Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра теоретичних основ радіотехніки

**ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №2**

з дисципліни: «Інформатика 1»

|  |  |
| --- | --- |
|  | Виконав : Поляков Владислав Сергійович  Група: РЕ-12  Викладачі: доцент Катін П.Ю.  Оцінка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Підпис: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Київ – 2021

**Мета роботи**: вивчити методи чисельного інтегрування; скласти програму обчислення визначеного інтегралу чисельними методами; дослідити залежність точності розрахунку інтегралу від кількості проміжків розбиття інтервалу інтегрування.

**Варіант**: 16

**Обрана функція**: 

**Ключові моменти**:

1. Досліджуємо методи чисельного обчислення інтегралу.
2. Створюємо алгоритм розрахунку.
3. Для кожного методу робимо текстовий інтерфейс (опис роботи).
4. Розрахунки:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод | n=10 | n=100 | n=1000 | N |
| Прямокутники | 0.713247 | 0.707441 | 0.693397 | 50, 104, 1002 |
| Трапеції | 0.726249 | 0.715212 | 0.694894 | 70, 104, 1004 |
| Сімпсон | 0.715874 | 0.703147 | 0.692830 | 46, 102, 1002 |

Код:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

void output(double I, double error, int n ) *//вивід похибки, кількості кроків, значення n*

{

printf("I1 = %lf | ", I);

printf("Error = %lf | ", error);

printf("Amount of steps = %d | \n", n);

}

double Calculate(double x)

{

return (1/(1+x)); *//визначення інтеграла*

}

double Rectangles(int n,double A, double B) *//метод прямокутників*

{

double h, I, x;

h=(B-A)/n;

for(x = A; x <= B; x = x + h)

{

I = I + Calculate(x);

}

I = I \* h;

return I;

}

double Trapezium (int n,double A, double B) *//метод трапецій*

{

double h,I=0,x; *//змінна "і" завжді буде спочатку = 0 при повторному виконанні функції*

h = (B-A) / n;

I = Calculate(A) + Calculate(B);

for(x = A; x <= B ;x = x + h)

{

I = I + Calculate(x);

}

I = I \* h;

return I;

}

double Simpson(int n,double A, double B) *//метод Сімпсона*

{

double h, I = 0, II = 0;

int i=0;

h=((B-A)/n);

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

{

if(i%2 != 0)

{

I=I+Calculate(A+h\*i);

}

if(i%2 == 0)

{

II=II+Calculate(A+h\*i);

}

}

I=((B-A)/(3\*n))\*(Calculate(A)+4\*I+2\*II+Calculate(B));

return I;

}

int main()

{

*//Варіант 16, межі від 0 do 1 F(x) = 1/(1+x)*

int var,n,i,j=1;

double h,d,x,I=0.0,II=0.0,A,B;

while(1) *//infinite while*

{

printf("1 - Rectangles | 2 - Trapezium | 3 - Simpson method | 4 - Standart \n | 5 - User func");

printf("Choose your method: ");

scanf("%d",&var);

switch("%d",var)

{

case 1: *//Rectangles*

printf("Amount of steps: ");

scanf("%d",&n);

printf("lower limit: ");

scanf("%lf",&A);

printf("upper border: ");

scanf("%lf",&B);

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

{

if(j==1)

{

I=Rectangles(n,A,B);

n=n+2;

}

II=Rectangles(n,A,B);

if(fabs(I-II)>0/00001 && fabs(I-II)<0.001)*//розрахунок похибки, fabs - модуль*

{

II=fabs(I-II);

output(I,II,n);

break;

}

else

{

j=0; *//у випадку невиконання умови починає цикл за початку*

}

}

break;

case 2: *//метод трапецій*

printf("Amount of steps: ");

scanf("%d",&n);

printf("lower limit: ");

scanf("%lf",&A);

printf("upper border: ");

scanf("%lf",&B);

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

{

if(j==1)

{

I = Trapezium(n,A,B);

n=n+2;

}

II = Trapezium(n,A,B);

if(fabs(I-II)>0.00001 && fabs(I-II)<0.001)

{

II = fabs(I-II);

output(I,II,n);

break;

}

else

{

j=0;

}

}

break;

case 3: *//метод сімпсона*

do

{

printf("Number of intervals must be even \n");

printf("Amount of steps: ");

scanf("%d",&n);

}while(n%2!=0);

printf("lower limit: ");

scanf("%lf",&A);

printf("upper border: ");

scanf("%lf",&B);

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

{

if(j==1)

{

I = Simpson(n,A,B);

n=n+2;

}

II = Simpson(n,A,B);

if(fabs(I-II)>0.00001 && fabs(I-II)<0.001)

{

II = fabs(I-II);

output(I,II,n);

break;

}

else

{

j=0;

}

}

break;

case 4: *//стандартний спосіб*

printf("lower limit: ");

scanf("%lf",&A);

printf("upper border: ");

scanf("%lf",&B);

I=((pow(A,2)\*pow(A,3))/6)+((2\*sqrt(A))/3);

II=((pow(B,2)\*pow(B,3))/6)+((2\*sqrt(B))/3);

I=II-I;

printf("Integral : %lf\n\n",I);

break;

case 5: *//user version y = x^3*

printf("lower limit: \n");

scanf("%lf",&A);

printf("upper border: \n");

scanf("%lf",&B);

printf("My integral (0->4) y = x^3 \n");

printf("Count of steps: \n");

scanf("%d",&n);

printf("Calc answer : 64\n");

*//1*

h = ((B-A)/n);

h=(B-A)/n;

for(x = A; x <= B ;x = x + h)

{

I=I+pow(x,3);

}

I=I\*h;

printf("Rectangles : %lf\n",I);

*//2*

h=(B-A)/n;

I=pow(A,3)+pow(B,3);

for(x = A; x <= B ;x = x + h)

{

I=I+pow(x,3);

}

I=I\*h;

printf("Trapezium : %lf\n",I);

*//3*

h=((B-A)/n);

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

{

if(i%2 != 0)

{

I=I+pow(A+h\*i,3);

}

if(i%2 == 0)

{

II=II+pow(A+h\*i,3);

}

}

I=((B-A)/(3\*n))\*(pow(A,2)+4\*I+2\*II+pow(B,2));

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

break;

default: *//якщо варіант не було обрано*

printf("Expected to be filled with an option\n");

break;

}

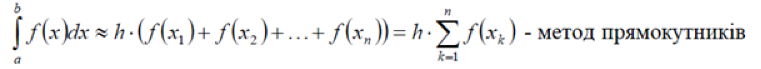
}

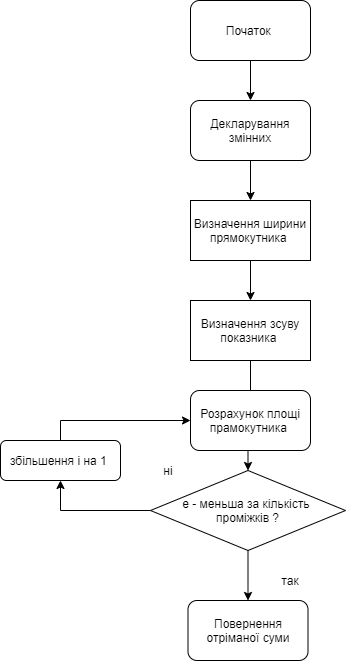
return 0;

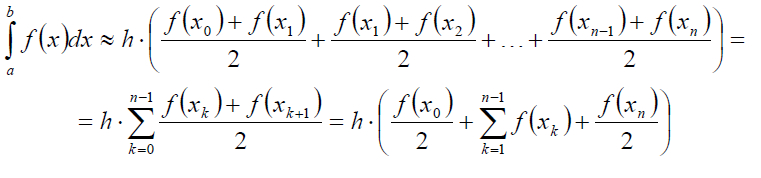
}

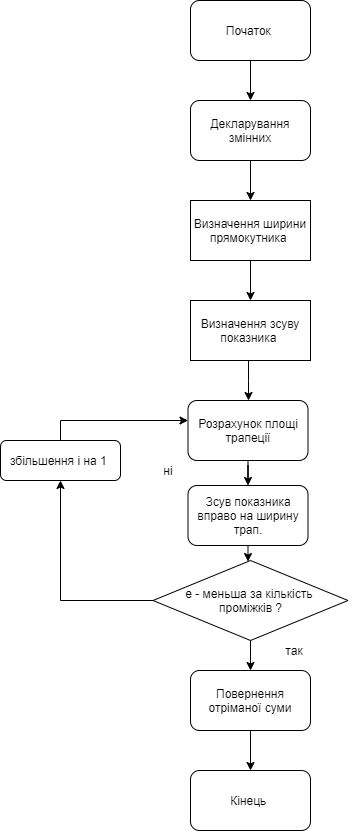
Блок-схеми кожного методу:

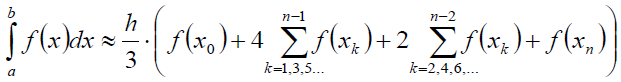
1. Метод прямокутника:

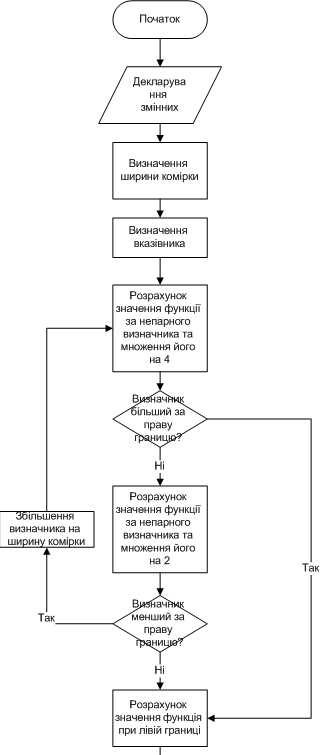


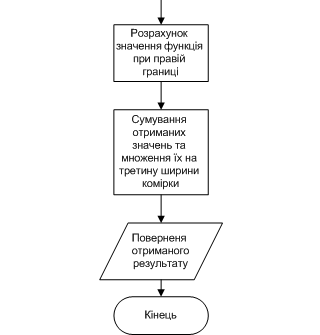


1. Метод трапецій: 

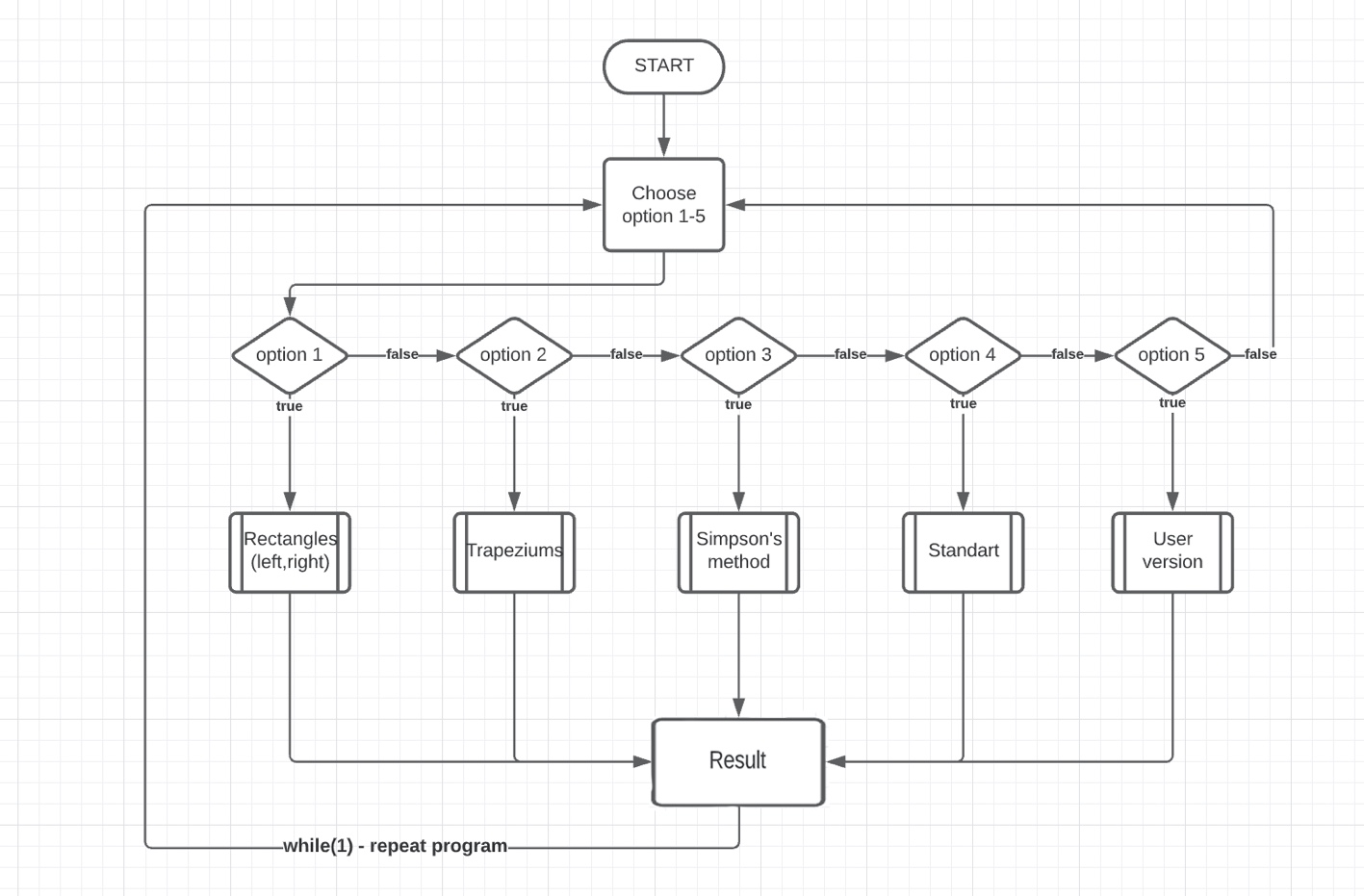


1. Метод Сімпсона (метод парабол): 





Основна блок-схема:



Графік: 