Національний технічний університет України

«Київський політехнічний інститут»

Кафедра автоматизованих систем обробки інформації та управління

Звіт про виконання лабораторної роботи №6 з дисципліни

«Алгоритмизація та програмування»

Перевірено: Ковалюк Т.В. Виконав ст. 1 курсу ФОІТ

Проскура С.Л. гр. ІС-52

Дорошенко А.В.

Київ 2015

## Лабораторна робота 6

# **Покажчики**

Варіант № 9

*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

|  |  |
| --- | --- |
| Знайти корені нелінійного рівняння виду . Пошук наближеного значення хоча б одного кореня рівняння f(x) = 0 на відрізку [a; b] здійснювати методами перебору та хорд. У функції, що реалізують алгоритми, передавати покажчики на тип float. |  |

### Блок-схема алгоритму



Рис.1 (Алгоритм знаходження кореня методом перебору)



Рис.2 (Головний алгоритм)



Рис.3 (Алгоритм знаходження кореня методом хорд)

### Код програми

#include <iostream>

#include <math.h>

using namespace std;

double step = 0.0001; //крок зміни для методу перебору

double sx1 = 0, //перший корінь знайдений методом перебору

sx2 = 0; //другий корінь знайдений методом перебору

double chx1 = 0, //перший корінь знайдений методом хорд

chx2 = 0; //другий корінь знайдений методом хорд

double a = 0; //нижня границя проміжка

double b = 0; //верхня границя проміка

double error = 0.0001; //точність для методу хорд

double func(double \*par) //--------обчислення значення функції-----------

{return (\*par) \* (\*par) - exp(-(\*par) \* (\*par));}

double search(double \*a, double \*step, double \*x) //-----знаходження кореня методом перебору-----

{

\*x = \*a + \*step;

while (func(a) \* func(x) > 0)

{

\*x += \*step;

}

return \*x;

}

double chord(double \*a, double \*b, double \*x) //------знаходження кореня методом хорд------

{

while (fabs(func(x)) >= error)

{

\*x = \*a + fabs(func(a) / (func(a) - func(b)))\*(\*b - \*a);

if (func(a) < 0)

a = x;

else

b = x;

}

return \*x;

}

int main()

{

int cont = 0;

do

{

cout << "Please, enter lower and upper bound" << endl;

cin >> a >> b;

double\* point\_sx1 = &sx1; //покажчик на перший корінь знайдений методом перебору

double\* point\_sx2 = &sx2; //покажчик на другий корінь знайдений методом перебору

double\* point\_chx1 = &chx1; //покажчик на перший корінь знайдений методом хорд

double\* point\_chx2 = &chx2; //покажчик на другий корінь знайдений методом хорд

double\* point\_step = &step; //покажчик на крок зміни для методу перебору

double\* point\_a = &a; //покажчик на нижню границю проміжка

double\* point\_b = &b; //покажчик на верхню границю проміжка

if (func(point\_a) \* func(point\_b) < 0)//умова коли рівняння на проміжку має один корінь

{

sx1 = search(point\_a, point\_step, point\_sx1);

cout << "root search = " << sx1 << endl;

chx1 = chord(point\_a, point\_b, point\_chx1);

cout << "chord root = " << chx1 << endl;

}

else if (func(point\_a) \* func(point\_b) >= 0)

{

if (func(point\_a) >= 0 && a < 0)//два корені

{//-----відокремлення коренів-----

double asec, //нижня границя проміжка де знаходиться 2 корінь

bsec; //верхня границя проміжка де знаходиться 2 корінь

double\* point\_asec = &asec; //покажчик на нижню границю

double\* point\_bsec = &bsec; //покажчик на верхню границю

\*point\_asec = 0;

\*point\_bsec = \*point\_b;

\*point\_b = 0;

cout << "first root" << endl; //-----уточнення коренів-----

sx1 = search(point\_a, point\_step, point\_sx1);

cout << "search root = " << sx1 << endl;

chx1 = chord(point\_a, point\_b, point\_chx1);

cout << "chord root = " << chx1 << endl;

cout << "second root" << endl;

sx2 = search(point\_asec, point\_step, point\_sx2);

cout << "search root = " << sx2 << endl;

chx2 = chord(point\_asec, point\_bsec, point\_chx2);

cout << "chord root = " << chx2 << endl;

}

Else cout << "There are not roots" << endl; //інакше коренів немає

}

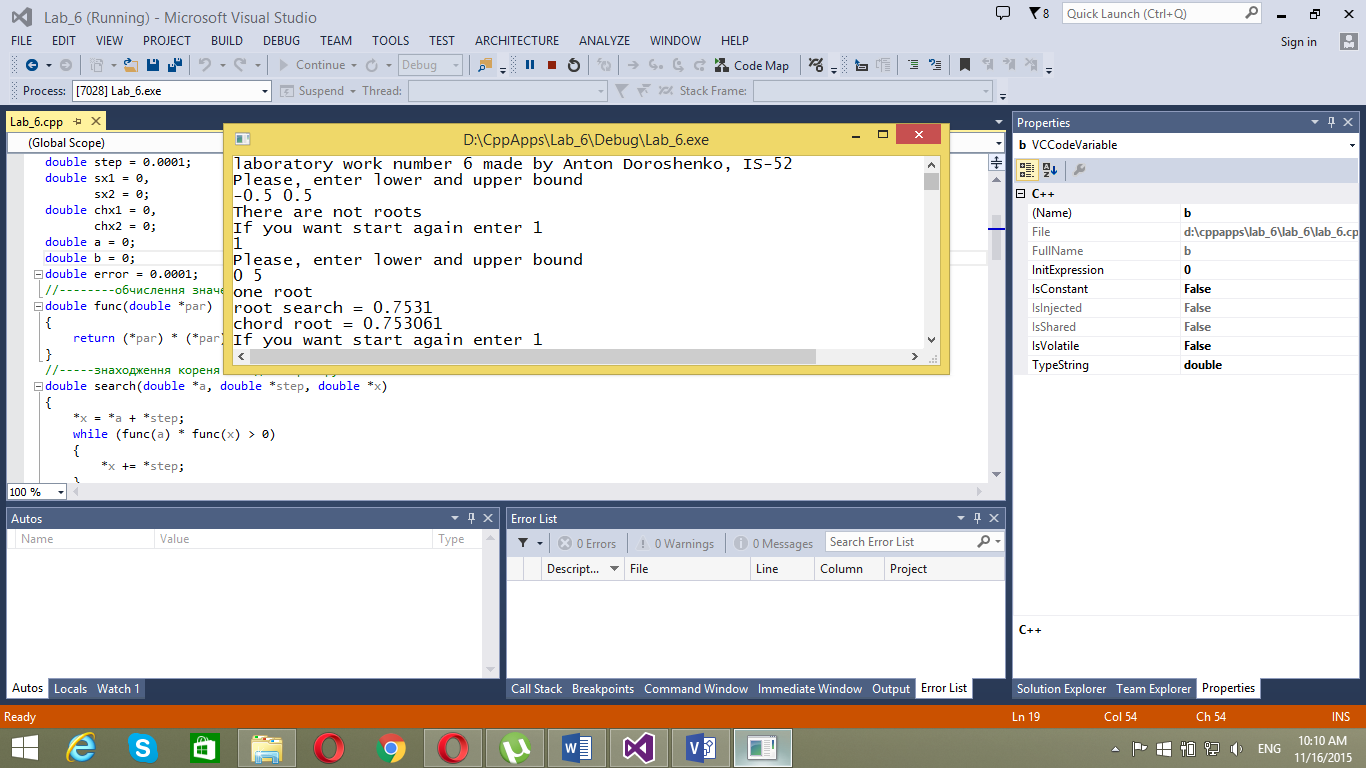
cout << "If you want start again enter 1" << endl;

cin >> cont;

} while (cont == 1);

}

1. **Screen Shot результатів**



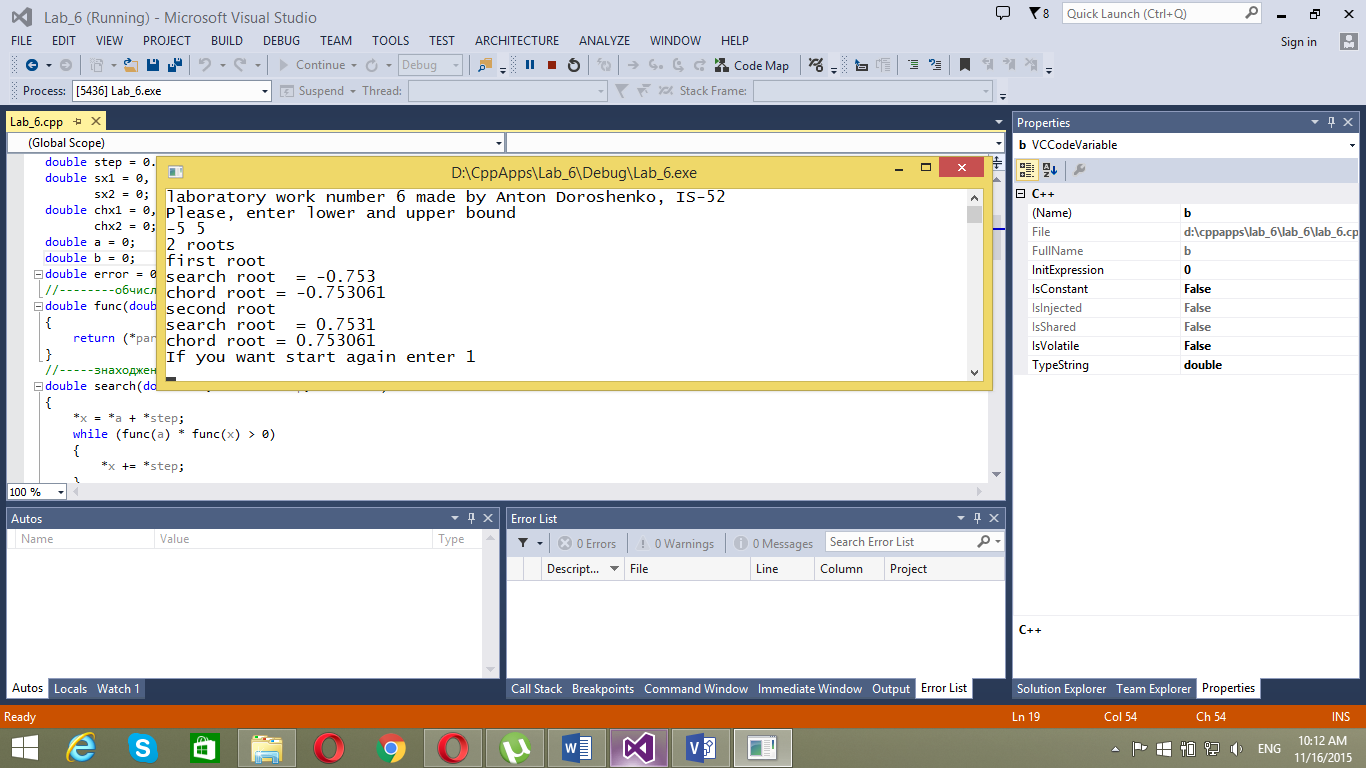


Рис.4 (Screen Shot результатів)

1. **Аналіз результатів**

Як ми бачимо зі ScreenShot’ів, ми ввели 3 варіанти даних.

1. Якщо нижня границя проміжка дорівнює -0.5, а верхня 0.5, то рівняння має 1 корінь що буде дорівнювати -0.753(метод перебору) і -0.753061 (метод хорд).
2. Якщо нижня границя проміжка дорівнює 0, а верхня 5, то рівняння не має коренів.
3. Якщо нижня границя проміжка дорівнює -5, а верхня 5, то рівняння має 2 корені. Перший буде дорівнювати -0.753(метод перебору) і -0.753061 (метод хорд), другий буде дорівнювати 0.7531(метод перебору) і 0.753061 (метод хорд).
4. **Висновок**

Програма працює правильно, про що свідчить аналіз результатів та ScreenShot’и. Програма коректно обчислює корені на різних проміжках. В програмі застосовуються визначники.