

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 $x_i = a + h \cdot i$, $h = (b - a)/10$, $i = 0, \dots, 10$ обчислити абсолютну похибку та залишковий член ряду. Результати подати у вигляді таблиці:

x_i	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:

21	$\sin x$	$[-0.33; 7.4]$
----	----------	----------------

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"<<" |<<" n<<" |<<" " <<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\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:

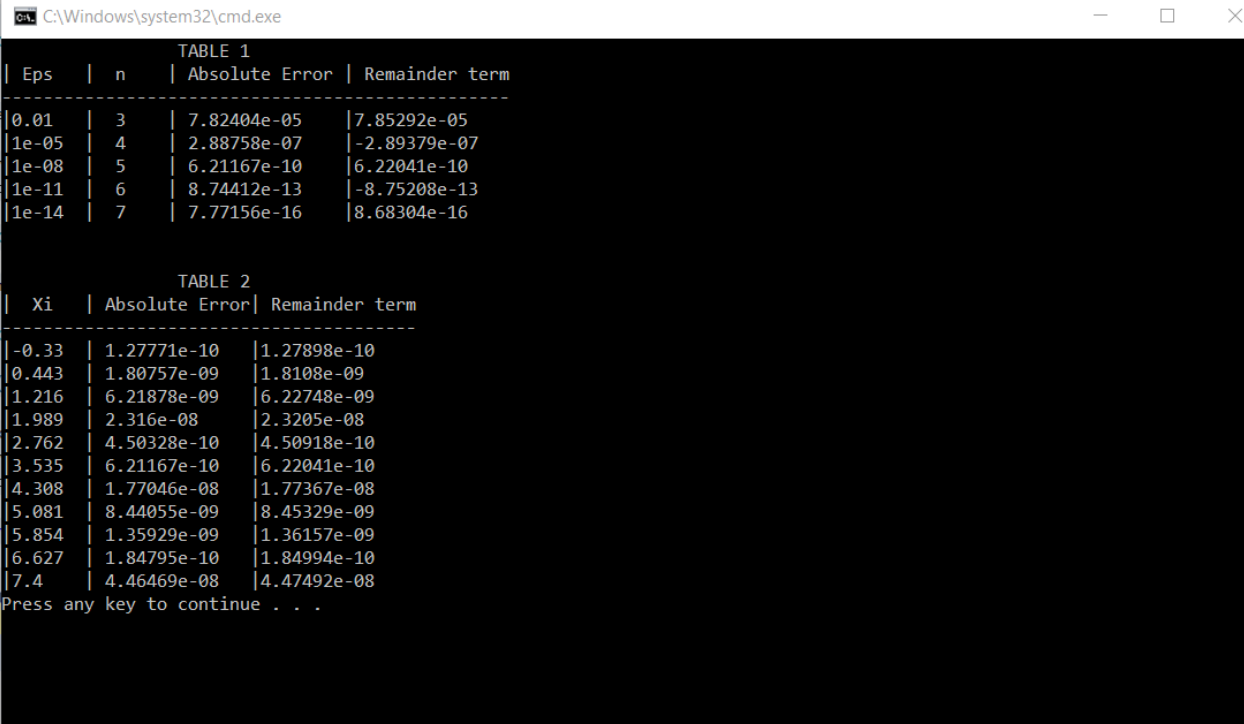


TABLE 1

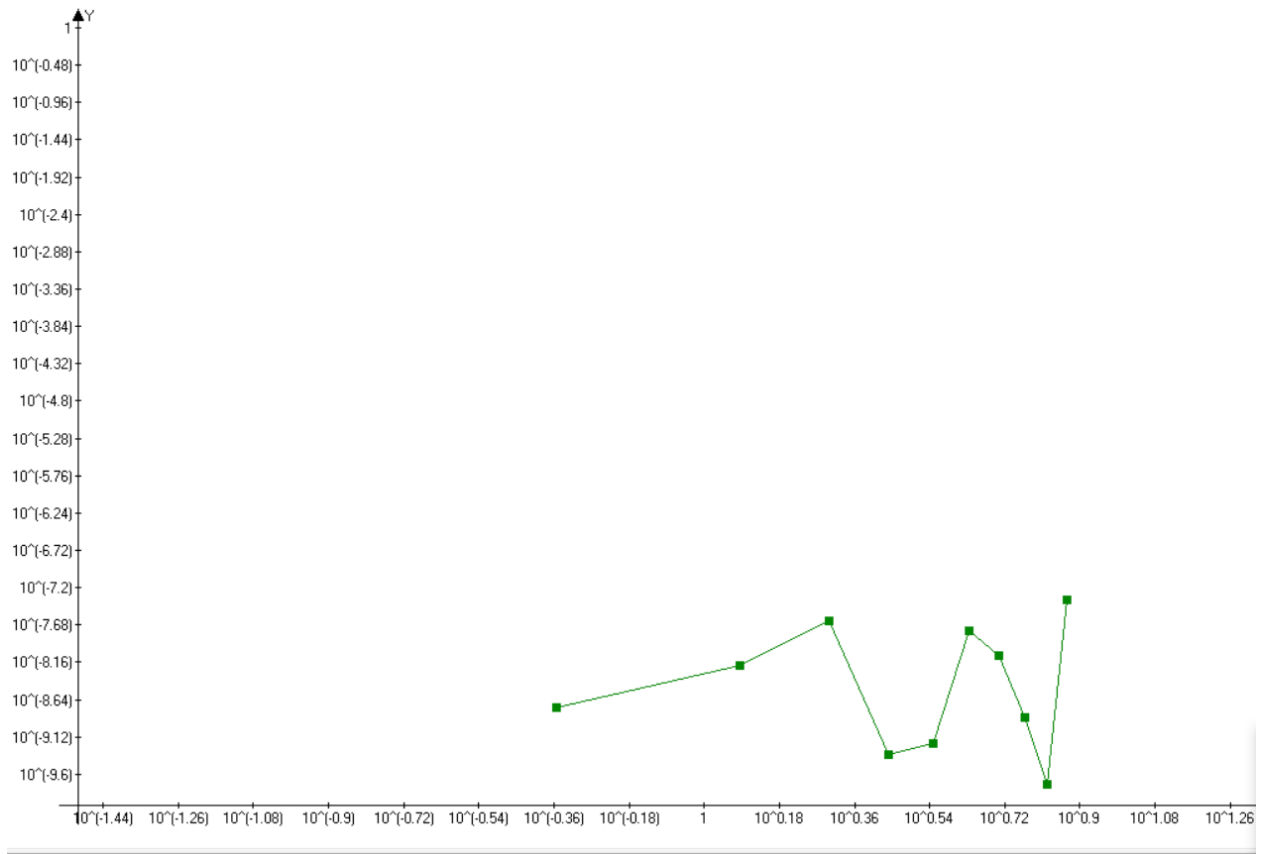
Eps	n	Absolute Error	Remainder term
0.01	3	7.82404e-05	7.85292e-05
1e-05	4	2.88758e-07	-2.89379e-07
1e-08	5	6.21167e-10	6.22041e-10
1e-11	6	8.74412e-13	-8.75208e-13
1e-14	7	7.77156e-16	8.68304e-16

TABLE 2

Xi	Absolute Error	Remainder term
-0.33	1.27771e-10	1.27898e-10
0.443	1.80757e-09	1.8108e-09
1.216	6.21878e-09	6.22748e-09
1.989	2.316e-08	2.3205e-08
2.762	4.50328e-10	4.50918e-10
3.535	6.21167e-10	6.22041e-10
4.308	1.77046e-08	1.77367e-08
5.081	8.44055e-09	8.45329e-09
5.854	1.35929e-09	1.36157e-09
6.627	1.84795e-10	1.84994e-10
7.4	4.46469e-08	4.47492e-08

Press any key to continue . . .

Графік:



Висновки:

В ході виконання лабораторної роботи ми обчислювали наближене значення функції $\sin(x)$ використовуючи розкладання в ряд Маклорена.

Перше завдання стосувалося побудови таблиці залежності довжини ряду, що забезпечує точність функції не меншу за задане значення ϵ_{ps} , від ϵ_{ps} . 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 отриманої залежності є нелінійним та періодичним, що можна пояснити властивостями функції $\sin(x)$.