НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ ТА СПЕЦІАЛІЗОВАНИХ КОМП’ЮТЕРНИХ СИСТЕМ

**Лабораторна робота №1  
з дисципліни «Алгоритми та методи обчислення»  
на тему: «Обчислення значення функції»**

**Варіант 21**

Виконав:  
студент 3-го курсу,  
гр. КВ-41,  
Яковенко Максим

Київ – 2016

**Завдання для лабораторної роботи**

Для заданого варіанта (табл. 1.1) виконати 3 завдання.

1. Побудувати таблицю залежності довжини ряду *n*, що забезпечує точність функції не меншу за задане значення *eps* у точці *x = (b + a)/*2, від *eps*:

|  |  |  |  |
| --- | --- | --- | --- |
| *eps* | *n* | Абсолютна похибка | Залишковий член |
| 10-2 | 4 | 0.005 | 0.001 |
| ... | ... | ... | ... |
|  |  |  |  |

Значення *eps* змінюється від 10-2 до 10-14 з кроком 10-3.

2. Для *n* (довжина ряду фіксована й дорівнює *n*), отриманого в п.1 при *eps* = 10-8, у точках *xi = a + h********i, h = (b – a)/*10*, i =* 0*, ...,* 10 обчислити абсолютну похибку та залишковий член ряду. Результати подати у вигляді таблиці:

|  |  |  |
| --- | --- | --- |
| *xi* | Абсолютна похибка | Залишковий член |
| 0 | 0.005 | 0.001 |
| ... | ... | ... |
|  |  |  |

3. За допомогою AdvancedGrapher побудувати графік залежності абсолютної похибки від *x* (у логарифмічному масштабі).

**Варіант 21:**

****

**Текст програми**:

**Result.h**

struct Result

{

double f\_x;

int n;

double absEr;

double remT;

Result();

~Result();

Result& operator=(Result& src);

};

**sinx.h**

#pragma once

#define \_USE\_MATH\_DEFINES

#include <cmath>

#include "Result.h"

class Sin

{

private:

enum Func { SIN, COS };

int sign;

Func f;

double cast(double x);

public:

Sin();

~Sin();

Result AccuracyValue(double x, double eps);

Result AbsoluteError(double x, int n);

};

**sinx.cpp**

#include "sinx.h"

#include <iostream>

Sin::Sin() {}

Sin::~Sin() {}

double Sin::cast(double x)

{

const double d\_pi = 2\*M\_PI;

sign = 1;

f = SIN;

if(x > 0)

while(x >= d\_pi) x -= d\_pi;

else

{

while (x <= d\_pi) x += d\_pi;

if (x >= d\_pi) x -= d\_pi;

};

if (x >= M\_PI)

{

sign = -1;

x -= M\_PI;

};

if (x >= M\_PI\_2) x = M\_PI - x;

if (x >= M\_PI\_4)

{

f = COS;

x = M\_PI\_2 - x;

};

return x;

}

Result Sin::AccuracyValue(double x, double eps) {

double U, result = 0;

int k;

double lib\_sin = sin(x);

Result Res;

x = cast(x);

U = x;

if (f == SIN)

{

for(k = 1; abs(U) >= eps; ++k)

{

result += U;

U \*= -x\*x/(2\*k \* (2\*k + 1));

}

}

else

{

U = 1;

for(k = 1; abs(U) >= eps; ++k)

{

result += U;

U \*= -x\*x/(2\*k \* (2\*k - 1));

}

}

result \*= sign;

Res.absEr = abs(result - lib\_sin);

Res.n = k;

Res.remT = U;

Res.f\_x = result;

return Res;

}

Result Sin::AbsoluteError(double x, int n) {

double U, result = 0;

double lib\_sin = sin(x);

int k = 1;

Result Res;

Res.remT = 0;

if(n > 0)

{

x = cast(x);

U = x;

x \*= x;

if (f == SIN)

{

while(--n)

{

result += U;

U \*= -x/(2\*k \* (2\*k + 1));

k++;

}

}

else

{

U = 1;

while(--n)

{

result += U;

U \*= -x/(2\*k \* (2\*k - 1));

k++;

}

}

result \*= sign;

Res.remT = U;

}

Res.absEr = abs(result - lib\_sin);

Res.n = k;

Res.f\_x = result;

return Res;

}

**Result.cpp**

#include "Result.h"

Result& Result::operator=(Result& src)

{

if (this == &src) return src;

f\_x = src.f\_x;

n = src.n;

absEr = src.absEr;

remT = src.remT;

return src;

}

Result::~Result() {};

Result::Result() {};

**main.cpp**

#include <iostream>

#include <fstream>

#include "sinx.h"

using namespace std;

int main() {

ofstream tbl("table.csv");

const double a = -0.33;

const double b = 7.4;

const double h = (b-a)/10;

double eps;

double x = (a+b)/2;

Sin \*sinx = new Sin();

Result result;

cout << "\t\t TABLE 1\t" << endl;

cout<<'|'<<" Eps\t"<< '|' <<" n\t"<< '|' <<" Absolute Error "<< '|' <<" Remainder term"<<endl;

cout << "-------------------------------------------------"<<endl;

for(eps = 1e-2; eps >= 1e-14; eps \*= 1e-3)

{

result = sinx->AccuracyValue(x, eps);

cout<< '|' <<eps<<"\t"<< '|' <<" "<<result.n<<"\t"<< '|' <<" "<<result.absEr<<"\t "<< '|' <<result.remT<<endl;

}

cout<<"\n"<<endl;

int n = sinx->AccuracyValue(x, 1e-8).n;

cout << "\t\t TABLE 2\t" << endl;

cout<< '|' <<" Xi\t"<< '|' <<" Absolute Error"<< '|' <<" Remainder term"<<endl;

cout << "----------------------------------------" << endl;

for (int i = 0; i <= 10; ++i)

{

x = a + h\*i;

result = sinx->AbsoluteError(x, n);

tbl << x << ';' << result.absEr << ';' << endl;

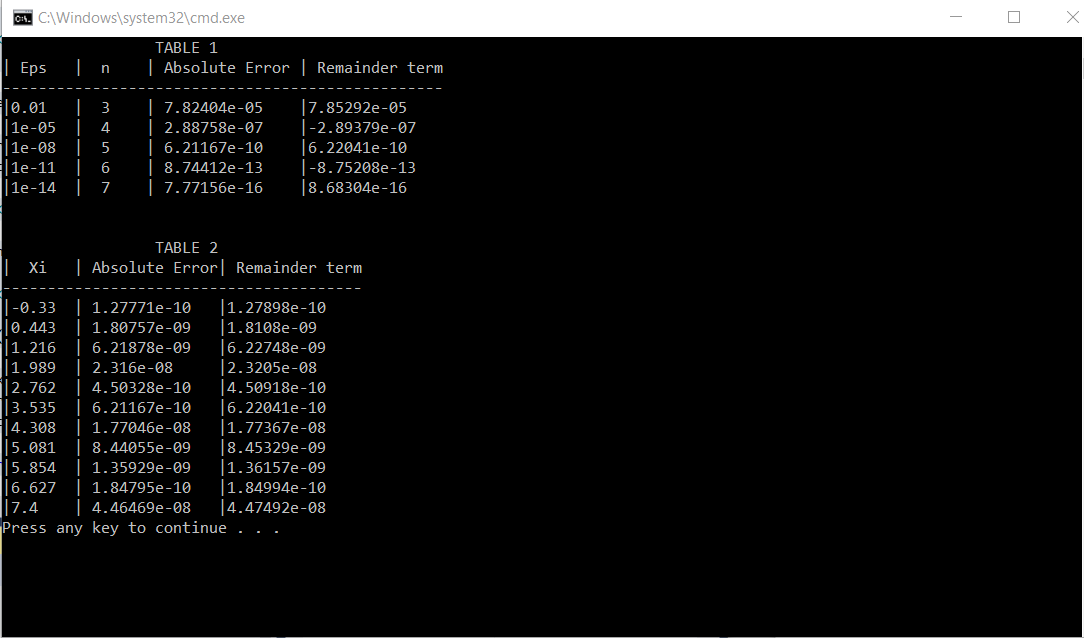
cout<< '|' <<x<<"\t"<< '|' <<" "<<result.absEr<<"\t"<< '|' <<result.remT<<endl;

}

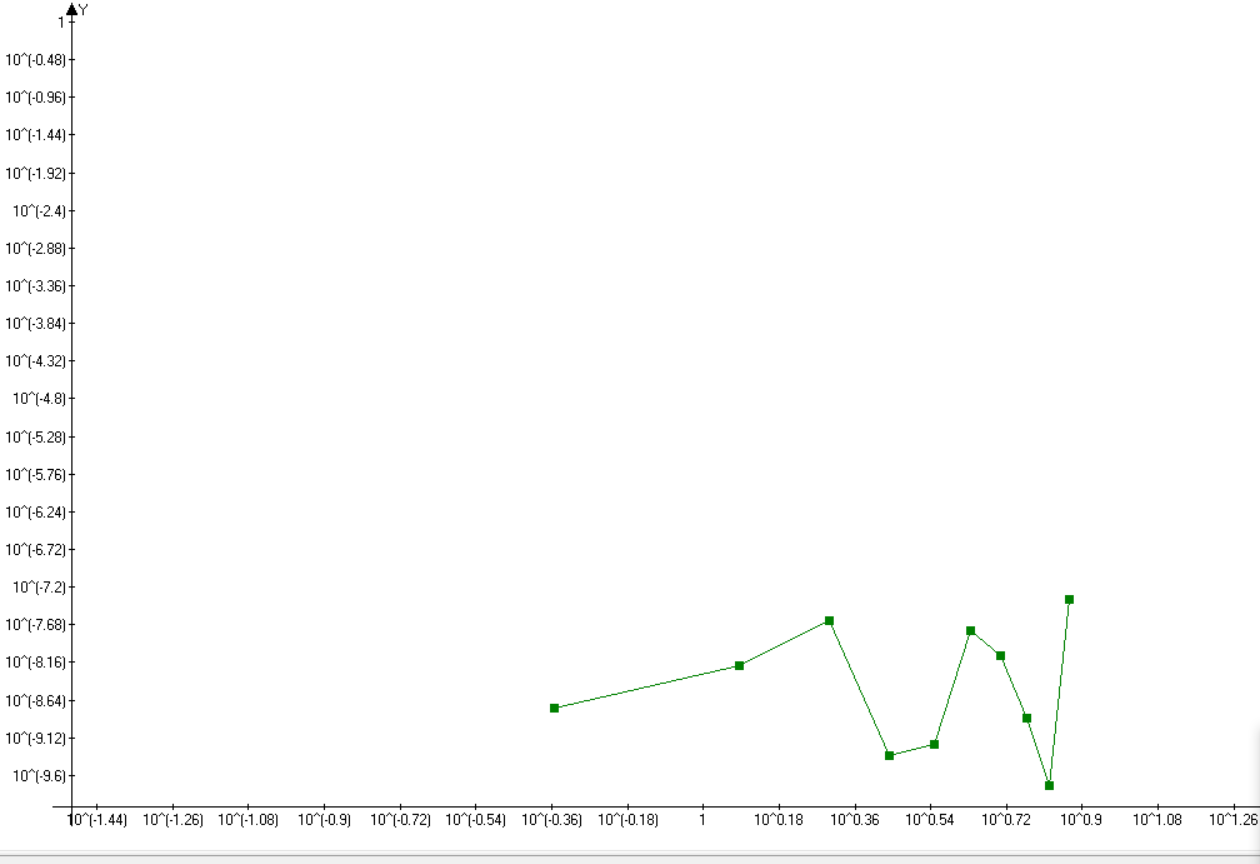
return 0;

}

**Таблиці результатів:**

****

**Графік:**

****

**Висновки:**

В ході виконання лабораторної роботи ми обчисляли наближене значення функції sin(x) викорестовуючи розкладання в ряд Маклорена.

Перше завдання стосувалося побудови таблиці залежності довжини ряду, що забеспечує точність функції не меншу за задане значення eps, від eps. Отриманні результати свідчать про те, що зі збільшенням точності(зменшенням значення допустимої похибки) збільшується кількість членів ряду Маклорена, необхідна для отримання результату з заданою точністю.

Друге завдання стосувалося обрахунку залежності абсолютної похибки від значення аргументу при заданій довжині ряду. Характер отриманної залежності є нелінійним та періодичним, що можна пояснити властивостями функції sin(x).