Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського» Кафедра теоретичних основ радіотехніки

# ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №2

з дисципліни: «Інформатика 1»

Київ – 2021

Виконав: Сокотун Олег

Група: РЕ-11

Викладачі: доцент Катін П.Ю. Оцінка: Підпис:

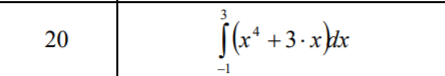
**Мета роботи**: вивчити методи чисельного інтегрування; скласти програму

обчислення визначеного інтегралу чисельними методами; дослідити залежність

точності розрахунку інтегралу від кількості проміжків розбиття інтервалу

інтегрування.

**Інтеграл**:



# Ключові моменти:

# 1)На початку програми я підключаю бібліотеку math.h, і записую 2

# константи, це будуть межі мого інтегралу:

# 2)Для обрахунку кожного з методів я зробив окремі функції:

# double Left\_Triangle(double b,double a,int n)-Метод правого трикутника

# double Right\_Triangle(double b,double a,int n)-Метод лівого трикутника

# double Trapezoid(double b,double a,int n) – Метод трапецій

# double Parabola(double b,double a, int n)– Метод СімпсонаКод

# Код:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define a -1

#define b 3

void Hovel(double C,double Err, int n)

{

printf("C = %lf\n",C);

printf("fault = %lf\n",Err);

printf("Steps = %d\n",n);

}

double Compute(double x)//Integral calculation function

{

return (pow(x,4) + 3 \* x);

}

double Pryamokutnyk(int n)//The function of calculating the integral by the method of rectangles

{

double h,C,x;

h=(b - a) / n;

for(x = a; x <= b ;x = x + h)

{

C = C + Compute(x);

}

C = C \* h;

return C;

}

double Trapecia(int n)//Trapezoidal integral calculation function

{

double h,C = 0,x;

h = (b - a) / n;

C = Compute(b) + Compute(a);

for(x = a; x <= b ;x = x + h)

{

C = C + Compute(x);

}

C = C \* h;

return C;

}

double Simpson(int n)//The function of calculating the integral by the Simpson method

{

double h,C = 0,C1 = 0;

int i = 0;

h = (b - a) / n;

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

{

if(i%2 != 0)

{

C = C + Compute(a+h\*i);

}

if(i%2 == 0)

{

C1 = C1 + Compute(a+h\*i);

}

}

C = ((b-a)/(3\*n))\*(Compute(a) + 4 \* C + 2 \* C1 + Compute(b));

return C;

}

int main()

{

int var, n, i, j=1;

double h,d,x,C=0.0,C1=0.0,L1,L2;

while(1) // infinite cycle

{

printf("Choise method: \n");

printf("1 - Method of rectangles\n2 - Method of trapezoidal\n3 - Simpson method\n4 - Usual method\n5 - Demo version(y = sin(x))\n");

scanf("%d",&var);

switch(var)

{

case 1://Rectangles

printf("Steps: \n");

scanf("%d",&n);

for(j; j<=2;j++)

{

if(j==1)

{

C = Pryamokutnyk(n);

n = n + 2;

}

C1 = Pryamokutnyk(n);

if(fabs(C-C1) > 0.00001 && fabs(C-C1) < 0.001)//whether the error satisfies the answer

{

C1 = fabs(C - C1);

Hovel(C, C1, n);

break;

}

else

{

j = 0;//start from the beginning if the error does not satisfy the answer

}

}

break;

case 2://Trapezoidal

printf("Steps: \n");

scanf("%d",&n);

for(j; j <= 2; j++)

{

if(j==1)

{

C = Trapecia(n);

n = n + 2;

}

C1 = Trapecia(n);

if(fabs(C - C1) > 0.00001 && fabs(C - C1) < 0.001)

{

C1 = fabs(C - C1);

Hovel(C, C1, n);

break;

}

else

{

j=0;

}

}

break;

case 3://Simpson

do

{

printf("This method must have an even number of intervals \n");

printf("Steps: \n");

scanf("%d",&n);

}

while(n%2!=0);

for(j; j<=2; j++)

{

if(j==1)

{

C = Simpson(n);

n = n + 2;

}

C1 = Simpson(n);

if(fabs(C - C1)>0.00001 && fabs(C - C1) < 0.001)

{

C1 = fabs(C - C1);

Hovel(C, C1, n);

break;

}

else

{

j=0;

}

}

break;

case 4://Usual integral

C = (0.5 \* (log (fabs(2 \* a + 1.5))));

C1 = (0.5 \* (log (fabs(2 \* b + 1.5))));

C = C1 - C;

printf("C : %lf\n\n",C);

break;

case 5:

printf("My integral (1->3) y = sin \n");

printf("Steps: \n");

scanf("%d",&n);

printf("Compute answer : 1.53029\n");

//1

h = ((b - a) / n);

for(x = a; x <= b ;x = x + h)

{

C = C + sin(x);

}

C = C \* h;

printf("Method of rectangles : %lf\n, a = %d",C ,a);

//2

h = (b - a) / n;

C = sin(a) + sin(b);

for(x = a; x <= b ;x = x + h)

{

C = C + sin(x);

}

C=C\*h;

printf("Method of trapezoidal : %lf\n",C);

//3

h =((b - a) / n);

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

{

if(i%2 != 0)

{

C = C + (sin(a+h\*i));

}

if(i%2 == 0)

{

C1 = C1 + (sin(a+h\*i));

}

}

C = ((b - a)/(3 \* n))\*(sin(a) + 4 \* C + 2 \* C1 + sin(b));

printf("Simpson method : %lf\n\n",C);

break;

default:

printf("No one variant\n");//If the user hasn't choosen a variant

break;

}

}

return 0;

