Національний технічний університет України

«Київський політехнічний інститут»

Кафедра автоматизованих систем обробки інформації та управління

Звіт про виконання лабораторної роботи №3 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 3

# **Циклічні обчислювальні процеси**

Варіант № 9

*Мета:*

1. Вивчити особливості циклічних обчислювальних процесів
2. Опанувати технологію використання операторів циклів
3. Навчитися розробляти алгоритми та програми циклічних процесів

### Завдання

Обчислити суму ряду, елемент котрого розраховується за формулою:

|  |  |
| --- | --- |
|  |  |

Значення похибки задати самостійно.

### Блок-схема алгоритму



*Рис. 1* (Головний алгоритм)

1. Алгоритм для обчисленнь і виведення поточних результатів

 *Рис.2* (Алгоритм для обчисленнь і виведення поточних результатів)

### Код програми

//Laboratory work

//written by student of the first curse

//of the group IC-52

//Anton Doroshenko

//2015.10.25

//==========================================================================

#include <iostream>

#include <math.h>

#include <iomanip>

using namespace std;

double calculating(double error);

const double minerror = 0.00015; // мінімальна точність обчисленнь

int main()

{

cout << "Lab\_3, IS-52, variant 9, made by Anton Doroshenko " << endl;

cout << "calculate the sum of row" << endl;

double error; // похибка

int cont; // параметр умови продовження обчисленнь

do

{

cout << "Please enter 'error' " << endl;

cin >> error;

if (error >= minerror) // умова запобігання переповнення комірки пам'яті

{

double result = calculating(error); // результат обчисленнь

cout << "result = " << result << endl;

}

else

cout << "overflow because error is very small \nPlease try again" << endl;

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

cin >> cont;

} while (cont == 1); // умова продовження обчисленнь

}

double calculating(double error) // функція для обчисленнь

{

double result = 0;

cout << "x" << setw(10) << "k" << setw(20) << "sum" << setw(20) << "result" << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

for (int x = 1; x <= 5; x++)

{ // х - параметр зовнішньої функції

double sum = 0; // внутрішня сума

double divsn; // елемент ряду (дріб)

int k = 1; //параметр внутрішньої суми

int denum = 0; //знаменник дробу

do

{

//-----обчислення чисельника дробу-----

double numenator = pow(-1, k) \* pow(x, (k - 1));

//-----обчислення знаменника дробу-----

int i = 2\*k; // лічильник

int mul = 1; // доданок знаменника

while (k <= i)

mul \*= i--;

denum += mul;

//-----обчислення дробу і внутрішньої суми-----

divsn = ((double) numenator) / ((double) denum);

sum += divsn;

//-----виведення поточних розрахунків-----

cout << x << setw(10) << k << setw(20) << sum << setw(20) << (result+sum) << endl;

k++;

} while (fabs(divsn) > error); //поки не досягнуто заданої точності

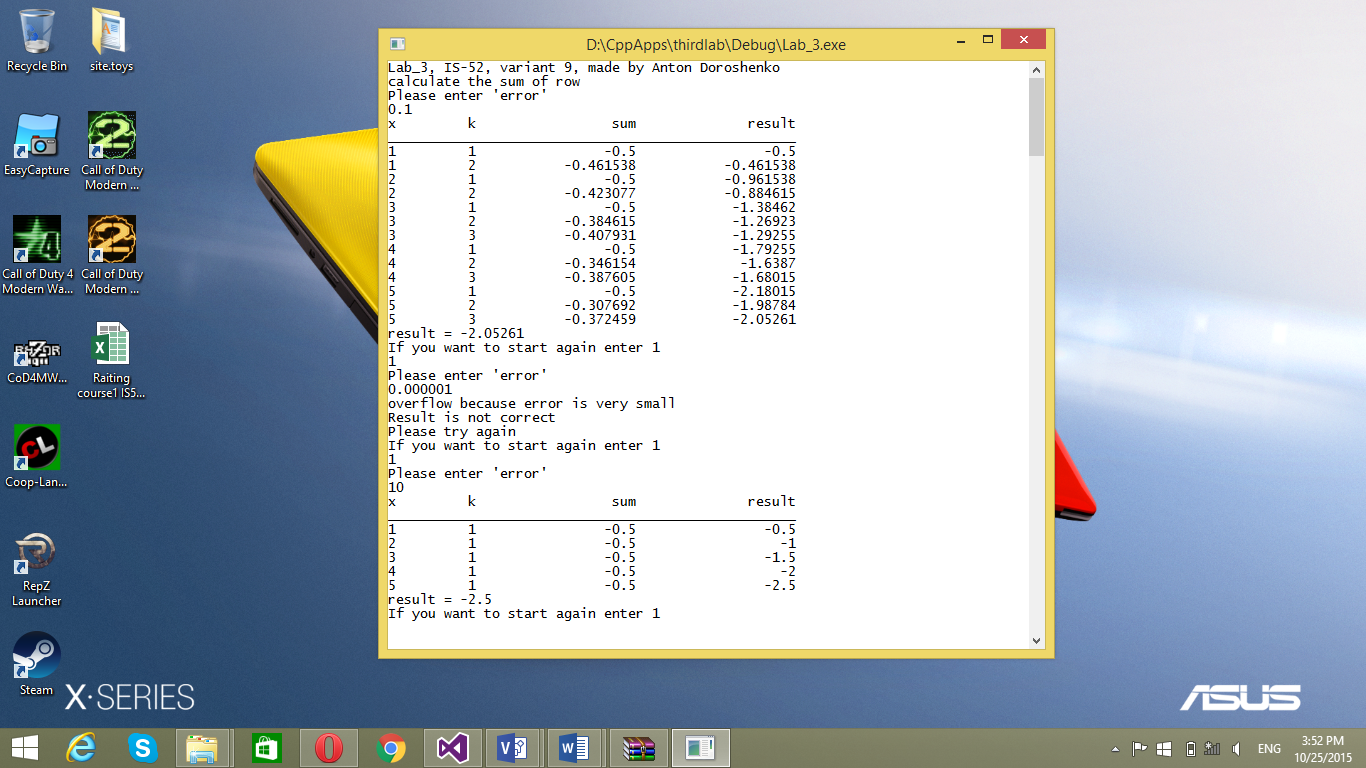
result += sum; //остаточний результат

}

return result;

}

1. **Screen Shot результатів**



*Рис.3*

1. **Аналіз результатів**

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

1. Якщо error = 0.1, то результат буде дорівнювати -2.05261 .
2. Якщо error = 0.000001, то програма запобігає переповненню комірки пам’яті через дуже малу похибку і виводить коректне повідомлення.
3. Якщо error = 10, то результат буде дорівнювати -2.5 .
4. **Висновок**

Програма працює правильно, про що свідчить аналіз результатів та ScreenShot’и. Програма захищена від переповнення комірок пам’яті. Програма коректно виводить проміжні результати. Використано циклиз лічильником та постумовою.