\_button\_text.setPosition(open\_button.getPosition().x + open\_button.getSize().x / 2, open\_button.getPosition().y + open\_button.getSize().y / 2);

Text save\_button\_text("Save", font, 24);

save\_button\_text.setFillColor(Color::Black);

save\_button\_text.setOrigin(save\_button\_text.getLocalBounds().left + save\_button\_text.getLocalBounds().width / 2, save\_button\_text.getLocalBounds().top + save\_button\_text.getLocalBounds().height / 2);

save\_button\_text.setPosition(save\_button.getPosition().x + save\_button.getSize().x / 2, save\_button.getPosition().y + save\_button.getSize().y / 2);

Text about\_button\_text("About", font, 24);

about\_button\_text.setFillColor(Color::Black);

about\_button\_text.setOrigin(about\_button\_text.getLocalBounds().left + about\_button\_text.getLocalBounds().width / 2, about\_button\_text.getLocalBounds().top + about\_button\_text.getLocalBounds().height / 2);

about\_button\_text.setPosition(about\_button.getPosition().x + about\_button.getSize().x / 2, about\_button.getPosition().y + about\_button.getSize().y / 2);

Text generate\_button\_text("Generate", font, 24);

generate\_button\_text.setFillColor(Color::Black);

generate\_button\_text.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + (USER\_INPUT\_WIDTH - generate\_button.getLocalBounds().width)/2 + (generate\_button.getLocalBounds().width - generate\_button\_text.getLocalBounds().width)/2, 144.5 + user\_hint.getLocalBounds().height + user\_hint\_A.getLocalBounds().height + user\_hint\_B.getLocalBounds().height + user\_hint\_N.getLocalBounds().height + (generate\_button.getLocalBounds().height - generate\_button\_text.getLocalBounds().height)/2);

// Створення текста підказки: вибора функцій, колір кожної функції

Text user\_hint\_F\_x("Functions", font, 24);

user\_hint\_F\_x.setFillColor(Color::Black);

user\_hint\_F\_x.setStyle(Text::Bold);

user\_hint\_F\_x.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + (USER\_INPUT\_WIDTH - user\_hint\_F\_x.getLocalBounds().width)/2, 345 - 1\*50);

Text user\_hint\_F("y = F(x) - Red", font, 24);

user\_hint\_F.setFillColor(Color::Black);

user\_hint\_F.setStyle(Text::Bold);

user\_hint\_F.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + (USER\_INPUT\_WIDTH - generate\_button.getLocalBounds().width)/2 + (generate\_button.getLocalBounds().width - generate\_button\_text.getLocalBounds().width)/2 - 100, 345 + 0\*50);

Text user\_hint\_cos("y = cos(x) - Darck Green", font, 24);

user\_hint\_cos.setFillColor(Color::Black);

user\_hint\_cos.setStyle(Text::Bold);

user\_hint\_cos.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + (USER\_INPUT\_WIDTH - generate\_button.getLocalBounds().width)/2 + (generate\_button.getLocalBounds().width - generate\_button\_text.getLocalBounds().width)/2 - 100, 345 + 1\*50);

Text user\_hint\_sin("y = sin(x) - Blue", font, 24);

user\_hint\_sin.setFillColor(Color::Black);

user\_hint\_sin.setStyle(Text::Bold);

user\_hint\_sin.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + (USER\_INPUT\_WIDTH - generate\_button.getLocalBounds().width)/2 + (generate\_button.getLocalBounds().width - generate\_button\_text.getLocalBounds().width)/2 - 100, 345 + 2\*50);

Text user\_hint\_rand("y = rand - Darck Yellow", font, 24);

user\_hint\_rand.setFillColor(Color::Black);

user\_hint\_rand.setStyle(Text::Bold);

user\_hint\_rand.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + (USER\_INPUT\_WIDTH - generate\_button.getLocalBounds().width)/2 + (generate\_button.getLocalBounds().width - generate\_button\_text.getLocalBounds().width)/2 - 100, 345 + 3\*50);

// Створення обʼєкта для відображення проміжків рандомної функції

Text text\_display\_written("", font, 20);

text\_display\_written.setFillColor(Color::Black);

text\_display\_written.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 +70+ 100 + (USER\_INPUT\_WIDTH - generate\_button.getLocalBounds().width)/2 + (generate\_button.getLocalBounds().width - generate\_button\_text.getLocalBounds().width)/2,345 + 3\*50);

// Створення підказок, що має ввести користувач

Text user\_hint\_F\_x\_move("Move and compress", font, 24);

user\_hint\_F\_x\_move.setFillColor(Color::Black);

user\_hint\_F\_x\_move.setStyle(Text::Bold);

user\_hint\_F\_x\_move.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + (USER\_INPUT\_WIDTH - user\_hint\_F\_x\_move.getLocalBounds().width)/2, 345 + 4\*50);

Text user\_hint\_k("k", font, 24);

user\_hint\_k.setFillColor(Color::Black);

user\_hint\_k.setStyle(Text::Bold);

user\_hint\_k.setPosition( FUNCTION\_GRAPH\_WIDTH + 20 + 5\*user\_hint\_k.getLocalBounds().width, 345 + 5\*50 + user\_hint.getLocalBounds().height -24);

Text user\_hint\_K("K", font, 24);

user\_hint\_K.setFillColor(Color::Black);

user\_hint\_K.setStyle(Text::Bold);

user\_hint\_K.setPosition( FUNCTION\_GRAPH\_WIDTH + 20 + 5\*user\_hint\_k.getLocalBounds().width, 345 + 8\*50 + user\_hint.getLocalBounds().height -7-10);

// Створення білих прямокутників із чорною оболочкою, для візуального розуміння де буде текст та куди клікати, щоб його написати

RectangleShape user\_k\_wright(Vector2f(15 \* user\_hint\_k.getLocalBounds().width, user\_hint\_k.getLocalBounds().height));

user\_k\_wright.setFillColor(Color::Transparent);

user\_k\_wright.setFillColor(Color::White);

user\_k\_wright.setOutlineColor(Color::Black);

user\_k\_wright.setOutlineThickness(2);

user\_k\_wright.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + 7\*user\_hint\_k.getLocalBounds().width, 345 + 5\*50 + user\_hint.getLocalBounds().height-17);

RectangleShape user\_K\_wright(Vector2f(15 \* user\_hint\_k.getLocalBounds().width, user\_hint\_K.getLocalBounds().height));

user\_K\_wright.setFillColor(Color::Transparent);

user\_K\_wright.setFillColor(Color::White);

user\_K\_wright.setOutlineColor(Color::Black);

user\_K\_wright.setOutlineThickness(2);

user\_K\_wright.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + 7\*user\_hint\_k.getLocalBounds().width, 345 + 8\*50 + user\_hint.getLocalBounds().height-10);

// Створення обʼєктів для відображення введеного текста

Text text\_display\_written\_k("", font, 24);

text\_display\_written\_k.setFillColor(Color::Black);

text\_display\_written\_k.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + 7\*user\_hint\_k.getLocalBounds().width, 345 + 5\*50 + user\_hint.getLocalBounds().height-24);

Text text\_display\_written\_K("", font, 24);

text\_display\_written\_K.setFillColor(Color::Black);

text\_display\_written\_K.setPosition(FUNCTION\_GRAPH\_WIDTH + 20 + 7\*user\_hint\_k.getLocalBounds().width, 345 + 8\*50 + user\_hint.getLocalBounds().height-7-10);

// Створення підказок для розуміння дій кнопок

Text user\_hint\_move\_x("Move by k units along the x-axis", font, 20);

user\_hint\_move\_x.setFillColor(Color::Black);

user\_hint\_move\_x.setStyle(Text::Bold);

user\_hint\_move\_x.setPosition(FUNCTION\_GRAPH\_WIDTH + 80, 345+ 5 + 6\*50);

Text user\_hint\_move\_y("Move by k units along the y-axis", font, 20);

user\_hint\_move\_y.setFillColor(Color::Black);

user\_hint\_move\_y.setStyle(Text::Bold);

user\_hint\_move\_y.setPosition(FUNCTION\_GRAPH\_WIDTH + 80, 345+5 + 7\*50);

Text user\_hint\_shrink\_x("Shrink by K units along the x-axis", font, 20);

user\_hint\_shrink\_x.setFillColor(Color::Black);

user\_hint\_shrink\_x.setStyle(Text::Bold);

user\_hint\_shrink\_x.setPosition(FUNCTION\_GRAPH\_WIDTH + 80, 345+ 6 + 9\*50);

Text user\_hint\_shrink\_y("Shrink by K units along the y-axis", font, 20);

user\_hint\_shrink\_y.setFillColor(Color::Black);

user\_hint\_shrink\_y.setStyle(Text::Bold);

user\_hint\_shrink\_y.setPosition(FUNCTION\_GRAPH\_WIDTH + 80, 345+6+ 10\*50);

// Відкриття головного вікна

while (window.isOpen()) {

// Створення поії

Event event;

// Обробка події

while (window.pollEvent(event)){

// Опрацювання кнопки закриття

if (event.type == Event::Closed)

window.close();

// Перевіряємо, чи відбулось натискання лівої кнопки миші

if (event.type == Event::MouseButtonPressed && event.mouseButton.button == Mouse::Left) {

Vector2i mousePosition = Mouse::getPosition(window);

Vector2f mousePositionF(mousePosition.x, mousePosition.y);

// Якщо мишка клікнула в області вводу для значення A

if (user\_A\_wright.getGlobalBounds().contains(mousePositionF)) {

click\_A = true;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області вводу для значення В

if (user\_B\_wright.getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = true;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області вводу для значення N

if (user\_N\_wright.getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = true;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області вводу для значення k

if (user\_k\_wright.getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = true;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області вводу для значення K

if (user\_K\_wright.getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = true;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області кнопки генерації

if (generate\_button.getGlobalBounds().contains(mousePositionF) || generate\_button\_text.getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = true;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області кнопки збереження

if (save\_button.getGlobalBounds().contains(mousePositionF) || save\_button\_text.getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = true;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області кнопки відкриття

if (open\_button.getGlobalBounds().contains(mousePositionF) || open\_button\_text.getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = true;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області кнопки про програму та розробника

if (about\_button.getGlobalBounds().contains(mousePositionF) || about\_button\_text.getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = true;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області кнопки для відображення індивідуальної функції

if (f\_x.get\_button().getGlobalBounds().contains(mousePositionF) || f\_x.get\_button\_text().getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(true);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області кнопки для відображення функції косинусів

if (f\_cos\_x.get\_button().getGlobalBounds().contains(mousePositionF) || f\_cos\_x.get\_button\_text().getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(true);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області кнопки для відображення функції сінусів

if (f\_sin\_x.get\_button().getGlobalBounds().contains(mousePositionF) || f\_sin\_x.get\_button\_text().getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(true);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області кнопки для відображення випадкової функції

if (f\_rand\_x.get\_button().getGlobalBounds().contains(mousePositionF) || f\_rand\_x.get\_button\_text().getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(true);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області кнопки для зсуву функції по віссі абцис

if (move\_x.get\_button().getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(true);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області кнопки для зсуву функції по віссі ординат

if (move\_y.get\_button().getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(true);

compress\_x.setClicked(false);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області кнопки для стиснення функції по віссі абцис

if (compress\_x.get\_button().getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(true);

compress\_y.setClicked(false);

}

// Якщо мишка клікнула в області кнопки для стиснення функції по віссі ординат

if (compress\_y.get\_button().getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_N = false;

click\_k = false;

click\_K = false;

click\_generate\_button = false;

click\_save\_button = false;

click\_open\_button = false;

click\_about\_button = false;

f\_x.setClicked(false);

f\_cos\_x.setClicked(false);

f\_sin\_x.setClicked(false);

f\_rand\_x.setClicked(false);

move\_x.setClicked(false);

move\_y.setClicked(false);

compress\_x.setClicked(false);

compress\_y.setClicked(true);

}

}

// Перевірка, чи подія є введенням тексту

if (event.type == Event::TextEntered) {

// Якщо натиснуто в області А то можна змінювати значення А, графіки закриваються для візуального перегляду

if(click\_A){ text\_input\_for\_A\_B\_k(event, text\_assistant\_A, text\_display\_written\_A, number\_A); F\_x\_.show\_function(false); F\_cos\_.show\_function(false); F\_sin\_.show\_function(false); F\_rand\_.show\_function(false); click\_k = false; click\_K = false; }

// Якщо натиснуто в області В то можна змінювати значення В, графіки закриваються для візуального перегляду

if(click\_B){ text\_input\_for\_A\_B\_k(event, text\_assistant\_B, text\_display\_written\_B, number\_B); F\_x\_.show\_function(false); F\_cos\_.show\_function(false); F\_sin\_.show\_function(false); F\_rand\_.show\_function(false); click\_k = false; click\_K = false;}

// Якщо натиснуто в області N то можна змінювати значення N, графіки закриваються для візуального перегляду

if(click\_N){ text\_input\_for\_N(event, text\_assistant\_N, text\_display\_written\_N, number\_N); F\_x\_.show\_function(false); F\_cos\_.show\_function(false); F\_sin\_.show\_function(false); F\_rand\_.show\_function(false); click\_k = false; click\_K = false;}

// Якщо натиснуто в області k то можна змінювати значення k

if(click\_k) text\_input\_for\_A\_B\_k(event, text\_assistant\_k, text\_display\_written\_k, number\_k);

// Якщо натиснуто в області K то можна змінювати значення K

if(click\_K) text\_input\_for\_K(event, text\_assistant\_K, text\_display\_written\_K, number\_K);

}

// Якщо була натиснула кнопка збереження

if(click\_save\_button){

string name\_file;

// Отримаємо введений файл та прибираємо всі символи, які відповідають за перехід на ноий рядкок

input\_file(font, name\_file);

name\_file.erase(remove(name\_file.begin(), name\_file.end(), '\n'), name\_file.end());

// Якщо назва файла не порожня то працюємо з нею

if(name\_file.empty()){}

else{

name\_file += ".txt";

ofstream fout;

fout.open(name\_file);

if(F\_x\_.get\_n()==0){fout << "0 ";}

else{

// Якщо файл не відкрився, то виводимо вікно про помилку

if(!fout.is\_open()){program\_does\_not\_save(font, name\_file);}

else{

int size = F\_x\_.get\_n();

fout << size << " " << text\_assistant\_A << " " << text\_assistant\_B << " ";

for(int i=0; i<size; i++) fout << F\_x\_.get\_point\_coordinates\_x(i) << " " << F\_x\_.get\_point\_coordinates\_y(i) << " ";

for(int i=0; i<size; i++) fout << F\_cos\_.get\_point\_coordinates\_x(i) << " " << F\_cos\_.get\_point\_coordinates\_y(i) << " ";

for(int i=0; i<size; i++) fout << F\_sin\_.get\_point\_coordinates\_x(i) << " " << F\_sin\_.get\_point\_coordinates\_y(i) << " ";

}

}

if(F\_rand\_.get\_n()==0){fout << "0 ";}

else{

if(!fout.is\_open()){}

else{

int size = F\_rand\_.get\_n();

fout << size << " " << text\_assistant\_A2 << " " << text\_assistant\_B2 << " ";

for(int i=0; i<size; i++) fout << F\_rand\_.get\_point\_coordinates\_x(i) << " " << F\_rand\_.get\_point\_coordinates\_y(i) << " ";

}

}

// Вивід вікна про збереження даних у файлі

program\_save(font, name\_file);

fout.close();

}

click\_save\_button = false;

}

// Якщо була натиснула кнопка відкриття

if (click\_open\_button) {

string name\_file;

// Отримаємо введений файл та прибираємо всі символи, які відповідають за перехід на ноий рядкок

output\_file(font, name\_file);

name\_file.erase(remove(name\_file.begin(), name\_file.end(), '\n'), name\_file.end());

// Якщо назва файла не порожня то працюємо з нею

if (name\_file.empty()) {}

else {

name\_file += ".txt";

ifstream fin;

fin.open(name\_file);

// Якщо файл не відкрився, то виводимо вікно про помилку

if (!fin.is\_open()) {program\_does\_not\_open(font, name\_file);}

else {

int size;

double number;

fin >> size;

number\_N = size;

if(size<=0){

}

else

{

F\_x\_.great\_points(size);

F\_cos\_.great\_points(size);

F\_sin\_.great\_points(size);

fin >> text\_assistant\_A >> text\_assistant\_B;

text\_display\_written\_A.setString(text\_assistant\_A);

text\_display\_written\_B.setString(text\_assistant\_B);

text\_assistant\_N = to\_string(size);

text\_display\_written\_N.setString(text\_assistant\_N);

number\_A = stod(text\_assistant\_A);

number\_B = stod(text\_assistant\_B);

for(int i=0; i<size; i++)

{

fin >> number;

F\_x\_.set\_point\_coordinates\_x(i, number);

fin >> number;

if(number<pow(10,-15)) number = pow(10,-5);

F\_x\_.set\_point\_coordinates\_y(i, number);

}

for(int i=0; i<size; i++)

{

fin >> number;

F\_cos\_.set\_point\_coordinates\_x(i, number);

fin >> number;

F\_cos\_.set\_point\_coordinates\_y(i, number);

}

for(int i=0; i<size; i++)

{

fin >> number;

F\_sin\_.set\_point\_coordinates\_x(i, number);

fin >> number;

F\_sin\_.set\_point\_coordinates\_y(i, number);

}

F\_x\_.show\_function(true);

F\_cos\_.show\_function(true);

F\_sin\_.show\_function(true);

F\_x\_.set\_n(size);

F\_sin\_.set\_n(size);

F\_cos\_.set\_n(size);

}

fin >> size;

if(size<=0){}

else

{

F\_rand\_.great\_points(size);

fin >> text\_assistant\_A2 >> text\_assistant\_B2;

number\_A2 = stod(text\_assistant\_A2);

number\_B2 = stod(text\_assistant\_B2);

for(int i=0; i<size; i++)

{

fin >> number;

F\_rand\_.set\_point\_coordinates\_x(i, number);

fin >> number;

F\_rand\_.set\_point\_coordinates\_y(i, number);

}

F\_rand\_.show\_function(true);

F\_rand\_.set\_n(size);

ostringstream oss;

oss << "[" << text\_assistant\_A2 << "; " << text\_assistant\_B2 << "]";

string result = oss.str();

text\_display\_written.setString(result);

}

// Вивід вікна про відкриття файлу з даними

program\_open(font, name\_file);

}

}

click\_open\_button = false;

}

// Якщо була натиснула кнопка про прогграму та користувача то виводимо відповідне вікно з інформацією

if (click\_about\_button){

job\_information(font);

click\_about\_button = false;

}

// Якщо була натиснута кнопка генерації то створюємо відповідні графікі

if(click\_generate\_button)

{

// Якщо щось не введено - виводимо помилку

if(text\_assistant\_A.empty() || text\_assistant\_B.empty() || text\_assistant\_N.empty() || number\_N <= 0 || number\_N > 100000 ) A\_B\_N\_were\_not\_entered(font);

// Якщо початкове значення більше за кінцеве - змінюємо місцями та виводимо повідомлення

if(number\_A > number\_B){

the\_beginning\_is\_greater\_than\_the\_end(font);

swap(number\_A,number\_B);

String temp = text\_display\_written\_A.getString();

String temp2 = text\_display\_written\_B.getString();

text\_assistant\_A = text\_display\_written\_B.getString();

text\_assistant\_B = text\_display\_written\_A.getString();

text\_display\_written\_A.setString(temp2);

text\_display\_written\_B.setString(temp);

}

F\_x\_.f\_x(number\_A, number\_B, number\_N, move, 2);

F\_cos\_.f\_x(number\_A, number\_B, number\_N, move, 2);

F\_sin\_.f\_x(number\_A, number\_B, number\_N, move, 2);

if(!text\_assistant\_A2.empty() && !text\_assistant\_B2.empty()) F\_rand\_.f\_x(number\_A2, number\_B2, number\_N, move, 2);

click\_generate\_button = false;

}

// Якщо натиснута кнопка відображення графіка то можна його відображати, якщо вона була вимкнута, а можна не відображати в проти лежному випадку - якщо вона відображала графік

if(f\_x.isClicked())

{

// Якщо щось не введено або змінено кількість точок і не сгенеровано функції- виводимо помилку

if(F\_x\_.get\_n()!=number\_N || text\_assistant\_A.empty() || text\_assistant\_B.empty() || text\_assistant\_N.empty() || number\_N <= 0 || number\_N > 100000) A\_B\_N\_were\_not\_entered(font);

else{

if(F\_x\_.show\_function()) F\_x\_.show\_function(false);

else F\_x\_.show\_function(true);

}

f\_x.setClicked(false);

}

// Якщо натиснута кнопка відображення графіка то можна його відображати, якщо вона була вимкнута, а можна не відображати в проти лежному випадку - якщо вона відображала графік

if(f\_cos\_x.isClicked())

{

// Якщо щось не введено або змінено кількість точок і не сгенеровано функції- виводимо помилку

if(F\_cos\_.get\_n()!=number\_N || text\_assistant\_A.empty() || text\_assistant\_B.empty() || text\_assistant\_N.empty() || number\_N <= 0 || number\_N > 100000) A\_B\_N\_were\_not\_entered(font);

else{

if(F\_cos\_.show\_function()) F\_cos\_.show\_function(false);

else F\_cos\_.show\_function(true);

}

f\_cos\_x.setClicked(false);

}

// Якщо натиснута кнопка відображення графіка то можна його відображати, якщо вона була вимкнута, а можна не відображати в проти лежному випадку - якщо вона відображала графік

if(f\_sin\_x.isClicked())

{

// Якщо щось не введено або змінено кількість точок і не сгенеровано функції- виводимо помилку

if(F\_sin\_.get\_n()!=number\_N || text\_assistant\_A.empty() || text\_assistant\_B.empty() || text\_assistant\_N.empty() || number\_N <= 0 || number\_N > 100000) A\_B\_N\_were\_not\_entered(font);

else{

if(F\_sin\_.show\_function()) F\_sin\_.show\_function(false);

else F\_sin\_.show\_function(true);

}

f\_sin\_x.setClicked(false);

}

// Якщо натиснута кнопка відображення графіка то можна його відображати, якщо вона була вимкнута, а можна не відображати в проти лежному випадку - якщо вона відображала графік. А також змінити область х, яка виводиться для випадкової функції

if(f\_rand\_x.isClicked())

{

// Якщо щось не введено або змінено кількість точок і не сгенеровано функції- виводимо помилку

if(F\_x\_.get\_n()!=number\_N || text\_assistant\_N.empty() || number\_N <= 0 || number\_N > 100000) A\_B\_N\_were\_not\_entered(font);

else{

if(F\_rand\_.show\_function()) F\_rand\_.show\_function(false);

else F\_rand\_.show\_function(true);

f\_rand\_x.setClicked(false);

if(F\_rand\_.show\_function()){

Random\_function\_settings\_A\_B(font, text\_assistant\_A2, text\_assistant\_B2);

if(text\_assistant\_A2.empty() || text\_assistant\_B2.empty())

{

while (!text\_assistant\_A2.empty()) text\_assistant\_A2.pop\_back();

while (!text\_assistant\_B2.empty()) text\_assistant\_B2.pop\_back();

}

if(!text\_assistant\_A2.empty() && !text\_assistant\_B2.empty())

{

number\_A2 = stod(text\_assistant\_A2);

number\_B2 = stod(text\_assistant\_B2);

F\_rand\_.f\_x(number\_A2, number\_B2, number\_N, move, 2);

}

}

ostringstream oss;

oss << "[" << text\_assistant\_A2 << "; " << text\_assistant\_B2 << "]";

string result = oss.str();

text\_display\_written.setString(result);

}

f\_rand\_x.setClicked(false);

}

// Якщо клікнута кнопка зсуву по х і вибрані графіки - перемістити храфіки по осі абцис на певну кількість одиниць

if(move\_x.isClicked()){

if(text\_assistant\_k.empty()) zero\_coefficient(font);

else

{

if(F\_x\_.show\_function()) F\_x\_.move\_x(number\_k, number\_N, move);

if(F\_cos\_.show\_function()) F\_cos\_.move\_x(number\_k, number\_N, move);

if(F\_sin\_.show\_function()) F\_sin\_.move\_x(number\_k, number\_N, move);

if(F\_rand\_.show\_function()){

F\_rand\_.move\_x(number\_k, number\_N, move);

number\_A2 += number\_k;

number\_B2 += number\_k;

text\_assistant\_A2 = to\_string(number\_A2);

text\_assistant\_B2 = to\_string(number\_B2);

text\_assistant\_A2 = trimFloatString(text\_assistant\_A2);

text\_assistant\_B2 = trimFloatString(text\_assistant\_B2);

ostringstream oss;

oss << "[" << text\_assistant\_A2 << "; " << text\_assistant\_B2 << "]";

string result = oss.str();

text\_display\_written.setString(result);

}

if(!F\_x\_.show\_function()&&!F\_cos\_.show\_function()&&!F\_sin\_.show\_function()&&!F\_rand\_.show\_function() || number\_N==0) there\_is\_a\_coefficient\_but\_there\_is\_no\_graph(font);

}

move\_x.setClicked(false);

}

// Якщо клікнута кнопка зсуву по у і вибрані графіки - перемістити храфіки по осі ординат на певну кількість одиниць

if(move\_y.isClicked()){

if(text\_assistant\_k.empty()) zero\_coefficient(font);

else

{

if(F\_x\_.show\_function()) F\_x\_.move\_y(number\_k, number\_N, move);

if(F\_cos\_.show\_function()) F\_cos\_.move\_y(number\_k, number\_N, move);

if(F\_sin\_.show\_function()) F\_sin\_.move\_y(number\_k, number\_N, move);

if(F\_rand\_.show\_function()) F\_rand\_.move\_y(number\_k, number\_N, move);

if(!F\_x\_.show\_function()&&!F\_cos\_.show\_function()&&!F\_sin\_.show\_function()&&!F\_rand\_.show\_function() || number\_N==0) there\_is\_a\_coefficient\_but\_there\_is\_no\_graph(font);

}

move\_y.setClicked(false);

}

// Якщо клікнута кнопка стиснення по х і вибрані графіки - стиснути храфіки по осі абцис на певний коефіцієнт. А також змінити область х, яка виводиться для випадкової функції

if(compress\_x.isClicked()){

if(text\_assistant\_K.empty()) zero\_coefficient(font);

else

{

if(F\_x\_.show\_function()) F\_x\_.compress\_x(number\_K, number\_N);

if(F\_cos\_.show\_function()) F\_cos\_.compress\_x(number\_K, number\_N);

if(F\_sin\_.show\_function()) F\_sin\_.compress\_x(number\_K, number\_N);

if(F\_rand\_.show\_function()){

F\_rand\_.compress\_x(number\_K, number\_N);

number\_A2 \*= number\_K;

number\_B2 \*= number\_K;

text\_assistant\_A2 = to\_string(number\_A2);

text\_assistant\_B2 = to\_string(number\_B2);

text\_assistant\_A2 = trimFloatString(text\_assistant\_A2);

text\_assistant\_B2 = trimFloatString(text\_assistant\_B2);

ostringstream oss;

oss << "[" << text\_assistant\_A2 << "; " << text\_assistant\_B2 << "]";

string result = oss.str();

text\_display\_written.setString(result);

}

if(!F\_x\_.show\_function()&&!F\_cos\_.show\_function()&&!F\_sin\_.show\_function()&&!F\_rand\_.show\_function() || number\_N==0) there\_is\_a\_coefficient\_but\_there\_is\_no\_graph(font);

}

compress\_x.setClicked(false);

}

// Якщо клікнута кнопка стиснення по у і вибрані графіки - стиснути храфіки по осі ординат на певний коефіцієнт

if(compress\_y.isClicked()){

if(text\_assistant\_K.empty()) zero\_coefficient(font);

else

{

if(F\_x\_.show\_function()) F\_x\_.compress\_y(number\_K, number\_N);

if(F\_cos\_.show\_function()) F\_cos\_.compress\_y(number\_K, number\_N);

if(F\_sin\_.show\_function()) F\_sin\_.compress\_y(number\_K, number\_N);

if(F\_rand\_.show\_function()) F\_rand\_.compress\_y(number\_K, number\_N);

if(!F\_x\_.show\_function()&&!F\_cos\_.show\_function()&&!F\_sin\_.show\_function()&&!F\_rand\_.show\_function() || number\_N==0) there\_is\_a\_coefficient\_but\_there\_is\_no\_graph(font);

}

compress\_y.setClicked(false);

}

}

// Створення точок для малювання графіків

CircleShape point1;

point1.setRadius(2);

point1.setFillColor(Color::Red);

CircleShape point2;

point2.setRadius(2);

point2.setFillColor(Color(0,100,0));

CircleShape point3;

point3.setRadius(2);

point3.setFillColor(Color::Blue);

CircleShape point4;

point4.setRadius(2);

point4.setFillColor(Color(139,139,0));

// Відмальовка кольору фону

window.clear(Color(109,109,109));

// Відмальовка фону де буде наш графік

window.draw(graph\_background);

// Відмаловка осі х із стрілками

window.draw(Ox);

window.draw(right\_side\_of\_the\_arrow\_x);

window.draw(left\_side\_of\_the\_arrow\_x);

// Відмаловка осі у із стрілками

window.draw(Oy);

window.draw(right\_side\_of\_the\_arrow\_y);

window.draw(left\_side\_of\_the\_arrow\_y);

// Відмаловка осі підписів осей

window.draw(x\_image);

window.draw(y\_image);

// Відмальовка міток осей

for (index = 0; index < 2\*number\_of\_segments\_x-1; index++)

{

window.draw(number\_x[index]);

if(index<=16 && index!= 8)

window.draw(number\_y[index]);

}

// Відмальовка одиничних відрізків

window.draw(positive\_segments\_x);

window.draw(negative\_segments\_x);

window.draw(positive\_segments\_y);

window.draw(negative\_segments\_y);

// Відмальовка індивідуального графіка, якщо він вибраний

if(F\_x\_.show\_function()){

for (index = 0; index < number\_N; index++){

point1.setPosition(F\_x\_.get\_point\_coordinates\_x(index),F\_x\_.get\_point\_coordinates\_y(index));

window.draw(point1);

}

}

// Відмальовка графіка косинуса, якщо він вибраний

if(F\_cos\_.show\_function()){

for (index = 0; index < number\_N; index++){

point2.setPosition(F\_cos\_.get\_point\_coordinates\_x(index),F\_cos\_.get\_point\_coordinates\_y(index));

window.draw(point2);

}

}

// Відмальовка графіка сінуса, якщо він вибраний

if(F\_sin\_.show\_function()){

for (index = 0; index < number\_N; index++){

point3.setPosition(F\_sin\_.get\_point\_coordinates\_x(index),F\_sin\_.get\_point\_coordinates\_y(index));

window.draw(point3);

}

}

// Відмальовка випадкового графіка, якщо він вибраний

if (F\_rand\_.show\_function() && !text\_assistant\_A2.empty() && !text\_assistant\_B2.empty()) {

if(number\_N==1)

{

point4.setPosition(F\_rand\_.get\_point\_coordinates\_x(0),F\_rand\_.get\_point\_coordinates\_y(0));

point4.setFillColor(Color(184, 134, 11));

window.draw(point4);

}

else{

Vertex line[number\_N];

for (int i = 0; i < number\_N - 1; i++) {

line[i].position = Vector2f(F\_rand\_.get\_point\_coordinates\_x(i),F\_rand\_.get\_point\_coordinates\_y(i));

line[i].color = Color(184, 134, 11);

line[i + 1].position = Vector2f(F\_rand\_.get\_point\_coordinates\_x(i+1),F\_rand\_.get\_point\_coordinates\_y(i+1));

line[i + 1].color = Color(184, 134, 11);

}

window.draw(line, number\_N, LinesStrip);

}

}

// Відмальовка поля меню та шапки

window.draw(user\_background);

window.draw(head\_background);

// Відмальовка підказок для введення, для чого тескт

window.draw(user\_hint);

window.draw(user\_hint\_A);

window.draw(user\_hint\_B);

window.draw(user\_hint\_N);

window.draw(user\_hint\_k);

window.draw(user\_hint\_K);

// Відмальовка підказок для натискання кнопок

window.draw(user\_hint\_F\_x);

window.draw(user\_hint\_F);

window.draw(user\_hint\_cos);

window.draw(user\_hint\_sin);

window.draw(user\_hint\_rand);

window.draw(user\_hint\_move\_x);

window.draw(user\_hint\_move\_y);

window.draw(user\_hint\_shrink\_x);

window.draw(user\_hint\_shrink\_y);

window.draw(user\_hint\_F\_x\_move);

// Відмальовка підказок для введення, де саме буде текст

window.draw(user\_A\_wright);

window.draw(user\_B\_wright);

window.draw(user\_N\_wright);

window.draw(user\_k\_wright);

window.draw(user\_K\_wright);

// Відмальовка написаного користувачем значень

window.draw(text\_display\_written\_A);

window.draw(text\_display\_written\_B);

window.draw(text\_display\_written\_N);

window.draw(text\_display\_written\_k);

window.draw(text\_display\_written\_K);

// відмальовка кнопок

window.draw(about\_button);

window.draw(open\_button);

window.draw(save\_button);

window.draw(generate\_button);

window.draw(f\_x.get\_button());

window.draw(f\_cos\_x.get\_button());

window.draw(f\_sin\_x.get\_button());

window.draw(f\_rand\_x.get\_button());

window.draw(move\_x.get\_button());

window.draw(move\_y.get\_button());

window.draw(compress\_x.get\_button());

window.draw(compress\_y.get\_button());

// Відмальовка текстів до кнопок

window.draw(about\_button\_text);

window.draw(open\_button\_text);

window.draw(save\_button\_text);

window.draw(generate\_button\_text);

// Відмальовка позначки що графік вибраний

if(F\_x\_.show\_function()) window.draw(f\_x.get\_button\_text());

if(F\_cos\_.show\_function()) window.draw(f\_cos\_x.get\_button\_text());

if(F\_sin\_.show\_function()) window.draw(f\_sin\_x.get\_button\_text());

if(F\_rand\_.show\_function() && F\_rand\_.show\_function() && !text\_assistant\_A2.empty() && !text\_assistant\_B2.empty() ) window.draw(f\_rand\_x.get\_button\_text());

// Відмальовка проміжка х для випадкової функції, якщо він є

if(!text\_assistant\_A2.empty()) window.draw(text\_display\_written);

window.display();

}

return 0;

}

# Додаток Б

Хедр файл Files.h

#ifndef Files

#define Files

// Підключення стандартноі бібліотеки для вводу/виводу та простору імен std

#include <iostream>

using namespace std;

// Підключення бібліотеки sfml для вграфіки та простору імен sf

#include <SFML/Graphics.hpp>

using namespace sf;

// Підключення бібліотеки для роботи з фалами

#include <sstream>

// Фунуція для записання дданних у файл

void input\_file(Font& font, string& str);

// Функція для надання данних із файлу

void output\_file(Font& font, string& text);

#endif

Сурс файл files.cpp

// Підключення хедр файлу Files, де написані функції, які треба визначити, надати тіло.

#include "Files.h"

// Фунуція для записання дданних у файл

void input\_file(Font& font, string& str) {

string text;

// Створення вікна

RenderWindow window(VideoMode(1700, 800),"Input file");

// Створення підказок, що має ввести користувач

Text user\_hint\_Message("Write the path and file name and it will be saved in txt format.\n\n\t\t\t\tIf the text does not fit, enter it using Enter.", font, 24);

user\_hint\_Message.setFillColor(Color::Black);

user\_hint\_Message.setPosition((1700 - user\_hint\_Message.getLocalBounds().width)/2, 2\*user\_hint\_Message.getLocalBounds().height);

// Створення білих прямокутників із чорною оболочкою візуального розуміння де буде текст та куди клікати, щоб його написати

RectangleShape user\_wright(Vector2f(1680, 3\*user\_hint\_Message.getLocalBounds().height));

user\_wright.setFillColor(Color::Transparent);

user\_wright.setFillColor(Color::White);

user\_wright.setOutlineColor(Color::Black);

user\_wright.setOutlineThickness(2);

user\_wright.setPosition(10, 5\*user\_hint\_Message.getLocalBounds().height);

// Створення обʼєктів для відображення введеного тексту

Text text\_display\_written("", font, 24);

text\_display\_written.setFillColor(Color::Black);

text\_display\_written.setPosition(10, 5\*user\_hint\_Message.getLocalBounds().height);

// Створення прямокутної кнопки

RectangleShape save\_button(Vector2f(100, user\_hint\_Message.getLocalBounds().height/2));

save\_button.setFillColor(Color::White);

save\_button.setPosition((1700 - save\_button.getLocalBounds().width)/2, 3\*user\_hint\_Message.getLocalBounds().height + text\_display\_written.getLocalBounds().height + user\_wright.getLocalBounds().height + ((3\*user\_hint\_Message.getLocalBounds().height + text\_display\_written.getLocalBounds().height + user\_wright.getLocalBounds().height) - save\_button.getLocalBounds().height)/2);

// Створення грані для кнопки

RectangleShape save\_button\_border(Vector2f(100, 50));

save\_button\_border.setFillColor(Color::Transparent);

save\_button\_border.setOutlineColor(Color::Black);

save\_button\_border.setOutlineThickness(2);

save\_button\_border.setPosition((1700 - save\_button.getLocalBounds().width)/2, 3\*user\_hint\_Message.getLocalBounds().height + text\_display\_written.getLocalBounds().height + user\_wright.getLocalBounds().height + ((3\*user\_hint\_Message.getLocalBounds().height + text\_display\_written.getLocalBounds().height + user\_wright.getLocalBounds().height) - save\_button.getLocalBounds().height)/2);

// Створення назви кнопки

Text save\_button\_text("Save", font, 24);

save\_button\_text.setFillColor(Color::Black);

save\_button\_text.setOrigin(save\_button\_text.getLocalBounds().left + save\_button\_text.getLocalBounds().width / 2, save\_button\_text.getLocalBounds().top + save\_button\_text.getLocalBounds().height / 2);

save\_button\_text.setPosition(save\_button\_border.getPosition().x + save\_button\_border.getSize().x / 2, save\_button\_border.getPosition().y + save\_button\_border.getSize().y / 2);

// Змінні для перевірки натискання

bool click\_user\_wright = false;

bool click\_save\_button = false;

while (window.isOpen()) {

Event event;

// Обробка події

while (window.pollEvent(event)) {

// Опрацювання кнопки закриття

if (event.type == Event::Closed)

window.close();

if (event.type == Event::MouseButtonPressed && event.mouseButton.button == Mouse::Left) {

Vector2i mousePosition = Mouse::getPosition(window);

Vector2f mousePositionF(mousePosition.x, mousePosition.y);

// Якщо мишка клікнула в області вводу для значення файлу

if (user\_wright.getGlobalBounds().contains(mousePositionF))

{

click\_user\_wright = true;

click\_save\_button = false;

}

// Якщо мишка клікнула в області кнопки зберігання

if (save\_button.getGlobalBounds().contains(mousePositionF) || save\_button\_border.getGlobalBounds().contains(mousePositionF) || save\_button\_border.getGlobalBounds().contains(mousePositionF))

{

click\_user\_wright = false;

click\_save\_button = true;

}

}

// Перевірка, чи подія є введенням тексту

if (event.type == Event::TextEntered) {

// Якщо вибрана область для написання назви файлу зберігання

if(click\_user\_wright)

{

// Якщо натиснута клавіша видалення елемента, то у випадку коли поле вводу не порожнє - видаляється останній елемент

if (event.text.unicode == '\b'){

if (!text.empty())

text.pop\_back();

}

// Якщо клавіша не для ввидалення - додаємо символ до строки

else text += (event.text.unicode);

text\_display\_written.setString(text);

}

}

// Якщо натиснута кнопка збереження, то строці, яка передається - передається на зовні, а вікно закривається

if(click\_save\_button)

{

str = text;

window.close();

}

window.clear(Color::White);

window.draw(user\_hint\_Message);

window.draw(user\_wright);

window.draw(text\_display\_written);

window.draw(save\_button);

window.draw(save\_button\_border);

window.draw(save\_button\_text);

window.display();

}

}

}

// Функція для надання данних із файлу

void output\_file(Font& font, string& text) {

RenderWindow window(VideoMode(1700, 800),"Output file");

// Створення вікна

Text user\_hint\_Message("Write the path and file name and it will be open.\n\n\t If the text does not fit, enter it using Enter.", font, 24);

user\_hint\_Message.setFillColor(Color::Black);

user\_hint\_Message.setPosition((1700 - user\_hint\_Message.getLocalBounds().width)/2, 2\*user\_hint\_Message.getLocalBounds().height);

// Створення підказок, що має ввести користувач

RectangleShape user\_wright(Vector2f(1680, 3\*user\_hint\_Message.getLocalBounds().height));

user\_wright.setFillColor(Color::Transparent);

user\_wright.setFillColor(Color::White);

user\_wright.setOutlineColor(Color::Black);

user\_wright.setOutlineThickness(2);

user\_wright.setPosition(10, 5\*user\_hint\_Message.getLocalBounds().height);

// Створення обʼєктів для відображення введеного тексту

Text text\_display\_written("", font, 24);

text\_display\_written.setFillColor(Color::Black);

text\_display\_written.setPosition(10, 5\*user\_hint\_Message.getLocalBounds().height);

// Створення прямокутної кнопки

RectangleShape open\_button(Vector2f(100, user\_hint\_Message.getLocalBounds().height/2));

open\_button.setFillColor(Color::White);

open\_button.setPosition((1700 - open\_button.getLocalBounds().width)/2, 3\*user\_hint\_Message.getLocalBounds().height + text\_display\_written.getLocalBounds().height + user\_wright.getLocalBounds().height + ((3\*user\_hint\_Message.getLocalBounds().height + text\_display\_written.getLocalBounds().height + user\_wright.getLocalBounds().height) - open\_button.getLocalBounds().height)/2);

// Створення грані для кнопки

RectangleShape open\_button\_border(Vector2f(100, 50));

open\_button\_border.setFillColor(Color::Transparent);

open\_button\_border.setOutlineColor(Color::Black);

open\_button\_border.setOutlineThickness(2);

open\_button\_border.setPosition((1700 - open\_button.getLocalBounds().width)/2, 3\*user\_hint\_Message.getLocalBounds().height + text\_display\_written.getLocalBounds().height + user\_wright.getLocalBounds().height + ((3\*user\_hint\_Message.getLocalBounds().height + text\_display\_written.getLocalBounds().height + user\_wright.getLocalBounds().height) - open\_button.getLocalBounds().height)/2);

// Створення назви кнопки

Text open\_button\_text("Open", font, 24);

open\_button\_text.setFillColor(Color::Black);

open\_button\_text.setOrigin(open\_button\_text.getLocalBounds().left + open\_button\_text.getLocalBounds().width / 2, open\_button\_text.getLocalBounds().top + open\_button\_text.getLocalBounds().height / 2);

open\_button\_text.setPosition(open\_button\_border.getPosition().x + open\_button\_border.getSize().x / 2, open\_button\_border.getPosition().y + open\_button\_border.getSize().y / 2);

// Змінні для перевірки натискання

bool click\_user\_wright = false;

bool click\_open\_button = false;

while (window.isOpen()) {

Event event;

// Обробка події

while (window.pollEvent(event)) {

// Опрацювання кнопки закриття

if (event.type == Event::Closed)

window.close();

if (event.type == Event::MouseButtonPressed && event.mouseButton.button == Mouse::Left) {

Vector2i mousePosition = Mouse::getPosition(window);

Vector2f mousePositionF(mousePosition.x, mousePosition.y);

// Якщо мишка клікнула в області вводу для значення файлу

if (user\_wright.getGlobalBounds().contains(mousePositionF))

{

click\_user\_wright = true;

click\_open\_button = false;

}

// Якщо мишка клікнула в області кнопки зберігання

if (open\_button.getGlobalBounds().contains(mousePositionF) || open\_button\_border.getGlobalBounds().contains(mousePositionF) || open\_button\_border.getGlobalBounds().contains(mousePositionF))

{

click\_user\_wright = false;

click\_open\_button = true;

}

}

// Перевірка, чи подія є введенням тексту

if (event.type == Event::TextEntered) {

// Якщо вибрана область для написання назви файлу зберігання

if(click\_user\_wright)

{

// Якщо натиснута клавіша видалення елемента, то у випадку коли поле вводу не порожнє - видаляється останній елемент

if (event.text.unicode == '\b'){

if (!text.empty())

text.pop\_back();

}

// Якщо клавіша не для ввидалення - додаємо символ до строки

else text += (event.text.unicode);

text\_display\_written.setString(text);

}

}

// Якщо натиснута кнопка відкриття, то вікно закривається

if(click\_open\_button)

window.close();

window.clear(Color::White);

window.draw(user\_hint\_Message);

window.draw(user\_wright);

window.draw(text\_display\_written);

window.draw(open\_button);

window.draw(open\_button\_border);

window.draw(open\_button\_text);

window.display();

}

}

}

# Додаток В

Хедр файл Error\_windows.h

#ifndef Error\_windows

#define Error\_windows

// Підключення бібліотеки sfml для вграфіки та простору імен sf

#include <SFML/Graphics.hpp>

using namespace sf;

// Вікно про нульовий коефіцієнт, який немає сенсу

void zero\_coefficient(Font& font);

// Вікно про помилку, коли людина намагається перемістити чи масштабувати неіснуюче

void there\_is\_a\_coefficient\_but\_there\_is\_no\_graph(Font& font);

// Вікно про помилку, коли початкове значення проміжка А і В більше ніж кінцеве

void the\_beginning\_is\_greater\_than\_the\_end(Font& font);

// Вікно про помилку, коли А, В чи N не введені

void A\_B\_N\_were\_not\_entered(Font& font);

#endif

Сурс файл error\_windows.cpp

// Підключення хедр файлу "Error\_windows, де написані функції, які треба визначити, надати тіло.

#include "Error\_windows.h"

// Вікно про нульовий коефіцієнт, який немає сенсу

void zero\_coefficient(Font& font){

// Створення вікна

RenderWindow window(VideoMode(500, 200), "Notification:");

// Створення тексту для виводу на екран

Text Message("I don't see the point in a zero coefficient.\n\nIf you add zero, nothing changes.\n\nIf you multiply by zero, nothing remains.", font, 24);

Message.setFillColor(Color::Black);

Message.setPosition((500 - Message.getLocalBounds().width)/2, (200 - Message.getLocalBounds().height)/2);

while (window.isOpen()) {

Event event;

// Обробка події

while (window.pollEvent(event))

// Опрацювання кнопки закриття

if (event.type == Event::Closed)

window.close();

window.clear(Color::White);

window.draw(Message);

window.display();

}

}

// Вікно про помилку, коли людина намагається перемістити чи масштабувати неіснуюче

void there\_is\_a\_coefficient\_but\_there\_is\_no\_graph(Font& font) {

// Створення вікна

RenderWindow window(VideoMode(600, 150), "Error:");

// Створення тексту для виводу на екран

Text Message("The function must be created and selected.", font, 24);

Message.setFillColor(Color::Red);

Message.setPosition((600 - Message.getLocalBounds().width)/2, (150 - Message.getLocalBounds().height)/2);

while (window.isOpen()) {

Event event;

// Обробка події

while (window.pollEvent(event))

// Опрацювання кнопки закриття

if (event.type == Event::Closed)

window.close();

window.clear(Color::White);

window.draw(Message);

window.display();

}

}

// Вікно про помилку, коли початкове значення проміжка А і В більше ніж кінцеве

void the\_beginning\_is\_greater\_than\_the\_end(Font& font) {

// Створення вікна

RenderWindow window(VideoMode(300, 150), "Error:");

// Створення тексту для виводу на екран

Text Message("A must be less than B.\n\t(A, B) = (B, A).", font, 24);

Message.setFillColor(Color::Red);

Message.setPosition((300 - Message.getLocalBounds().width)/2, (150 - Message.getLocalBounds().height)/2);

while (window.isOpen()) {

Event event;

// Обробка події

while (window.pollEvent(event))

// Опрацювання кнопки закриття

if (event.type == Event::Closed)

window.close();

window.clear(Color::White);

window.draw(Message);

window.display();

}

}

// Вікно про помилку, коли А, В чи N не введені

void A\_B\_N\_were\_not\_entered(Font& font) {

// Створення вікна

RenderWindow window(VideoMode(500, 150), "Error:");

// Створення тексту для виводу на екран

Text Message("Values entered incorrectly: A, B, N.\nA < B\n0 < N <= 100 000\nAnd firstly click a button \"Generate\"", font, 24);

Message.setFillColor(Color::Red);

Message.setPosition((500 - Message.getLocalBounds().width)/2, (150 - Message.getLocalBounds().height)/2);

while (window.isOpen()) {

Event event;

// Обробка події

while (window.pollEvent(event))

// Опрацювання кнопки закриття

if (event.type == Event::Closed)

window.close();

window.clear(Color::White);

window.draw(Message);

window.display();

}

}

# Додаток Г

Хедр файл Text\_input.h

#ifndef Text\_input

#define Text\_input

// Підключення стандартноі бібліотеки для вводу/виводу та простору імен std

#include <iostream>

using namespace std;

// Підключення бібліотеки sfml для вграфіки та простору імен sf

#include <SFML/Graphics.hpp>

using namespace sf;

// Функція для заповнення проміжків відображення функцій та коефіцієнта зсуву

void text\_input\_for\_A\_B\_k(Event& event, string& text\_assistant, Text& text\_display\_written, float& number);

// Функція для заповнення кількості точок для відображення графіка

void text\_input\_for\_N(Event& event, string& text\_assistant, Text& text\_display\_written, int& number);

// Функція для заповнення коефіцієнта стискання

void text\_input\_for\_K(Event& event, string& text\_assistant, Text& text\_display\_written, float& number);

#endif

Сурс файл text\_input.cpp

// Підключення хедр файлу Text\_input, де написані функції, які треба визначити, надати тіло.

#include "Text\_input.h"

// Функція для заповнення проміжків відображення функцій та коефіцієнта зсуву

void text\_input\_for\_A\_B\_k(Event& event, string& text\_assistant, Text& text\_display\_written, float& number)

{

// Якщо наснута операція видалення, то якщо є символи в рядку є - видаляється останній

if (event.text.unicode == '\b'){

if (!text\_assistant.empty())

text\_assistant.pop\_back();

}

else

// Можна вводити числа від 0 до 9, лише одну крапку та мінус лише першим

if ((event.text.unicode >= '0' && event.text.unicode <= '9') || (event.text.unicode == '.' && text\_assistant.find('.') == string::npos) || (event.text.unicode == '-' && text\_assistant.empty()))

text\_assistant += (event.text.unicode);

// Надання виводу тексту на єкран введеного текста

text\_display\_written.setString(text\_assistant);

// Перекидання лише 20 символів у дробову змінну

char text[21];

strncpy(text, text\_assistant.c\_str(), 20);

number = atof(text);

}

// Функція для заповнення кількості точок для відображення графіка

void text\_input\_for\_N(Event& event, string& text\_assistant, Text& text\_display\_written, int& number)

{

// Якщо наснута операція видалення, то якщо є символи в рядку є - видаляється останній

if (event.text.unicode == '\b'){

if (!text\_assistant.empty())

text\_assistant.pop\_back();

}

else

// Можна вводити числа від 0 до 9

if ((event.text.unicode >= '0' && event.text.unicode <= '9'))

text\_assistant += (event.text.unicode);

// Надання виводу тексту на єкран введеного текста

text\_display\_written.setString(text\_assistant);

// Перекидання лише 20 символів у цілу змінну

char text[21];

strncpy(text, text\_assistant.c\_str(), 20);

number = atof(text);

}

// Функція для заповнення коефіцієнта стискання

void text\_input\_for\_K(Event& event, string& text\_assistant, Text& text\_display\_written, float& number)

{

// Якщо наснута операція видалення, то якщо є символи в рядку є - видаляється останній

if (event.text.unicode == '\b'){

if (!text\_assistant.empty())

text\_assistant.pop\_back();

}

else

// Можна вводити числа від 0 до 9 та лише одну крапку

if ((event.text.unicode >= '0' && event.text.unicode <= '9') || (event.text.unicode == '.' && text\_assistant.find('.') == string::npos))

text\_assistant += (event.text.unicode);

// Надання виводу тексту на єкран введеного текста

text\_display\_written.setString(text\_assistant);

// Перекидання лише 20 символів у дробову змінну

char text[21];

strncpy(text, text\_assistant.c\_str(), 20);

number = atof(text);

}

# Додаток Ґ

Хедр файл Functions.h

#ifndef Functions

#define Functions

// Підключення стандартноі бібліотеки для математичних функцій

#include <cmath>

// Створення батьківського класу функцій

class functions{

// Відкриваємо доступ до наслідування

protected:

// Масиви для зберігання точок функції

float \*point\_coordinates\_x;

float \*point\_coordinates\_y;

// Змінні для вираховування точок функції

float function\_values\_at\_point\_x;

float function\_values\_at\_point\_y;

// Змінні для зберігання центру осей

float center\_coordinate\_point\_x;

float center\_coordinate\_point\_y;

// Змінна для зберігання умови відображення графіка

bool show\_or\_not\_show;

// Зммінні для кількості точок, початку та кінця інтервалу побудови графіка

int number\_of\_points;

float initial\_boundary;

float final\_boundary;

// Відкриття публічного доступу

public:

// Оголошення конструктора бкз параметрів

functions();

// Оголошення диструктора

~functions();

};

#endif

Сурс файл functions.cpp

// Підключення хедр файлу Functions, де написані функції, які треба визначити, надати тіло.

#include "Functions.h"

// Визначення конструктора без параметрів

functions::functions(): function\_values\_at\_point\_x(0), function\_values\_at\_point\_y(0), center\_coordinate\_point\_x(0), center\_coordinate\_point\_y(0), show\_or\_not\_show(false), number\_of\_points(0), initial\_boundary(0), final\_boundary(0){}

// Визначення деструктора

functions::~functions(){

function\_values\_at\_point\_x = 0;

function\_values\_at\_point\_y = 0;

center\_coordinate\_point\_x = 0;

center\_coordinate\_point\_y = 0;

show\_or\_not\_show = false;

number\_of\_points = 0;

initial\_boundary = 0;

final\_boundary = 0;

}

# Додаток Д

Хедр файл F\_х.h

#ifndef F\_x

#define F\_x

// Підключення хедр файлу Functions для використання в цьому модулі класу нащадка function

#include "Functions.h"

// Створення класу нащадка f\_F від functions

class f\_F: public functions{

// Відкриття публічного доступу

public:

// Оголошення конструктора бкз параметрів

f\_F();

// Оголошення конструктора з двома параметрами

f\_F(int x, int y);

// Оголошення диструктора

~f\_F();

// Перевірка на пустоту

bool empty();

// Головна функція, яка створює точки (xi,yi)

void f\_x(float a, float b, int n, int move, int size);

// Функція для повернення xi

float get\_point\_coordinates\_x(int i);

// Функція для повернення yi

float get\_point\_coordinates\_y(int i);

// Функція для встановлення xi

void set\_point\_coordinates\_x(int i, float x);

// Функція для встановлення yi

void set\_point\_coordinates\_y(int i, float y);

// Функція для встановлення відповідної кількості точок

void great\_points(int n);

// Функція для перевірки відображення

bool show\_function();

// Функція для встановлення типу відображення або не відображення

void show\_function(bool show);

// Функція для ссува графіка по осі абцис

void move\_x(float x, int n, int move);

// Функція для ссува графіка по осі оординат

void move\_y(float y, int n, int move);

// Функція для стиснення графіка по осі абцис

void compress\_x(float x, int n);

// Функція для стиснення графіка по осі оординат

void compress\_y(float y, int n);

// Функція для повернення кількості точок для функції

int get\_n();

// Функція для задання кількості точок для функції

void set\_n(int n);

};

#endif

Сурс файл f\_x.cpp

// Підключення хедр файлу F\_x, де написані функції, які треба визначити, надати тіло.

#include "F\_x.h"

// Визначення конструктора без параметрів

f\_F::f\_F(){};

// Визначення деструктора

f\_F::~f\_F(){};

// Визначення конструктора з параметрами

f\_F::f\_F(int x, int y){

center\_coordinate\_point\_x = x;

center\_coordinate\_point\_y = y;

}

// Повернення істини, якщо точок немає

bool f\_F::empty(){return point\_coordinates\_x == nullptr;}

// Визначення головної функції, яка створює точки (xi,yi)

void f\_F::f\_x(float a, float b, int n, int move, int size){

// Надання проміжків відрисовки функції та кількості точок

f\_F::initial\_boundary=a;

f\_F::final\_boundary=b;

f\_F::number\_of\_points=n;

if(n<=0){}

else{

// Створення масивві для точок (xi, yi)

point\_coordinates\_x=new float[n];

point\_coordinates\_y=new float[n];

// Якщо одна точка.

if(n==1){

// За х беремо середнє арифметичне проміжку

function\_values\_at\_point\_x = (a+b)\*0.5;

// За y беремо потрібну функцію але із знаком мінус попереду

if(6<function\_values\_at\_point\_x && function\_values\_at\_point\_x<10) function\_values\_at\_point\_y=-((abs(function\_values\_at\_point\_x\*log(4\*function\_values\_at\_point\_x)-4)\*sqrt(function\_values\_at\_point\_x))/(exp((5\*function\_values\_at\_point\_x-1)/7)));

else if(-3<function\_values\_at\_point\_x && function\_values\_at\_point\_x<=1) function\_values\_at\_point\_y = -(7+ pow(function\_values\_at\_point\_x,-4));

else function\_values\_at\_point\_y = -pow(function\_values\_at\_point\_x,-10);

// (x,y) це буде сума початкових координат та значень функції множножених на масштаб відображення та мінус масштаб точки point\_coordinates\_x[0]=center\_coordinate\_point\_x+function\_values\_at\_point\_x\*move-size; point\_coordinates\_y[0]=center\_coordinate\_point\_y+function\_values\_at\_point\_y\*move-size;

}else{

// Шукаємо крок між вузловими точками

double step=(b-a)/(n-1);

for(int i=0; i<n; i++){

// За х беремо суму початкового значення відрізка відображення функції та кроку, у перший раз крок нульовий

function\_values\_at\_point\_x=a+step\*i;

// За y беремо потрібну функцію але із знаком мінус попереду

if(6<function\_values\_at\_point\_x && function\_values\_at\_point\_x<10) function\_values\_at\_point\_y=-((abs(function\_values\_at\_point\_x\*log(4\*function\_values\_at\_point\_x)-4)\*sqrt(function\_values\_at\_point\_x))/(exp((5\*function\_values\_at\_point\_x-1)/7)));

else if(-3<function\_values\_at\_point\_x && function\_values\_at\_point\_x<=1) function\_values\_at\_point\_y = -(7+ pow(function\_values\_at\_point\_x,-4));

else function\_values\_at\_point\_y = -pow(function\_values\_at\_point\_x,-10);

// (xi,yi) це буде сума початкових координат та значень функції множножених на масштаб відображення та мінус масштаб точки

point\_coordinates\_x[i]=center\_coordinate\_point\_x+function\_values\_at\_point\_x\*move-size;

point\_coordinates\_y[i]=center\_coordinate\_point\_y+function\_values\_at\_point\_y\*move-size;

}

}

}

}

// Визначаємо функцію для повернення хі

float f\_F::get\_point\_coordinates\_x(int i){ return point\_coordinates\_x[i];}

// Визначаємо функцію для повернення уі

float f\_F::get\_point\_coordinates\_y(int i){ return point\_coordinates\_y[i];}

// Визначаємо функцію для встановлення хі

void f\_F::set\_point\_coordinates\_x(int i, float x){ point\_coordinates\_x[i] = x;}

// Визначаємо функцію для встановлення уі

void f\_F::set\_point\_coordinates\_y(int i, float y){ point\_coordinates\_y[i] = y;}

// Визначення функції створення відповідної кількості точок для відображення

void f\_F::great\_points(int n){

point\_coordinates\_x=new float[n];

point\_coordinates\_y=new float[n];

}

// Повернення істнини, якщо графік слід відображати

bool f\_F::show\_function() {return show\_or\_not\_show;}

// Надання умову відображення чи не відображення графіка

void f\_F::show\_function(bool show) {show\_or\_not\_show=show;}

// Зсув графіка по осі абцис

void f\_F::move\_x(float x, int n, int move){

// Додавання до кожної координати х певного коефіцієнта

for(int i=0; i<n; i++)

point\_coordinates\_x[i] += x\*move;

}

// Зсув графіка по осі ординат

void f\_F::move\_y(float y, int n, int move){

// Додавання до кожної координати y певного коефіцієнта

for(int i=0; i<n; i++)

point\_coordinates\_y[i] -= y\*move;

}

// Стиснення/розтягування графіка відносно осі абцис

void f\_F::compress\_x(float x, int n){

for(int i=0; i<n; i++)

{

point\_coordinates\_x[i] -= center\_coordinate\_point\_x;

point\_coordinates\_x[i] += 2;

point\_coordinates\_x[i] \*= x;

point\_coordinates\_x[i] += center\_coordinate\_point\_x;

point\_coordinates\_x[i] -= 2;

}

}

// Стиснення/розтягування графіка відносно осі ординат

void f\_F::compress\_y(float y, int n){

for(int i=0; i<n; i++)

{

point\_coordinates\_y[i] -= center\_coordinate\_point\_y;

point\_coordinates\_y[i] += 2;

point\_coordinates\_y[i] \*= y;

point\_coordinates\_y[i] += center\_coordinate\_point\_y;

point\_coordinates\_y[i] -= 2;

}

}

// Повернення кількості точок у функції

int f\_F::get\_n(){return number\_of\_points;}

// Встановлення кількості точок у функції

void f\_F::set\_n(int n){number\_of\_points = n;}

# Додаток Е

Хедр файл F\_cos.h

#ifndef F\_cos

#define F\_cos

// Підключення хедр файлу Functions для використання в цьому модулі класу нащадка function

#include "Functions.h"

// Створення класу нащадка f\_cos від functions

class f\_cos: public functions{

// Відкриття публічного доступу

public:

// Оголошення конструктора бкз параметрів

f\_cos();

// Оголошення конструктора з двома параметрами

f\_cos(int x, int y);

// Оголошення диструктора

~f\_cos();

// Перевірка на пустоту

bool empty();

// Головна функція, яка створює точки (xi,yi)

void f\_x(float a, float b, int n, int move, int size);

// Функція для повернення xi

float get\_point\_coordinates\_x(int i);

// Функція для повернення yi

float get\_point\_coordinates\_y(int i);

// Функція для встановлення xi

void set\_point\_coordinates\_x(int i, float x);

// Функція для встановлення yi

void set\_point\_coordinates\_y(int i, float y);

// Функція для встановлення відповідної кількості точок

void great\_points(int n);

// Функція для перевірки відображення

bool show\_function();

// Функція для встановлення типу відображення або не відображення

void show\_function(bool show);

// Функція для ссува графіка по осі абцис

void move\_x(float x, int n, int move);

// Функція для ссува графіка по осі оординат

void move\_y(float y, int n, int move);

// Функція для стиснення графіка по осі абцис

void compress\_x(float x, int n);

// Функція для стиснення графіка по осі оординат

void compress\_y(float y, int n);

// Функція для повернення кількості точок для функції

int get\_n();

// Функція для задання кількості точок для функції

void set\_n(int n);

};

#endif

Сурс файл f\_cos.cpp

// Підключення хедр файлу F\_cos, де написані функції, які треба визначити, надати тіло.

#include "F\_cos.h"

// Визначення конструктора без параметрів

f\_cos::f\_cos(){};

// Визначення деструктора

f\_cos::~f\_cos(){};

// Визначення конструктора з параметрами

f\_cos::f\_cos(int x, int y){

center\_coordinate\_point\_x = x;

center\_coordinate\_point\_y = y;

}

// Повернення істини, якщо точок немає

bool f\_cos::empty(){return point\_coordinates\_x == nullptr;}

// Визначення головної функції, яка створює точки (xi,yi)

void f\_cos::f\_x(float a, float b, int n, int move, int size){

// Надання проміжків відрисовки функції та кількості точок

f\_cos::initial\_boundary=a;

f\_cos::final\_boundary=b;

f\_cos::number\_of\_points=n;

if(n<=0){}

else{

// Створення масивві для точок (xi, yi)

point\_coordinates\_x=new float[n];

point\_coordinates\_y=new float[n];

// Якщо одна точка.

if(n==1){

// За х беремо середнє арифметичне проміжку

function\_values\_at\_point\_x = (a+b)\*0.5;

// За y беремо потрібну функцію але із знаком мінус попереду

function\_values\_at\_point\_y = -cos(function\_values\_at\_point\_x);

// (x,y) це буде сума початкових координат та значень функції множножених на масштаб відображення та мінус масштаб точки point\_coordinates\_x[0]=center\_coordinate\_point\_x+function\_values\_at\_point\_x\*move-size;

point\_coordinates\_y[0]=center\_coordinate\_point\_y+function\_values\_at\_point\_y\*move-size;

}else{

// Шукаємо крок між вузловими точками

double step=(b-a)/(n-1);

for(int i=0; i<n; i++){

// За х беремо суму початкового значення відрізка відображення функції та кроку, у перший раз крок нульовий

function\_values\_at\_point\_x=a+step\*i;

// За y беремо потрібну функцію але із знаком мінус попереду

function\_values\_at\_point\_y = -cos(function\_values\_at\_point\_x);

// (xi,yi) це буде сума початкових координат та значень функції множножених на масштаб відображення та мінус масштаб точки

point\_coordinates\_x[i]=center\_coordinate\_point\_x+function\_values\_at\_point\_x\*move-size; point\_coordinates\_y[i]=center\_coordinate\_point\_y+function\_values\_at\_point\_y\*move-size;

}

}

}

}

// Визначаємо функцію для повернення хі

float f\_cos::get\_point\_coordinates\_x(int i){ return point\_coordinates\_x[i];}

// Визначаємо функцію для повернення уі

float f\_cos::get\_point\_coordinates\_y(int i){ return point\_coordinates\_y[i];}

// Визначаємо функцію для встановлення хі

void f\_cos::set\_point\_coordinates\_x(int i, float x){ point\_coordinates\_x[i] = x;}

// Визначаємо функцію для встановлення уі

void f\_cos::set\_point\_coordinates\_y(int i, float y){ point\_coordinates\_y[i] = y;}

// Визначення функції створення відповідної кількості точок для відображення

void f\_cos::great\_points(int n){

point\_coordinates\_x=new float [n];

point\_coordinates\_y=new float [n];

}

// Повернення істнини, якщо графік слід відображати

bool f\_cos::show\_function() {return show\_or\_not\_show;}

// Надання умову відображення чи не відображення графіка

void f\_cos::show\_function(bool show) {show\_or\_not\_show=show;}

// Зсув графіка по осі абцис

void f\_cos::move\_x(float x, int n, int move){

// Додавання до кожної координати х певного коефіцієнта

for(int i=0; i<n; i++)

point\_coordinates\_x[i] += x\*move;

}

// Зсув графіка по осі ординат

void f\_cos::move\_y(float y, int n, int move){

// Додавання до кожної координати y певного коефіцієнта

for(int i=0; i<n; i++)

point\_coordinates\_y[i] -= y\*move;

}

// Стиснення/розтягування графіка відносно осі абцис

void f\_cos::compress\_x(float x, int n){

for(int i=0; i<n; i++)

{

point\_coordinates\_x[i] -= center\_coordinate\_point\_x;

point\_coordinates\_x[i] += 2;

point\_coordinates\_x[i] \*= x;

point\_coordinates\_x[i] += center\_coordinate\_point\_x;

point\_coordinates\_x[i] -= 2;

}

}

// Стиснення/розтягування графіка відносно осі ординат

void f\_cos::compress\_y(float y, int n){

for(int i=0; i<n; i++)

{

point\_coordinates\_y[i] -= center\_coordinate\_point\_y;

point\_coordinates\_y[i] += 2;

point\_coordinates\_y[i] \*= y;

point\_coordinates\_y[i] += center\_coordinate\_point\_y;

point\_coordinates\_y[i] -= 2;

}

}

// Повернення кількості точок у функції

int f\_cos::get\_n(){return number\_of\_points;}

// Встановлення кількості точок у функції

void f\_cos::set\_n(int n){number\_of\_points = n;}

# Додаток Є

Хедр файл F\_sin.h

#ifndef F\_sin

#define F\_sin

// Підключення хедр файлу Functions для використання в цьому модулі класу нащадка function

#include "Functions.h"

// Створення класу нащадка f\_sin від functions

class f\_sin: public functions{

// Відкриття публічного доступу

public:

// Оголошення конструктора бкз параметрів

f\_sin();

// Оголошення конструктора з двома параметрами

f\_sin(int x, int y);

// Оголошення диструктора

~f\_sin();

// Перевірка на пустоту

bool empty();

// Головна функція, яка створює точки (xi,yi)

void f\_x(float a, float b, int n, int move, int size);

// Функція для повернення xi

float get\_point\_coordinates\_x(int i);

// Функція для повернення yi

float get\_point\_coordinates\_y(int i);

// Функція для встановлення xi

void set\_point\_coordinates\_x(int i, float x);

// Функція для встановлення yi

void set\_point\_coordinates\_y(int i, float y);

// Функція для встановлення відповідної кількості точок

void great\_points(int n);

// Функція для перевірки відображення

bool show\_function();

// Функція для встановлення типу відображення або не відображення

void show\_function(bool show);

// Функція для ссува графіка по осі абцис

void move\_x(float x, int n, int move);

// Функція для ссува графіка по осі оординат

void move\_y(float y, int n, int move);

// Функція для стиснення графіка по осі абцис

void compress\_x(float x, int n);

// Функція для стиснення графіка по осі оординат

void compress\_y(float y, int n);

// Функція для повернення кількості точок для функції

int get\_n();

// Функція для задання кількості точок для функції

void set\_n(int n);

};

#endif

Сурс файл f\_sin.cpp

// Підключення хедр файлу F\_sin, де написані функції, які треба визначити, надати тіло.

#include "F\_sin.h"

// Визначення конструктора без параметрів

f\_sin::f\_sin(){};

// Визначення деструктора

f\_sin::~f\_sin(){};

// Визначення конструктора з параметрами

f\_sin::f\_sin(int x, int y){

center\_coordinate\_point\_x = x;

center\_coordinate\_point\_y = y;

}

// Повернення істини, якщо точок немає

bool f\_sin::empty(){return point\_coordinates\_x == nullptr;}

// Визначення головної функції, яка створює точки (xi,yi)

void f\_sin::f\_x(float a, float b, int n, int move, int size){

// Надання проміжків відрисовки функції та кількості точок

f\_sin::initial\_boundary=a;

f\_sin::final\_boundary=b;

f\_sin::number\_of\_points=n;

if(n<=0){}

else{

// Створення масивві для точок (xi, yi)

point\_coordinates\_x=new float[n];

point\_coordinates\_y=new float[n];

// Якщо одна точка.

if(n==1){

// За х беремо середнє арифметичне проміжку

function\_values\_at\_point\_x = (a+b)\*0.5;

// За y беремо потрібну функцію але із знаком мінус попереду

function\_values\_at\_point\_y = -sin(function\_values\_at\_point\_x);

// (x,y) це буде сума початкових координат та значень функції множножених на масштаб відображення та мінус масштаб точки

point\_coordinates\_x[0]=center\_coordinate\_point\_x+function\_values\_at\_point\_x\*move-size; point\_coordinates\_y[0]=center\_coordinate\_point\_y+function\_values\_at\_point\_y\*move-size;

}else{

// Шукаємо крок між вузловими точками

double step=(b-a)/(n-1);

for(int i=0; i<n; i++){

// За х беремо суму початкового значення відрізка відображення функції та кроку, у перший раз крок нульовий

function\_values\_at\_point\_x=a+step\*i;

// За y беремо потрібну функцію але із знаком мінус попереду

function\_values\_at\_point\_y = -sin(function\_values\_at\_point\_x);

// (xi,yi) це буде сума початкових координат та значень функції множножених на масштаб відображення та мінус масштаб точки

point\_coordinates\_x[i]=center\_coordinate\_point\_x+function\_values\_at\_point\_x\*move-size; point\_coordinates\_y[i]=center\_coordinate\_point\_y+function\_values\_at\_point\_y\*move-size;

}

}

}

}

// Визначаємо функцію для повернення хі

float f\_sin::get\_point\_coordinates\_x(int i){ return point\_coordinates\_x[i];}

// Визначаємо функцію для повернення уі

float f\_sin::get\_point\_coordinates\_y(int i){ return point\_coordinates\_y[i];}

// Визначаємо функцію для встановлення хі

void f\_sin::set\_point\_coordinates\_x(int i, float x){ point\_coordinates\_x[i] = x;}

// Визначаємо функцію для встановлення уі

void f\_sin::set\_point\_coordinates\_y(int i, float y){ point\_coordinates\_y[i] = y;}

// Визначення функції створення відповідної кількості точок для відображення

void f\_sin::great\_points(int n){

point\_coordinates\_x=new float[n];

point\_coordinates\_y=new float[n];

}

// Повернення істнини, якщо графік слід відображати

bool f\_sin::show\_function() {return show\_or\_not\_show;}

// Надання умову відображення чи не відображення графіка

void f\_sin::show\_function(bool show) {show\_or\_not\_show=show;}

// Зсув графіка по осі абцис

void f\_sin::move\_x(float x, int n, int move){

// Додавання до кожної координати х певного коефіцієнта

for(int i=0; i<n; i++)

point\_coordinates\_x[i] += x\*move;

}

// Зсув графіка по осі ординат

void f\_sin::move\_y(float y, int n, int move){

// Додавання до кожної координати y певного коефіцієнта

for(int i=0; i<n; i++)

point\_coordinates\_y[i] -= y\*move;

}

// Стиснення/розтягування графіка відносно осі абцис

void f\_sin::compress\_x(float x, int n){

for(int i=0; i<n; i++)

{

point\_coordinates\_x[i] -= center\_coordinate\_point\_x;

point\_coordinates\_x[i] += 2;

point\_coordinates\_x[i] \*= x;

point\_coordinates\_x[i] += center\_coordinate\_point\_x;

point\_coordinates\_x[i] -= 2;

}

}

// Стиснення/розтягування графіка відносно осі ординат

void f\_sin::compress\_y(float y, int n){

for(int i=0; i<n; i++)

{

point\_coordinates\_y[i] -= center\_coordinate\_point\_y;

point\_coordinates\_y[i] += 2;

point\_coordinates\_y[i] \*= y;

point\_coordinates\_y[i] += center\_coordinate\_point\_y;

point\_coordinates\_y[i] -= 2;

}

}

// Повернення кількості точок у функції

int f\_sin::get\_n(){return number\_of\_points;}

// Встановлення кількості точок у функції

void f\_sin::set\_n(int n){number\_of\_points = n;}

# Додаток Ж

Хедр файл F\_rand.h

#ifndef F\_rand

#define F\_rand

// Підключення хедр файлу Functions для використання в цьому модулі класу нащадка function

#include "Functions.h"

// Створення класу нащадка f\_rand від functions

class f\_rand: public functions{

// Відкриття публічного доступу

public:

// Оголошення конструктора бкз параметрів

f\_rand();

// Оголошення конструктора з двома параметрами

f\_rand(int x, int y);

// Оголошення диструктора

~f\_rand();

// Перевірка на пустоту

bool empty();

// Головна функція, яка створює точки (xi,yi)

void f\_x(float a, float b, int n, int move, int size);

// Функція для повернення xi

float get\_point\_coordinates\_x(int i);

// Функція для повернення yi

float get\_point\_coordinates\_y(int i);

// Функція для встановлення xi

void set\_point\_coordinates\_x(int i, float x);

// Функція для встановлення yi

void set\_point\_coordinates\_y(int i, float y);

// Функція для встановлення відповідної кількості точок

void great\_points(int n);

// Функція для перевірки відображення

bool show\_function();

// Функція для встановлення типу відображення або не відображення

void show\_function(bool show);

// Функція для ссува графіка по осі абцис

void move\_x(float x, int n, int move);

// Функція для ссува графіка по осі оординат

void move\_y(float y, int n, int move);

// Функція для стиснення графіка по осі абцис

void compress\_x(float x, int n);

// Функція для стиснення графіка по осі оординат

void compress\_y(float y, int n);

// Функція для повернення кількості точок для функції

int get\_n();

// Функція для задання кількості точок для функції

void set\_n(int n);

};

#endif

Сурс файл f\_rand.cpp

// Підключення хедр файлу F\_rand, де написані функції, які треба визначити, надати тіло.

#include "F\_rand.h"

// Визначення конструктора без параметрів

f\_rand::f\_rand(){};

// Визначення деструктора

f\_rand::~f\_rand(){};

// Визначення конструктора з параметрами

f\_rand::f\_rand(int x, int y){

center\_coordinate\_point\_x = x;

center\_coordinate\_point\_y = y;

}

// Повернення істини, якщо точок немає

bool f\_rand::empty(){return point\_coordinates\_x == nullptr;}

// Визначення головної функції, яка створює точки (xi,yi)

void f\_rand::f\_x(float a, float b, int n, int move, int size){

// Надання проміжків відрисовки функції та кількості точок

f\_rand::initial\_boundary=a;

f\_rand::final\_boundary=b;

f\_rand::number\_of\_points=n;

if(n<=0){}

else{

// Створення масивві для точок (xi, yi)

point\_coordinates\_x=new float[n];

point\_coordinates\_y=new float[n];

// Якщо одна точка.

if(n==1){

// За х беремо середнє арифметичне проміжку

function\_values\_at\_point\_x = (a+b)\*0.5;

// За y беремо потрібну функцію але із знаком мінус попереду

function\_values\_at\_point\_y = -(rand()%(8+8+1)-8);

// (x,y) це буде сума початкових координат та значень функції множножених на масштаб відображення та мінус масштаб точки

point\_coordinates\_x[0]=center\_coordinate\_point\_x+function\_values\_at\_point\_x\*move-size; point\_coordinates\_y[0]=center\_coordinate\_point\_y+function\_values\_at\_point\_y\*move-size;

}else{

// Шукаємо крок між вузловими точками

double step=(b-a)/(n-1);

for(int i=0; i<n; i++){

// За х беремо суму початкового значення відрізка відображення функції та кроку, у перший раз крок нульовий

function\_values\_at\_point\_x=a+step\*i;

// За y беремо потрібну функцію але із знаком мінус попереду

function\_values\_at\_point\_y = -(rand()%(8+8+1)-8);

// (xi,yi) це буде сума початкових координат та значень функції множножених на масштаб відображення та мінус масштаб точки

point\_coordinates\_x[i]=center\_coordinate\_point\_x+function\_values\_at\_point\_x\*move-size; point\_coordinates\_y[i]=center\_coordinate\_point\_y+function\_values\_at\_point\_y\*move-size;

}

}

}

}

// Визначаємо функцію для повернення хі

float f\_rand::get\_point\_coordinates\_x(int i){ return point\_coordinates\_x[i];}

// Визначаємо функцію для повернення уі

float f\_rand::get\_point\_coordinates\_y(int i){ return point\_coordinates\_y[i];}

// Визначаємо функцію для встановлення хі

void f\_rand::set\_point\_coordinates\_x(int i, float x){ point\_coordinates\_x[i] = x;}

// Визначаємо функцію для встановлення уі

void f\_rand::set\_point\_coordinates\_y(int i, float y){ point\_coordinates\_y[i] = y;}

// Визначення функції створення відповідної кількості точок для відображення

void f\_rand::great\_points(int n){

point\_coordinates\_x=new float[n];

point\_coordinates\_y=new float[n];

}

// Повернення істнини, якщо графік слід відображати

bool f\_rand::show\_function() {return show\_or\_not\_show;}

// Надання умову відображення чи не відображення графіка

void f\_rand::show\_function(bool show) {show\_or\_not\_show=show;}

// Зсув графіка по осі абцис

void f\_rand::move\_x(float x, int n, int move){

// Додавання до кожної координати х певного коефіцієнта

for(int i=0; i<n; i++)

point\_coordinates\_x[i] += x\*move;

}

// Зсув графіка по осі ординат

void f\_rand::move\_y(float y, int n, int move){

// Додавання до кожної координати y певного коефіцієнта

for(int i=0; i<n; i++)

point\_coordinates\_y[i] -= y\*move;

}

// Стиснення/розтягування графіка відносно осі абцис

void f\_rand::compress\_x(float x, int n){

for(int i=0; i<n; i++)

{

point\_coordinates\_x[i] -= center\_coordinate\_point\_x;

point\_coordinates\_x[i] += 2;

point\_coordinates\_x[i] \*= x;

point\_coordinates\_x[i] += center\_coordinate\_point\_x;

point\_coordinates\_x[i] -= 2;

}

}

// Стиснення/розтягування графіка відносно осі ординат

void f\_rand::compress\_y(float y, int n){

for(int i=0; i<n; i++)

{

point\_coordinates\_y[i] -= center\_coordinate\_point\_y;

point\_coordinates\_y[i] += 2;

point\_coordinates\_y[i] \*= y;

point\_coordinates\_y[i] += center\_coordinate\_point\_y;

point\_coordinates\_y[i] -= 2;

}

}

// Повернення кількості точок у функції

int f\_rand::get\_n(){return number\_of\_points;}

// Встановлення кількості точок у функції

void f\_rand::set\_n(int n){number\_of\_points = n;}

# Додаток З

Хедр файл Additional\_windows.h

#ifndef Additional\_windows

#define Additional\_windows

// Підключення стандартноі бібліотеки для вводу/виводу та простору імен std

#include <iostream>

using namespace std;

// Підключення бібліотеки sfml для вграфіки та простору імен sf

#include <SFML/Graphics.hpp>

using namespace sf;

// Додавання проміжків для рандомної функції

void Random\_function\_settings\_A\_B(Font& font, string& A, string& B);

// Вікно з інформацією про програму та користувача

void job\_information(Font& font);

// Вікно з інформацією про упішне збереження

void program\_save(Font& font, string& text);

// Вікно з інформацією про упішне відкриття

void program\_open(Font& font, string& text);

// Вікно з інформацією про невдачу зберегти файл

void program\_does\_not\_save(Font& font, string& text);

// Вікно з інформацією про невдачу відкрити файл

void program\_does\_not\_open(Font& font, string& text);

#endif

Сурс файл additional\_windows.cpp

// Підключення хедр файлу Additional\_windows, де написані функції, які треба визначити, надати тіло.

#include "Additional\_windows.h"

// Підключення хедр файлів Text\_input та Error\_windows, для використання в цьому модулі функцій з цих файлів

#include "Text\_input.h"

#include "Error\_windows.h"

// Додавання проміжків для рандомної функції

void Random\_function\_settings\_A\_B(Font& font, string& A, string& B) {

// Створення вікна

RenderWindow window(VideoMode(500, 250), "Introduce an additional interval for the rand function");

//Створення підказок, що має ввести користувач

Text user\_hint("User Input", font, 24);

user\_hint.setFillColor(Color::Black);

user\_hint.setStyle(Text::Bold);

user\_hint.setPosition((500 - user\_hint.getLocalBounds().width)/2,10);

Text user\_hint\_A("A", font, 24);

user\_hint\_A.setFillColor(Color::Black);

user\_hint\_A.setStyle(Text::Bold);

user\_hint\_A.setPosition(5\*user\_hint\_A.getLocalBounds().width, 50 + user\_hint.getLocalBounds().height);

Text user\_hint\_B("B", font, 24);

user\_hint\_B.setFillColor(Color::Black);

user\_hint\_B.setStyle(Text::Bold);

user\_hint\_B.setPosition(5\*user\_hint\_A.getLocalBounds().width, 100 + user\_hint.getLocalBounds().height + user\_hint\_B.getLocalBounds().height);

// Створення білих прямокутників із чорною оболочкою, для візуального розуміння де буде текст та куди клікати, щоб його написати

RectangleShape user\_A\_wright(Vector2f(15 \* user\_hint\_A.getLocalBounds().width, user\_hint\_A.getLocalBounds().height));

user\_A\_wright.setFillColor(Color::Transparent);

user\_A\_wright.setFillColor(Color::White);

user\_A\_wright.setOutlineColor(Color::Black);

user\_A\_wright.setOutlineThickness(2);

user\_A\_wright.setPosition(7\*user\_hint\_A.getLocalBounds().width, 57 + user\_hint.getLocalBounds().height);

RectangleShape user\_B\_wright(Vector2f(15 \* user\_hint\_A.getLocalBounds().width, user\_hint\_A.getLocalBounds().height));

user\_B\_wright.setFillColor(Color::Transparent);

user\_B\_wright.setFillColor(Color::White);

user\_B\_wright.setOutlineColor(Color::Black);

user\_B\_wright.setOutlineThickness(2);

user\_B\_wright.setPosition(7\*user\_hint\_A.getLocalBounds().width, 107 + user\_hint.getLocalBounds().height + user\_hint\_B.getLocalBounds().height);

// Створення обʼєктів для відображення введеного тексту

Text text\_display\_written\_A("", font, 24);

text\_display\_written\_A.setFillColor(Color::Black);

text\_display\_written\_A.setPosition(7\*user\_hint\_A.getLocalBounds().width, 50 + user\_hint.getLocalBounds().height);

Text text\_display\_written\_B("", font, 24);

text\_display\_written\_B.setFillColor(Color::Black);

text\_display\_written\_B.setPosition(7\*user\_hint\_A.getLocalBounds().width, 100 + user\_hint.getLocalBounds().height + user\_hint\_B.getLocalBounds().height);

// Створення прямокутної кнопки

RectangleShape generate\_button(Vector2f(150, 50));

generate\_button.setPosition((500 - generate\_button.getLocalBounds().width)/2, (150 + user\_hint.getLocalBounds().height + user\_hint\_B.getLocalBounds().height));

generate\_button.setFillColor(Color::Transparent);

generate\_button.setFillColor(Color::White);

generate\_button.setOutlineColor(Color::Black);

generate\_button.setOutlineThickness(2);

// Створення назви кнопки

Text generate\_button\_text("Generate", font, 24);

generate\_button\_text.setFillColor(Color::Black);

generate\_button\_text.setOrigin(generate\_button\_text.getLocalBounds().left + generate\_button\_text.getLocalBounds().width / 2, generate\_button\_text.getLocalBounds().top + generate\_button\_text.getLocalBounds().height / 2);

generate\_button\_text.setPosition(generate\_button.getPosition().x + generate\_button.getSize().x / 2, generate\_button.getPosition().y + generate\_button.getSize().y / 2);

// Змінні для розрахункових дій

float number\_A = 0.0;

float number\_B = 0.0;

// Змінні для перевірки натискання

bool click\_A = false;

bool click\_B = false;

bool click\_generate\_button = false;

// змінні для перевірки вводу

string text\_assistant\_A;

string text\_assistant\_B;

while (window.isOpen()) {

Event event;

// Обробка події

while (window.pollEvent(event)) {

// Опрацювання кнопки закриття

if (event.type == Event::Closed)

window.close();

// Опрацювання кліку миші

if (event.type == Event::MouseButtonPressed && event.mouseButton.button == Mouse::Left) {

Vector2i mousePosition = Mouse::getPosition(window);

Vector2f mousePositionF(mousePosition.x, mousePosition.y);

// Якщо мишка клікнула в області вводу для значення А

if (user\_A\_wright.getGlobalBounds().contains(mousePositionF)) {

click\_A = true;

click\_B = false;

click\_generate\_button = false;

}

// Якщо мишка клікнула в області вводу для значення И

if (user\_B\_wright.getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = true;

click\_generate\_button = false;

}

// Якщо мишка клікнула в області кнопки генерації

if (generate\_button.getGlobalBounds().contains(mousePositionF) || generate\_button\_text.getGlobalBounds().contains(mousePositionF)) {

click\_A = false;

click\_B = false;

click\_generate\_button = true;

}

}

// Опрацювання дій коли користувач вводить текст

if (event.type == Event::TextEntered) {

// Якщо клікнули в області вводу А - можна вводити числа в області А і вони будуть зберігатися та відтворюватися на екрані

if(click\_A)

text\_input\_for\_A\_B\_k(event, text\_assistant\_A, text\_display\_written\_A, number\_A);

// Якщо клікнули в області вводу В - можна вводити числа в області В і вони будуть зберігатися та відтворюватися на екрані

if(click\_B)

text\_input\_for\_A\_B\_k(event, text\_assistant\_B, text\_display\_written\_B, number\_B);

}

// Якщо натиснута кнопка генерації

if(click\_generate\_button)

{

// Якщо значення А та В порожні - виводиться вікно про помилку

if(text\_assistant\_A.empty() || text\_assistant\_B.empty())

A\_B\_N\_were\_not\_entered(font);

// Якщо початкове значення більше ніж кінцеве - змінити місцями значення А та В, вивісти відповідне повідомлення

if(number\_A > number\_B){

the\_beginning\_is\_greater\_than\_the\_end(font);

swap(number\_A,number\_B);

String temp = text\_display\_written\_A.getString();

String temp2 = text\_display\_written\_B.getString();

text\_assistant\_A = text\_display\_written\_B.getString();

text\_assistant\_B = text\_display\_written\_A.getString();

text\_display\_written\_A.setString(temp2);

text\_display\_written\_B.setString(temp);

}

// Надання значень проміжків, які було написані в цьому вікні.

A = text\_assistant\_A;

B = text\_assistant\_B;

window.close();

}

}

window.clear(Color::White);

window.draw(user\_hint);

window.draw(user\_hint\_A);

window.draw(user\_hint\_B);

window.draw(user\_A\_wright);

window.draw(user\_B\_wright);

window.draw(generate\_button);

window.draw(generate\_button\_text);

window.draw(text\_display\_written\_A);

window.draw(text\_display\_written\_B);

window.display();

}

}

// Вікно з інформацією про програму та користувача

void job\_information(Font& font){

// Створення вікна

RenderWindow window(VideoMode(1700, 900), "Job\_information");

// Створення тексту для повідомлення

char text[] = "Program name: Graph of a function.\n\nAuthor's name: Andrey.\n\nAuthor's last name: Ovdienko.\n\nAuthor's patronymic: Vladimirovich.\n\nGender: male.\n\nDate of birth: 05/17/2005.\n\nPlace of study: Faculty of Applied Mathematics (113) DNU named after. Olesya Gonchar.\n\nGroup: PA-22-2.\n\nProgramming language: C++.\n\nArchitectural Software: Apple M1 Pro.\n\nPurpose: educational practice.\n\nMission: Demonstrate programming skills in C++, as well as knowledge of OOP.\n\nMain library: SFML.\n\nDownload the core library and learn how to use it: https://www.sfml-dev.org/.\n\nTarget audience: people involved in graphical programming.\n\nReason for choosing SFML: completing the course \"Computer Graphics and Animation\" in the C++ programming language, using OpenGL libraries.";

// Створення тексту для виводу на екран

Text Message(text, font, 24);

Message.setFillColor(Color::Black);

Message.setStyle(Text::Bold);

Message.setPosition((1700 - Message.getLocalBounds().width)/2, (900 - Message.getLocalBounds().height)/2);

while (window.isOpen()) {

Event event;

// Обробка події

while (window.pollEvent(event))

// Опрацювання кнопки закриття

if (event.type == Event::Closed)

window.close();

window.clear(Color::White);

window.draw(Message);

window.display();

}

}

// Вікно з інформацією про упішне збереження

void program\_save(Font& font, string& text){

// Створення вікна

RenderWindow window(VideoMode(500, 500), "");

string str = "Program saved: " + text;

// Створення тексту для виводу на екран

Text Message(str, font, 24);

Message.setFillColor(Color::Black);

Message.setStyle(Text::Bold);

Message.setPosition((500 - Message.getLocalBounds().width)/2, (250 - Message.getLocalBounds().height)/2);

// Створення прямокутної кнопки

RectangleShape ok\_button(Vector2f(100, 100));

ok\_button.setPosition((500 - ok\_button.getLocalBounds().width)/2, (500 - Message.getLocalBounds().height)/2 + 3\*Message.getLocalBounds().height );

ok\_button.setFillColor(Color::Transparent);

ok\_button.setFillColor(Color::White);

ok\_button.setOutlineColor(Color::Black);

ok\_button.setOutlineThickness(2);

// Створення назви кнопки

Text ok\_button\_text("OK", font, 24);

ok\_button\_text.setFillColor(Color::Black);

ok\_button\_text.setOrigin(ok\_button\_text.getLocalBounds().left + ok\_button\_text.getLocalBounds().width / 2, ok\_button\_text.getLocalBounds().top + ok\_button\_text.getLocalBounds().height / 2);

ok\_button\_text.setPosition(ok\_button.getPosition().x + ok\_button.getSize().x / 2, ok\_button.getPosition().y + ok\_button.getSize().y / 2);

// Змінна для перевірки натискання

bool click\_ok\_button = false;

while (window.isOpen()) {

Event event;

// Обробка події

while (window.pollEvent(event)) {

// Опрацювання кнопки закриття

if (event.type == Event::Closed)

window.close();

// Опрацювання кліку пиші

if (event.type == Event::MouseButtonPressed && event.mouseButton.button == Mouse::Left) {

Vector2i mousePosition = Mouse::getPosition(window);

Vector2f mousePositionF(mousePosition.x, mousePosition.y);

// Якщо мишка клікнула в області значення кнопки згоди

if (ok\_button.getGlobalBounds().contains(mousePositionF) || ok\_button\_text.getGlobalBounds().contains(mousePositionF))

click\_ok\_button = true;

}

}

// Опрацювання дії при натиснені кнопки ОК

if(click\_ok\_button)

window.close();

window.clear(Color::White);

window.draw(Message);

window.draw(ok\_button);

window.draw(ok\_button\_text);

window.display();

}

}

// Вікно з інформацією про упішне відкриття

void program\_open(Font& font, string& text){

// Створення вікна

RenderWindow window(VideoMode(500, 500), "");

string str = "Program opened: " + text;

// Створення тексту для виводу на екран

Text Message(str, font, 24);

Message.setFillColor(Color::Black);

Message.setStyle(Text::Bold);

Message.setPosition((500 - Message.getLocalBounds().width)/2, (250 - Message.getLocalBounds().height)/2);

// Створення прямокутної кнопки

RectangleShape ok\_button(Vector2f(100, 100));

ok\_button.setPosition((500 - ok\_button.getLocalBounds().width)/2, (500 - Message.getLocalBounds().height)/2 + 3\*Message.getLocalBounds().height );

ok\_button.setFillColor(Color::Transparent);

ok\_button.setFillColor(Color::White);

ok\_button.setOutlineColor(Color::Black);

ok\_button.setOutlineThickness(2);

// Створення назви кнопки

Text ok\_button\_text("OK", font, 24);

ok\_button\_text.setFillColor(Color::Black);

ok\_button\_text.setOrigin(ok\_button\_text.getLocalBounds().left + ok\_button\_text.getLocalBounds().width / 2, ok\_button\_text.getLocalBounds().top + ok\_button\_text.getLocalBounds().height / 2);

ok\_button\_text.setPosition(ok\_button.getPosition().x + ok\_button.getSize().x / 2, ok\_button.getPosition().y + ok\_button.getSize().y / 2);

// Змінна для перевірки натискання

bool click\_ok\_button = false;

while (window.isOpen()) {

Event event;

// Обробка події

while (window.pollEvent(event)) {

// Опрацювання кнопки закриття

if (event.type == Event::Closed)

window.close();

// Опрацювання кліку пиші

if (event.type == Event::MouseButtonPressed && event.mouseButton.button == Mouse::Left) {

Vector2i mousePosition = Mouse::getPosition(window);

Vector2f mousePositionF(mousePosition.x, mousePosition.y);

// Якщо мишка клікнула в області значення кнопки згоди

if (ok\_button.getGlobalBounds().contains(mousePositionF) || ok\_button\_text.getGlobalBounds().contains(mousePositionF))

click\_ok\_button = true;

}

}

// Опрацювання дії при натиснені кнопки ОК

if(click\_ok\_button)

window.close();

window.clear(Color::White);

window.draw(Message);

window.draw(ok\_button);

window.draw(ok\_button\_text);

window.display();

}

}

// Вікно з інформацією про невдачу зберегти файл

void program\_does\_not\_save(Font& font, string& text){

// Створення вікна

RenderWindow window(VideoMode(500, 500), "");

string str = "Program didn't save: " + text;

// Створення тексту для виводу на екран

Text Message(str, font, 24);

Message.setFillColor(Color::Black);

Message.setStyle(Text::Bold);

Message.setPosition((500 - Message.getLocalBounds().width)/2, (250 - Message.getLocalBounds().height)/2);

// Створення прямокутної кнопки

RectangleShape ok\_button(Vector2f(100, 100));

ok\_button.setPosition((500 - ok\_button.getLocalBounds().width)/2, (500 - Message.getLocalBounds().height)/2 + 3\*Message.getLocalBounds().height );

ok\_button.setFillColor(Color::Transparent);

ok\_button.setFillColor(Color::White);

ok\_button.setOutlineColor(Color::Black);

ok\_button.setOutlineThickness(2);

// Створення назви кнопки

Text ok\_button\_text("OK", font, 24);

ok\_button\_text.setFillColor(Color::Black);

ok\_button\_text.setOrigin(ok\_button\_text.getLocalBounds().left + ok\_button\_text.getLocalBounds().width / 2, ok\_button\_text.getLocalBounds().top + ok\_button\_text.getLocalBounds().height / 2);

ok\_button\_text.setPosition(ok\_button.getPosition().x + ok\_button.getSize().x / 2, ok\_button.getPosition().y + ok\_button.getSize().y / 2);

// Змінна для перевірки натискання

bool click\_ok\_button = false;

while (window.isOpen()) {

Event event;

// Обробка події

while (window.pollEvent(event)) {

// Опрацювання кнопки закриття

if (event.type == Event::Closed)

window.close();

// Опрацювання кліку пиші

if (event.type == Event::MouseButtonPressed && event.mouseButton.button == Mouse::Left) {

Vector2i mousePosition = Mouse::getPosition(window);

Vector2f mousePositionF(mousePosition.x, mousePosition.y);

// Якщо мишка клікнула в області значення кнопки згоди

if (ok\_button.getGlobalBounds().contains(mousePositionF) || ok\_button\_text.getGlobalBounds().contains(mousePositionF))

click\_ok\_button = true;

}

}

// Опрацювання дії при натиснені кнопки ОК

if(click\_ok\_button)

window.close();

window.clear(Color::White);

window.draw(Message);

window.draw(ok\_button);

window.draw(ok\_button\_text);

window.display();

}

}

// Вікно з інформацією про невдачу відкрити файл

void program\_does\_not\_open(Font& font, string& text){

// Створення вікна

RenderWindow window(VideoMode(500, 500), "");

string str = "Program didn't open: " + text;

// Створення тексту для виводу на екран

Text Message(str, font, 24);

Message.setFillColor(Color::Black);

Message.setStyle(Text::Bold);

Message.setPosition((500 - Message.getLocalBounds().width)/2, (250 - Message.getLocalBounds().height)/2);

// Створення прямокутної кнопки

RectangleShape ok\_button(Vector2f(100, 100));

ok\_button.setPosition((500 - ok\_button.getLocalBounds().width)/2, (500 - Message.getLocalBounds().height)/2 + 3\*Message.getLocalBounds().height );

ok\_button.setFillColor(Color::Transparent);

ok\_button.setFillColor(Color::White);

ok\_button.setOutlineColor(Color::Black);

ok\_button.setOutlineThickness(2);

// Створення назви кнопки

Text ok\_button\_text("OK", font, 24);

ok\_button\_text.setFillColor(Color::Black);

ok\_button\_text.setOrigin(ok\_button\_text.getLocalBounds().left + ok\_button\_text.getLocalBounds().width / 2, ok\_button\_text.getLocalBounds().top + ok\_button\_text.getLocalBounds().height / 2);

ok\_button\_text.setPosition(ok\_button.getPosition().x + ok\_button.getSize().x / 2, ok\_button.getPosition().y + ok\_button.getSize().y / 2);

// Змінна для перевірки натискання

bool click\_ok\_button = false;

while (window.isOpen()) {

Event event;

// Обробка події

while (window.pollEvent(event)) {

// Опрацювання кнопки закриття

if (event.type == Event::Closed)

window.close();

// Опрацювання кліку пиші

if (event.type == Event::MouseButtonPressed && event.mouseButton.button == Mouse::Left) {

Vector2i mousePosition = Mouse::getPosition(window);

Vector2f mousePositionF(mousePosition.x, mousePosition.y);

// Якщо мишка клікнула в області значення кнопки згоди

if (ok\_button.getGlobalBounds().contains(mousePositionF) || ok\_button\_text.getGlobalBounds().contains(mousePositionF))

click\_ok\_button = true;

}

}

// Опрацювання дії при натиснені кнопки ОК

if(click\_ok\_button)

window.close();

window.clear(Color::White);

window.draw(Message);

window.draw(ok\_button);

window.draw(ok\_button\_text);

window.display();

}

}

# Додаток И

Хедр файл Additional\_functions.h

#ifndef Additional\_functions

#define Additional\_functions

// Підключення стандартноі бібліотеки для вводу/виводу та простору імен std

#include <iostream>

using namespace std;

// Оголошення функції, видалить всі нулі після крапки, якщо далі немає інших символів крім нуля

string trimFloatString(const string& str);

#endif

Сурс файл additional\_functions.cpp

// Підключення хедр файлу Additional\_functions, де написані функції, які треба визначити, надати тіло.

#include "Additional\_functions.h"

// Відкидання всіх нулів після крапки, якщо далі немає інших символів крім нуля

string trimFloatString(const string& str) {

size\_t dotPos = str.find('.');

if (dotPos != string::npos) {

size\_t lastNonZero = str.find\_last\_not\_of('0');

if (lastNonZero != string::npos && lastNonZero > dotPos) {

return str.substr(0, lastNonZero + 1);

} else {

return str.substr(0, dotPos);

}

}

return str;

}

# Додаток І

Хедр файл Graph\_display\_switch.h

#ifndef Graph\_display\_switch

#define Graph\_display\_switch

// Підключення бібліотеки sfml для вграфіки та простору імен sf

#include <SFML/Graphics.hpp>

using namespace sf;

// Створення класу кнопок для відображення функцій

class square\_buttons\_with\_selection\_of\_displayed\_functions {

// Створюємо приватні змінні

private:

// Кнопка

RectangleShape button;

// Текст кнопки

Text button\_text;

// Змінна для зберігання умови натиснення на кнопку

bool click\_button;

// Відкриття публічного доступу

public:

// Оголошення конструктора з двома параметрами

square\_buttons\_with\_selection\_of\_displayed\_functions(Font& font, const int& k);

// Функція для повернення кнопки

RectangleShape& get\_button();

// Функція для відображення тексту кнопки

Text& get\_button\_text();

// Функція для перевірки натискання на кнопку

bool isClicked();

// Функція для встановлення значення чи натиснута кнопка

void setClicked(bool clicked);

};

#endif

Сурс файл graph\_display\_switch.cpp

// Підключення хедр файлу Graph\_displacement\_and\_compression, де написані функції, які треба визначити, надати тіло.

#include "Graph\_display\_switch.h"

// Визначення конструктора з двома параметрами

square\_buttons\_with\_selection\_of\_displayed\_functions::square\_buttons\_with\_selection\_of\_displayed\_functions(Font& font, const int& k) {

// Встановлення розмірру позиції та кольору кнопки

button.setSize(Vector2f(30, 30));

button.setPosition(1300, 350 + k\*50);

button.setFillColor(Color::Transparent);

button.setFillColor(Color::White);

button.setOutlineColor(Color::Black);

button.setOutlineThickness(2);

// Встановлення шрифта, текста, розмірата кольора

button\_text.setFont(font);

button\_text.setString("+");

button\_text.setCharacterSize(24);

button\_text.setFillColor(Color::Black);

// Централізація текста відносно кнопки

button\_text.setOrigin(button\_text.getLocalBounds().left + button\_text.getLocalBounds().width / 2, button\_text.getLocalBounds().top + button\_text.getLocalBounds().height / 2);

button\_text.setPosition(1300 + button.getSize().x / 2, 350 + k \* 50 + button.getSize().y / 2);

click\_button = false;

}

// Визначення функції для повернення кнопки

RectangleShape& square\_buttons\_with\_selection\_of\_displayed\_functions::get\_button() {return button;}

// Визначення функції для повернення текста кнопки

Text& square\_buttons\_with\_selection\_of\_displayed\_functions::get\_button\_text() {return button\_text;}

// Визначення функції для перевірки натискання на кнопку

bool square\_buttons\_with\_selection\_of\_displayed\_functions::isClicked() {return click\_button;}

// Визначення функції для встановлення значення чи натиснута кнопка

void square\_buttons\_with\_selection\_of\_displayed\_functions::setClicked(bool clicked) {click\_button = clicked;}

# Додаток Ї

Хедр файл Graph\_displacement\_and\_compression.h

#ifndef Graph\_displacement\_and\_compression

#define Graph\_displacement\_and\_compression

// Підключення бібліотеки sfml для вграфіки та простору імен sf

#include <SFML/Graphics.hpp>

using namespace sf;

// Створення класу кнопок для переміщення та стиснення

class round\_buttons\_for\_moving\_and\_compressing\_graphs {

// Створюємо приватні змінні

private:

// Кнопка

CircleShape button;

// Змінна для зберігання умови натиснення на кнопку

bool click\_button;

// Відкриття публічного доступу

public:

// Оголошення конструктора з двома параметрами

round\_buttons\_for\_moving\_and\_compressing\_graphs(Font& font, const int& k);

// Функція для повернення кнопки

CircleShape& get\_button();

// Функція для перевірки натискання на кнопку

bool isClicked();

// Функція для встановлення значення чи натиснута кнопка

void setClicked(bool clicked);

};

#endif

Сурс файл graph\_displacement\_and\_compression.cpp

// Підключення хедр файлу Graph\_displacement\_and\_compression, де написані функції, які треба визначити, надати тіло.

#include "Graph\_displacement\_and\_compression.h"

// Визначення конструктора з двома параметрами

round\_buttons\_for\_moving\_and\_compressing\_graphs::round\_buttons\_for\_moving\_and\_compressing\_graphs(Font& font, const int& k) {

// Встановлення розмірру позиції та кольору кнопки

button.setRadius(15.f);

button.setPosition(1300, 350 + k \* 50);

button.setFillColor(Color::Transparent);

button.setFillColor(Color::White);

button.setOutlineColor(Color::Black);

button.setOutlineThickness(2);

click\_button = false;

}

// Визначення функції для повернення кнопки

CircleShape& round\_buttons\_for\_moving\_and\_compressing\_graphs::get\_button() {return button;}

// Визначення функції для перевірки натискання на кнопку

bool round\_buttons\_for\_moving\_and\_compressing\_graphs::isClicked() {return click\_button;}

// Визначення функції для встановлення значення чи натиснута кнопка

void round\_buttons\_for\_moving\_and\_compressing\_graphs::setClicked(bool clicked) {click\_button = clicked;}

# Додаток Й

Скрипт компіляції run.bat

#!/bin/bash

# Відкривання командної строки в данному середовищі

# Компіляція програми та встановлення шляху до OpenGL

g++ main.cpp files.cpp error\_windows.cpp text\_input.cpp functions.cpp f\_x.cpp f\_cos.cpp f\_sin.cpp f\_rand.cpp additional\_windows.cpp additional\_functions.cpp graph\_display\_switch.cpp graph\_displacement\_and\_compression.cpp -o Graph\_of\_a\_function -I/opt/homebrew/include -L/opt/homebrew/lib -lsfml-graphics -lsfml-window -lsfml-system -lglfw -framework OpenGL

# Відкривання

./Graph\_of\_a\_function