МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

Національний технічний університет України

"Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Програмування. Частина 1. Основи програмування

Лабораторна робота №5

Тема: Реалізація алгоритмів з використанням функцій та покажчиків

Варіант №11

Виконав: Коваль Б.А.

студент групи ІС-31

Перевірив:

Драган М.С.

Київ-2023

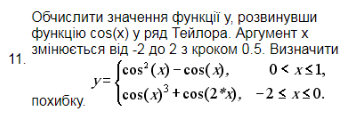
МЕТА :

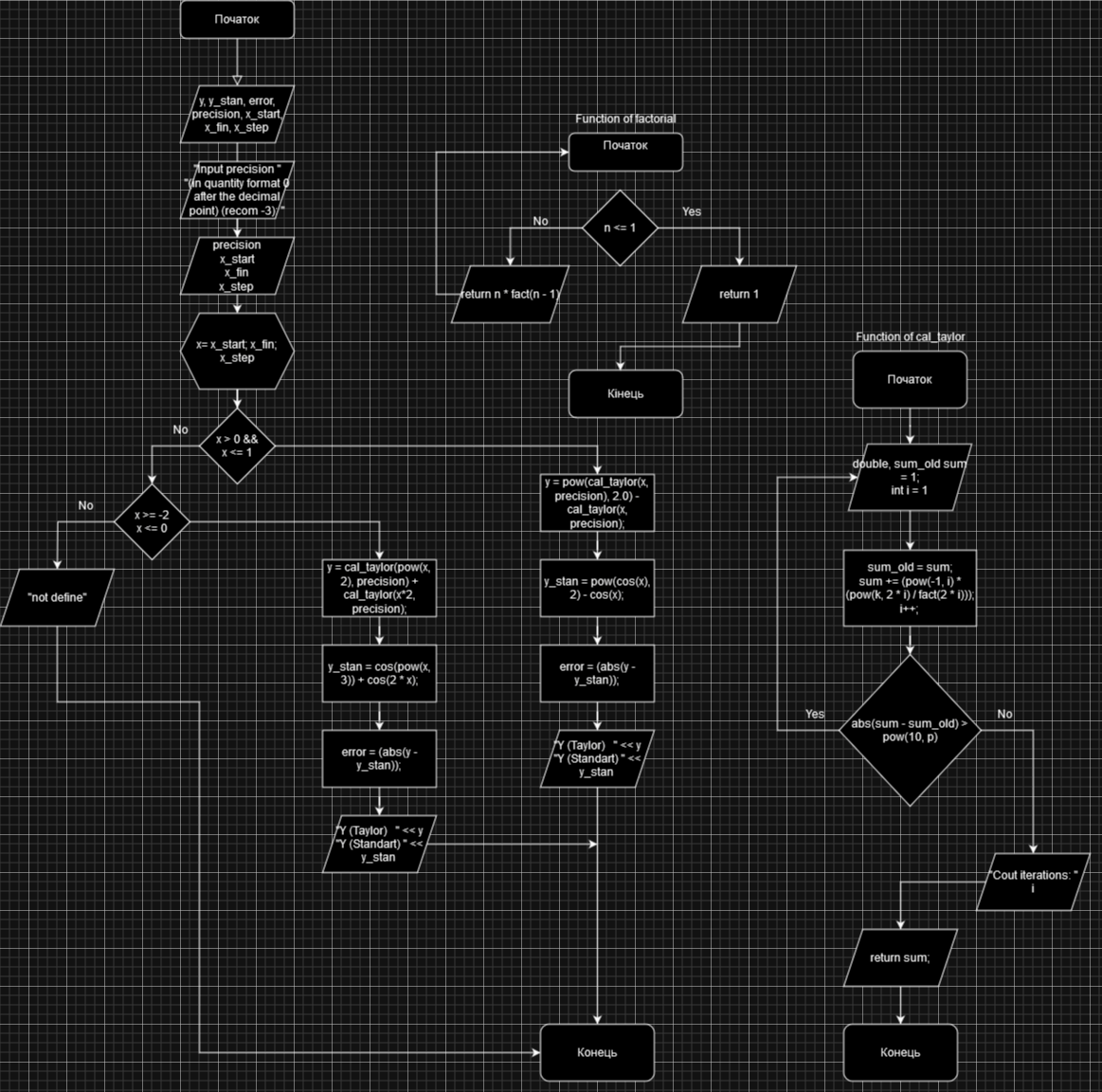
1. Ознайомитися з особливостями посилальних типів даних;

2. Опанувати технологію застосування посилальних типів даних;

3. Навчитися розробляти алгоритми та програми із застосуванням посилальних типів даних.

ЗАВДАННЯ №1





#include <iostream>

#include <cmath>

using namespace std;

float y, y\_stan, error, precision, x\_start, x\_fin, x\_step;

//y - Сама функція для обрахунку за формулою Тейлора

//y\_stan - Звичайно підрахована функція

//x\_start - аргумент функції

//error - похибка

//precision - точність

int fact(int n) {

if (n <= 1)

return 1;

return n \* fact(n - 1);

}

float cal\_taylor(float k, float p) {

double sum = 1;

double sum\_old;

int i = 1;

do {

sum\_old = sum;

sum += (pow(-1, i) \* (pow(k, 2 \* i) / fact(2 \* i)));

i++;

} while(abs(sum - sum\_old) > pow(10, p));

cout << "Cout iterations: " << i << endl;

return sum;

}

int main()

{

cout << "Input precision " << endl << "(in quantity format 0 after the decimal point) (recom -3): ";

cin >> precision;

cout << "Input x [start] (recom -2): ";

cin >> x\_start;

cout << "Input x [finish] (recom = 2): ";

cin >> x\_fin;

cout << "Input x [step] (recom = 0.5): ";

cin >> x\_step;

for (float x = x\_start; x <= x\_fin; x += x\_step) {

cout << endl << endl;

cout << "X: " << x << endl;

if (x > 0 && x <= 1) {

y = pow(cal\_taylor(x, precision), 2.0) - cal\_taylor(x, precision);

y\_stan = pow(cos(x), 2) - cos(x);

error = (abs(y - y\_stan));

cout << "Y (Taylor) " << y << endl;

cout << "Y (Standart) " << y\_stan << endl;

cout << "Error:" << error << " (%): " << error \* 100 << "%" << endl << endl;

}

else if (x >= -2 && x <= 0) {

y = cal\_taylor(pow(x, 2), precision) + cal\_taylor(x\*2, precision);

y\_stan = cos(pow(x, 3)) + cos(2 \* x);

error = (abs(y - y\_stan));

cout << "Y (Taylor) " << y << endl;

cout << "Y (Standart) " << y\_stan << endl;

cout << "Error:" << error << " (%): " << error \* 100 << "%" << endl << endl;

}

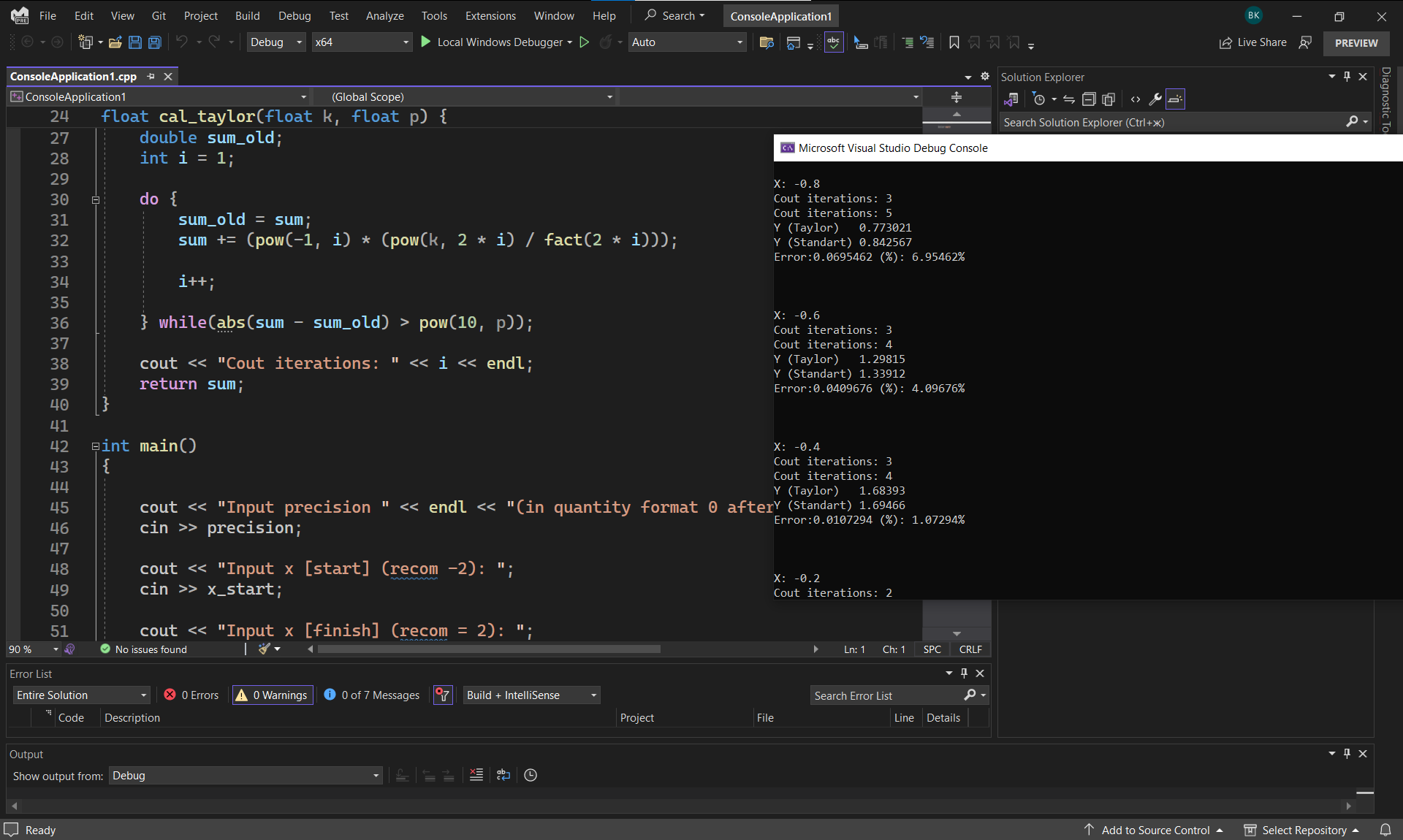
else {

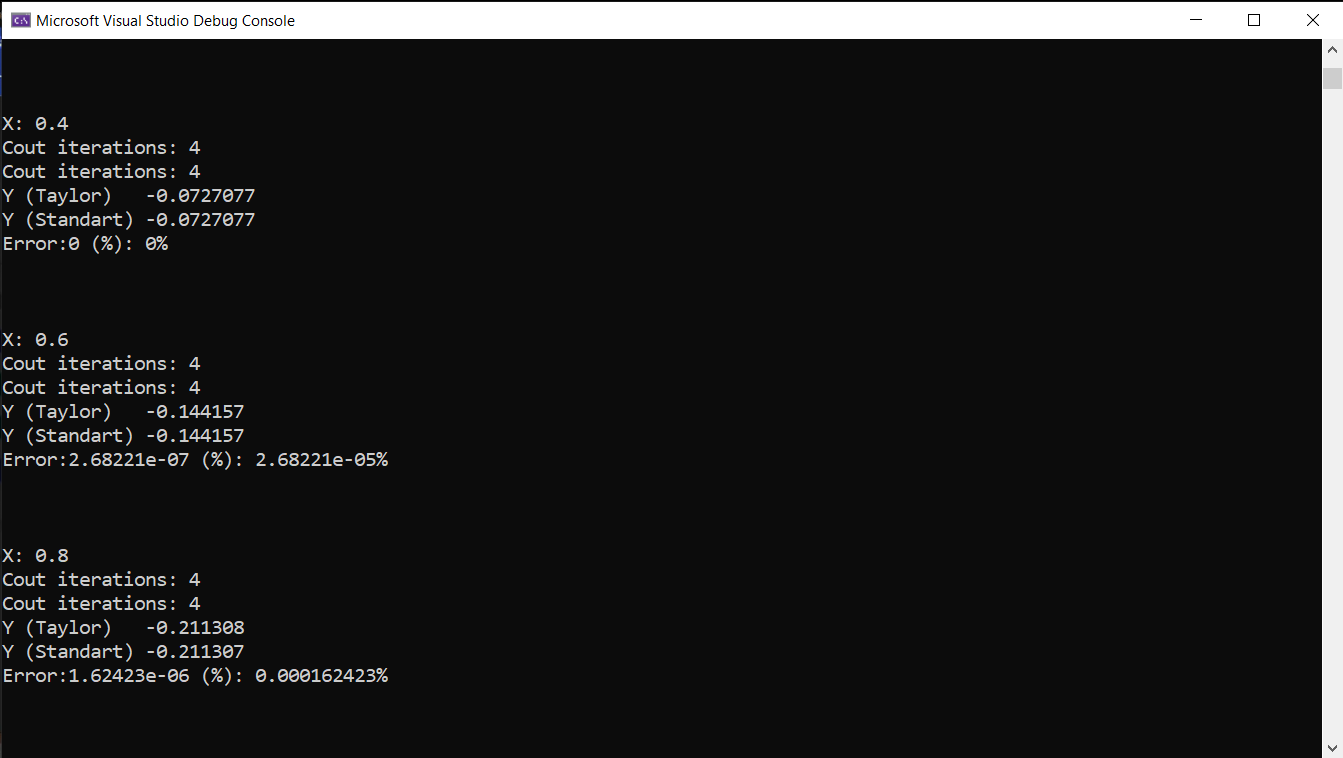
cout << "not define" << endl;

}

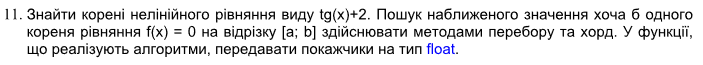
}

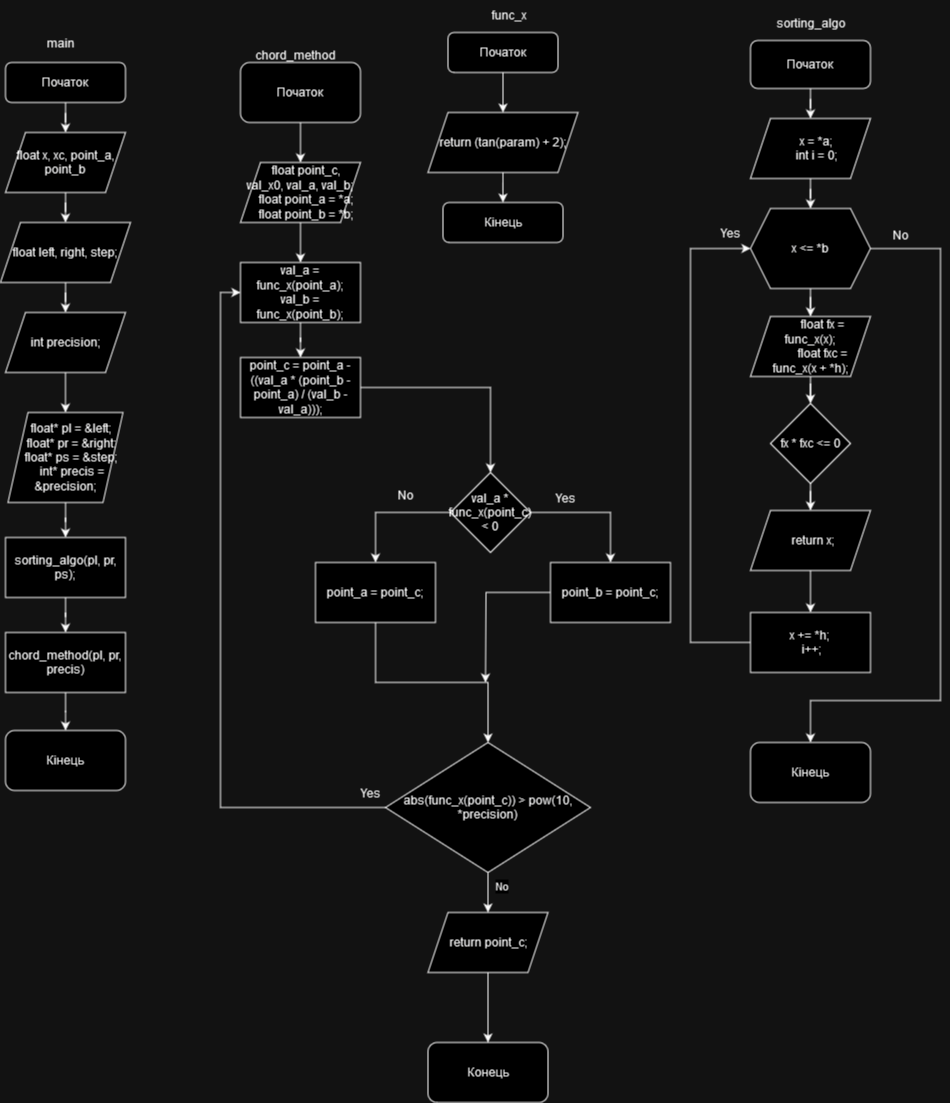
}





ЗАВДАННЯ №2





#include <iostream>

#include <cmath>

using namespace std;

float x, xc, point\_a, point\_b;

float func\_x(float param) {

return (tan(param) + 2);

}

float chord\_method(float\* a, float\* b, int\* precision) {

cout << endl << "#2 Chord Algoritm";

float point\_c, val\_x0, val\_a, val\_b;

float point\_a = \*a;

float point\_b = \*b;

int i = 0;

do {

val\_a = func\_x(point\_a);

val\_b = func\_x(point\_b);

point\_c = point\_a - ((val\_a \* (point\_b - point\_a) / (val\_b - val\_a)));

if (val\_a \* func\_x(point\_c) < 0)

point\_b = point\_c;

else

point\_a = point\_c;

i++;

} while (abs(func\_x(point\_c)) > pow(10, \*precision) && i < 100);

cout << endl << "Found for " << i << " iterations";

return point\_c;

}

float sorting\_algo(float\* a, float\* b, float\* h) {

cout << endl << "#1 Sorting Algoritm";

x = \*a;

int i = 0;

while (x <= \*b) {

float fx = func\_x(x);

float fxc = func\_x(x + \*h);

if (fx \* fxc <= 0) {

cout << endl << "Found for " << i << " iterations" << endl;

cout << endl << "x: " << x << " fx: " << fx << endl;

cout << "x: " << x + \*h << " fxc: " << fxc << endl;

return x;

}

x += \*h;

i++;

}

}

int main()

{

float left, right, step;

int precision;

cout << "Input [a, b] (a - start, b - finish, c - step)" << endl;

cout << "Start (Recommend 2): ";

cin >> left;

cout << "Finish: ";

cin >> right;

cout << "Step (Affects the accuracy of algorithm #1): ";

cin >> step;

cout << endl << "Precision (Format coutns zero after coma)" << endl << "Affects the accuracy of algorithm #2): ";

cin >> precision;

float\* pl = &left;

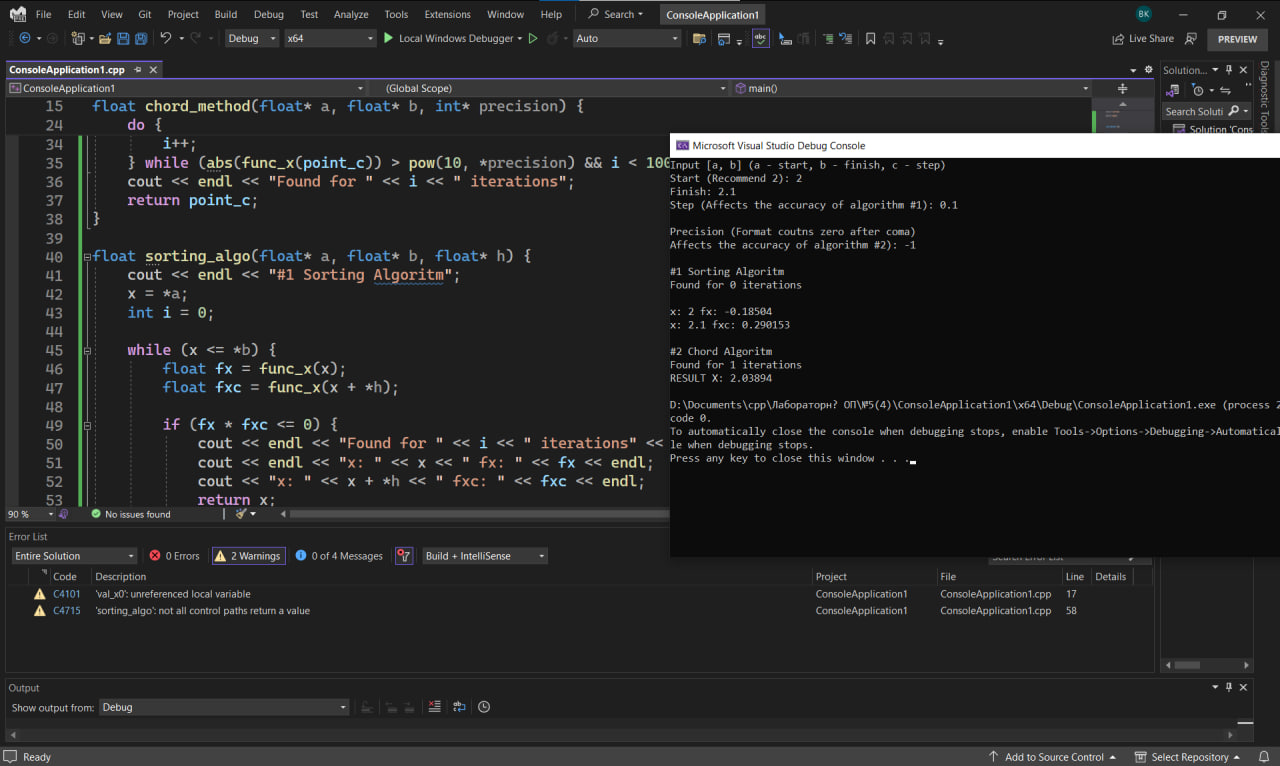
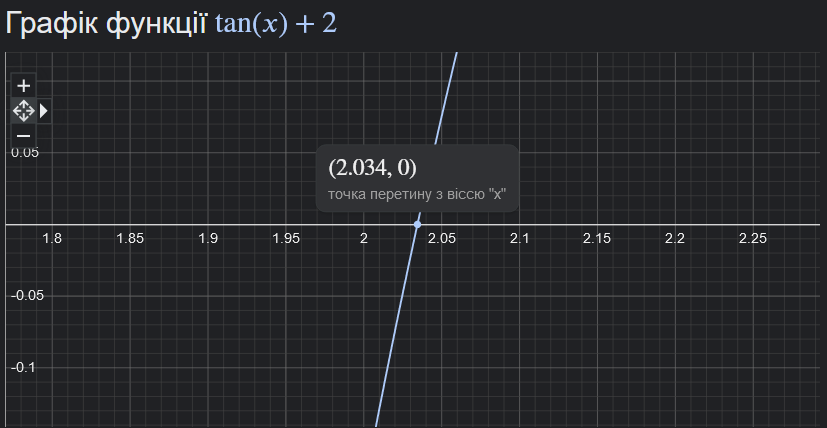
float\* pr = &right;

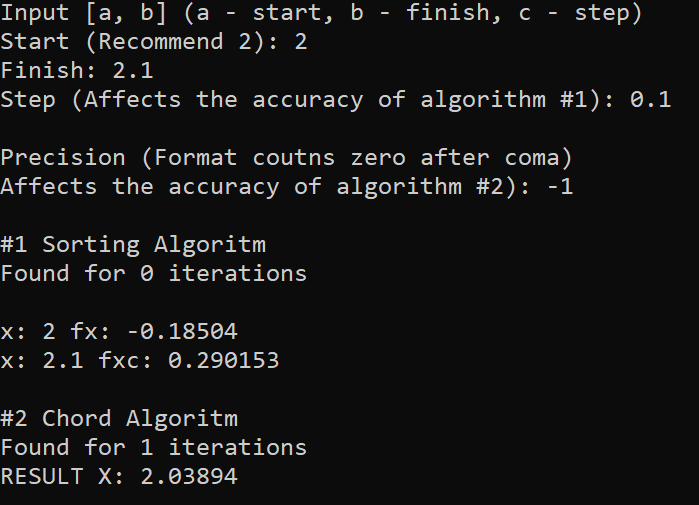
float\* ps = &step;

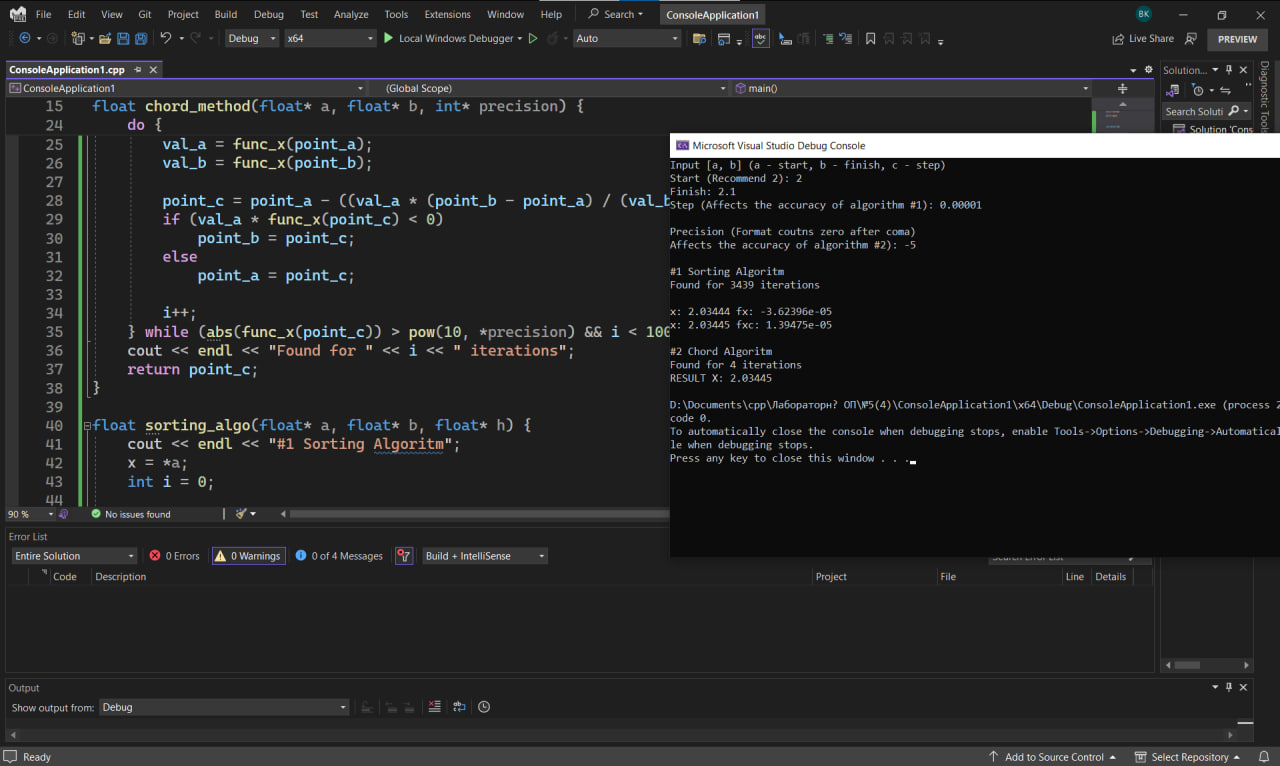
int\* precis = &precision;

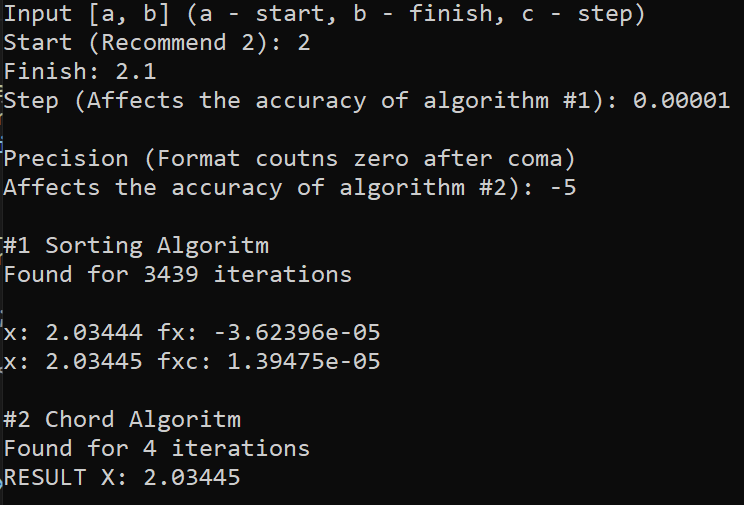
sorting\_algo(pl, pr, ps);

cout << endl << "RESULT X: " << chord\_method(pl, pr, precis) << endl;

}







КОНТРОЛЬНІ ЗАПИТАННЯ:

1. ***Посилальні типи даних (reference types)*** - це типи даних в програмуванні, які не зберігають самі дані, а лише посилаються на місце, де знаходяться фактичні дані. Посилальні типи дозволяють працювати з даними без їхньої копіювання і дозволяють створювати посилання на існуючі об'єкти.

2. ***Доступ до значення змінної*** через посилальний тип даних відбувається шляхом використання посилання (reference). Ви замість копіювання значення отримуєте посилання на оригінальну змінну і можете працювати з нею.

3. ***Щоб оголосити покажчик*** на певний тип та посилання на змінну

int x = 10;

int \*ptr = &x; // Оголошення покажчика на змінну x

int &ref = x; // Оголошення посилання на змінну x

4. ***Ініціалізація покажчика*** та посилання полягає в призначенні їм адреси змінної, на яку вони вказують.

5. ***Звернення до неініціалізованого покажчика*** може призвести до невизначеного поведінки програми або викликати помилку. Ініціалізуйте покажчик перед його використанням.

6. ***Покажчику можна присвоювати адреси*** інших змінних або об'єктів. Значення покажчика - це адреса в пам'яті, на яку він вказує.

7. ***Для покажчиків можна виконувати операції***, такі як додавання і віднімання цілих чисел (арифметика покажчиків), звертання до значення, порівняння і т. д.

8. ***"Розіменування покажчика" означає отримання значення***, на яке вказує покажчик. В мовах програмування це зазвичай робиться з використанням оператора "\*".

9***. Значення покажчика змінюється при додаванні чи відніманні*** цілого числа на адресу, на яку він вказує. Наприклад, якщо ptr - покажчик на int, то ptr + 1 вказуватиме на наступний int в пам'яті.

10. ***Покажчики типу `void \*`*** - це спеціальні покажчики, які не мають типу даних і можуть вказувати на будь-який тип. Вони корисні, коли вам потрібно зберігати адресу об'єкта без знання його типу.

11. Покажчики на функції використовуються для передачі функцій як аргументів інших функцій, для створення таблиць функцій і багато іншого. Вони дозволяють програмі бути більш гнучкою і динамічною в обробці функцій.

ВИСНОВОК:

Під час виконання цієї роботи я успішно ознайомився з особливостями використання посилальних типів даних, опанував технологію їх застосування та навчився розробляти алгоритми та програми з їх використанням. Важливо відзначити, що знання цих концепцій є важливим етапом у моєму розвитку навичок програмування.

Посилальні типи даних дозволяють мені ефективно маніпулювати даними та використовувати їх у алгоритмах. Вони дозволяють передавати адреси пам'яті, замість копіювання великих об'єктів, що може значно зменшити обсяг пам'яті та оптимізувати роботу програми.

Також використання функцій та покажчиків дозволяє мені розробляти більш гнучкі та модульні програми, сприяючи більш легкій розширюваності та підтримці коду.

**В результаті роботи програм**

**№1 *(з параметрами start = -2, finish = 2, precision = -3, step = 0.2)***

**Отримав результат:**

**Х = -1**

**За алгоритмом Тейлера:** Y (Taylor) 0.124148,

**За підстановкою:** Y (Standart) 0.124156.

**Похибка за алгоритмом Тейлера становила:** 0.000826269%.

**Х = -0.8**

**За алгоритмом Тейлера:** Y (Taylor) 0.772895,

**За підстановкою:** Y (Standart) 0.842567.

**Похибка за алгоритмом Тейлера становила:** 6.96719%.

**№2 *(з параметрами start = 2, finish = 2.1, step = 0.0001, precision = -5)***

**Отримав результат:**

**За алгоритмом перебору:** a = 2.03436 b = 2.03446,

**За алгоритмом хорд:** x = 2.03445, що означає коректність роботи програми.

**Точка перетину з абцисою:** ~2.034.