MINISTRY OF SCIENCE AND EDUCATION NATIONAL TECHNICAL UNIVERSITY “KHARKIV POLYTECHNIC INSTITUTE”

DEPARTMENT OF SOFTWARE ENGINEERING AND MANAGEMENT INFORMATION TECHNOLOGIES

Report from lab#5

Discipline “Fundamentals of Software Engineering”

Executed by: Student of group 1.КН.201.8г

CHUKWU IRELE AMORIN

Checked by: Senior lecturer

K.V. Melnik

Kharkiv

2018

**Goal: Learning basic principles of testing C++ code Tasks:**

1. Study principles of using functions in C++.

2. Study Exception Handling in C++.

3. Modify the code from lab 2 according to 1 and 2 tasks.

4. Implement unit testing for developed program.

5. Make all necessary actions on xp-dev.com. Show the iteration where you made updating information.

6. Prepare the report of the work

The code:

#include "pch.h"

#include <iostream>

#include <math.h>

using namespace std;

void checkinput(int n)

{

if (n <= 0)

{

throw " incorrect data";

}

}

int getX()

{

int x;

cout << "input x : ";

cin >> x;

return x;

}

int getN()

{

int n;

cout << " \n input n ,n should be greater than 0 'zero': ";

cin >> n;

checkinput(n);

return n;

}

double functionY(int x, int n)

{

int i, j;

double y=0, z;

if (x < 0)

{

for (i = 1; i < n + 1; i++)

{

y = y + (i + x) \* (i + x);

}

return y;

}

else

{

for (i=0; i < n- 1; i++)

{

z = 1;

for (j = 1; j <= n; j++)

{

z = z \* ((x + i) / (i + j));

}

y = y + z;

}

return y;

}

}

int main()

{

try {

cout << "the result of the function of x is : " << functionY(getX(), getN()) << endl; ;

}

catch (const char\* ex)

{

cout << ex << endl;

return -1;

}

catch (...)

{

cout << "the program has an error " << endl;

}

}

Testing code:

#include "stdafx.h"

#include "CppUnitTest.h"

#include "C:\Users\irele\source\repos\Solution1\lab3-individual assignment\lab3-individual assignment.cpp"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace UnitTest

{

TEST\_CLASS(UnitTest1)

{

public:

TEST\_METHOD(functinY)

{

int n = 5;

int x = -2;

double expected = 15;

double actual = functionY(x , n);

Assert::AreEqual(expected, actual);

// TODO: Your test code here

}

TEST\_METHOD(funcY)

{

int n = 5;

int x = 2;

double expected = 1.4755456349206348;

double actual = functionY(x, n);

Assert::AreEqual(expected, actual);

// TODO: Your test code here

}

};

}