РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ им. ПАТРИСА ЛУМУБЫ

Факультет физико-математических и естественных наук

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

дисциплина: *Вычислительные методы*

Студент:

Мажитов Магомед Асхабович

Группа:

НКНбд-01-21

МОСКВА

2023 г.

Цель:

Написать программу для расчета приближенного значение интеграла.

Теоретическая справка:

1. Реализовать в программе методы левых и правых прямоугольников, метод трапеций и метод Симпсона для приближенного расчета интеграла $\int^b_a f(x)dx$. В программной реализации предусмотреть разбиение отрезка [a ;b], по которому ведется интегрирование, на M отрезков равной длины.
2. Вычислить аналитически значение интеграла $I = \int^b_a f(x)dx$. Сравнить в программе полученное аналитическое значение I с приближенными значениями интеграла $\int^b_a f(x)dx$, вычисленными с помощью методов левых и правых прямоугольников, метода трапеций и метода Симпсона.
3. Вычислить для метода левых прямоугольников минимальное значение целого числа M , при котором погрешность интегрирования меньше, чем 10^{-3} .
4. Вычислить для метода правых прямоугольников минимальное значение целого числа M , при котором погрешность интегрирования меньше, чем 10^{-3} .
5. Вычислить для метода трапеций минимальное значение целого числа M , при котором погрешность интегрирования меньше, чем 10^{-3} .
6. Вычислить для метода Симпсона минимальное значение целого числа M , при котором погрешность интегрирования меньше, чем 10^{-3} .
7. В сравнить полученные в пп.3-6 значения M для каждого из реализованных методов, проанализировать полученные результаты

Листинг:

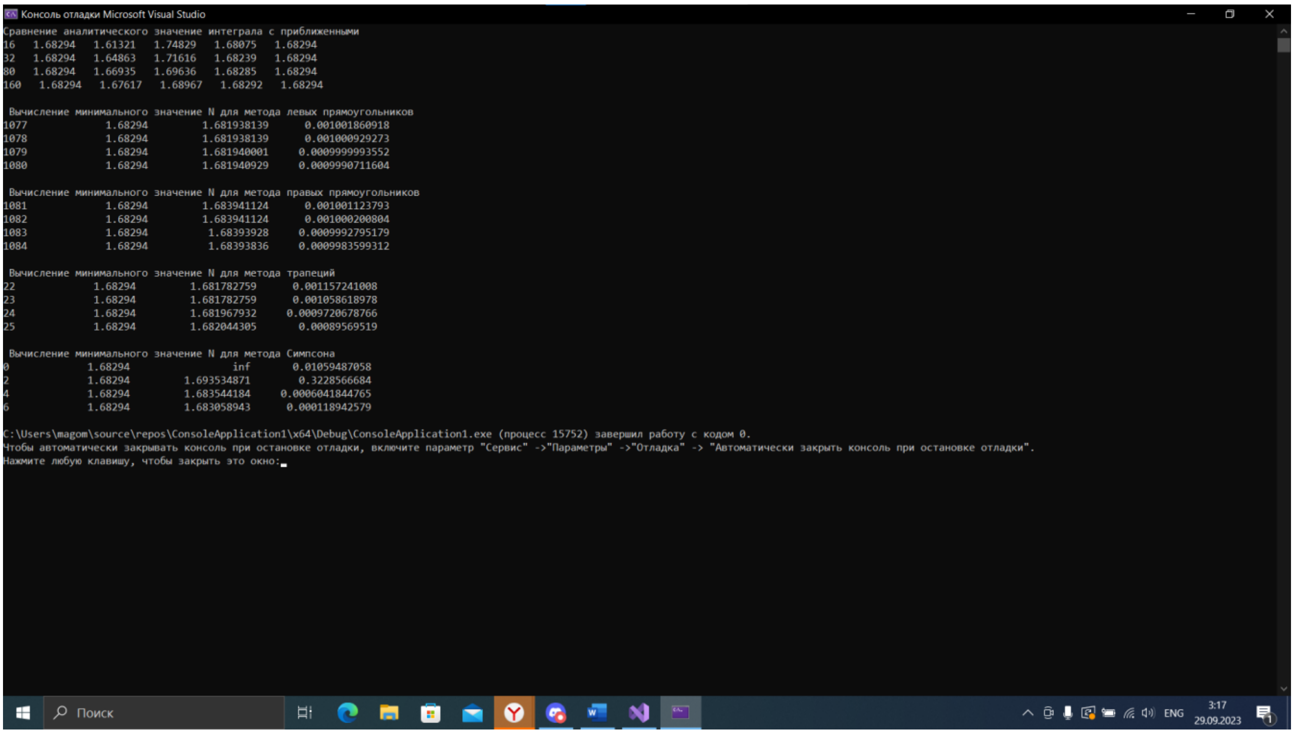
```
#include <iostream>
#include <cmath>
#include <iomanip>

using namespace std;

double f(double x){ //функция подсчета значения функции
    return (x+1)* cos(x);
}

double L_Rectangle(double a, double b, int n){ //метод левых прямоугольников
    double h = (b-a)/n, sum = 0;
    for(int i = 0; i <= n-1; i++){//проходим по циклу от 0 до n-1 и считаем
площади прямоугольников и складываем их
        sum+= h*f(a+i*h);
    }
    return sum; //возвращаем сумму площадей
}

double R_Rectangle(double a, double b, int n){//метод правых прямоугольников
```

```
    double h = (b-a)/n, sum = 0;
    for(int i = 1; i <= n; i++){//проходим по циклу от 1 до n и считаем площади
прямоугольников и складываем их
        sum+= h*f(a+i*h);

    }
    return sum;//возвращаем сумму площадей
}

double Trapezoid(double a, double b, int n){//метод трапеций
    double h = (b-a)/n, sum = f(a) + f(b);//изначально задаем значение sum сумму
функций в начальной и конечной точках
    for(int i = 1; i <= n-1; i++){ //проходим по циклу и считаем значение
функции в каждой точке, и умножаем на 2, и складываем их
        sum+= 2*f(a+i*h);
    }
    sum *= h/2; //умножаем значение sum на шаг и делим на 2
    return sum; //возвращаем полученное значение
}

double Simpson(double a, double b, int n){//метод Симпсона
    double h = (b-a)/n, sum = f(a) + f(b);//изначально задаем значение sum сумму
функций в начальной и конечной точках
    for(int i = 1; i <= n-1; i++){//проходим по циклу и считаем значение функции
в каждой точке, и умножаем на 2 если i четное или на 4 если i нечетное, и
складываем
        if(i%2 == 1){
            sum+= 4*f(a+i*h);
        }
        else{
            sum+= 2*f(a+i*h);
        }
    }
    sum *= h/3; //умножаем значение sum на шаг и делим на 3
    return sum; //возвращаем полученное значение
}

void print(int n, double a, double b, double i){//функция для сравнения
аналитического значение интеграла с приближенными
    cout << setprecision(6) << n << setw(10);
    cout << i << setw(10);
    cout << L_Rectangle(a,b,n) << setw(10);
    cout << R_Rectangle(a,b,n) << setw(10);
    cout << Trapezoid(a,b,n) << setw(10);
    cout << Simpson(a,b,n) << setprecision(10) << endl;
}

void Error_L(double a, double b, double i){ //находим минимальное значение n при
котором погрешность будет меньше 0.001
    bool t = true;
    int n = 1;
    do{ //проходим по циклу пока погрешность не будет меньше 0.001
        if(i - L_Rectangle(a,b,n) >= 0.001){
            n+=1;
        }
        else t = false;
    } while (t);
    cout << " Вычисление минимального значение N для метода левых
прямоугольников" << endl;
    cout << (n - 2) << setw(20) << i << setw(20) << L_Rectangle(a,b,n-2) <<
setw(20) << i - L_Rectangle(a,b,n-2) << endl;
    cout << (n - 1) << setw(20) << i << setw(20) << L_Rectangle(a,b,n-2) <<
setw(20) << i - L_Rectangle(a,b,n-1) << endl;
    cout << n << setw(20) << i << setw(20) << L_Rectangle(a,b,n) << setw(20) <<
```

```
i - L_Rectangle(a,b,n) << endl;
    cout << (n + 1) << setw(20) << i << setw(20) << L_Rectangle(a,b,n+1) <<
setw(20) << i - L_Rectangle(a,b,n+1) << endl;
}

void Error_R(double a, double b, double i){//находим минимальное значение n при
котором погрешность будет меньше 0.001
    bool t = true;
    int n = 1;
    do{//проходим по циклу пока погрешность не будет меньше 0.001
        if(abs(i - R_Rectangle(a,b,n)) > 0.001){
            n+=1;
        }
        else t = false;
    } while (t);
    cout << " Вычисление минимального значение N для метода правых
прямоугольников" << endl;
    cout << (n - 2) << setw(20) << i << setw(20) << R_Rectangle(a,b,n-2) <<
setw(20) << abs(i - R_Rectangle(a,b,n-2)) << endl;
    cout << (n - 1) << setw(20) << i << setw(20) << R_Rectangle(a,b,n-2) <<
setw(20) << abs(i - R_Rectangle(a,b,n-1)) << endl;
    cout << n << setw(20) << i << setw(20) << R_Rectangle(a,b,n) << setw(20) <<
abs(i - R_Rectangle(a,b,n)) << endl;
    cout << (n + 1) << setw(20) << i << setw(20) << R_Rectangle(a,b,n+1) <<
setw(20) << abs(i - R_Rectangle(a,b,n+1)) << endl;
}

void Error_T(double a, double b, double i){//находим минимальное значение n при
котором погрешность будет меньше 0.001
    bool t = true;
    int n = 1;
    do{//проходим по циклу пока погрешность не будет меньше 0.001
        if(abs(i - Trapezoid(a,b,n)) > 0.001){
            n+=1;
        }
        else t = false;
    } while (t);
    cout << " Вычисление минимального значение N для метода трапеций" << endl;
    cout << (n - 2) << setw(20) << i << setw(20) << Trapezoid(a,b,n-2) <<
setw(20) << abs(i - Trapezoid(a,b,n-2)) << endl;
    cout << (n - 1) << setw(20) << i << setw(20) << Trapezoid(a,b,n-2) <<
setw(20) << abs(i - Trapezoid(a,b,n-1)) << endl;
    cout << n << setw(20) << i << setw(20) << Trapezoid(a,b,n) << setw(20) <<
abs(i - Trapezoid(a,b,n)) << endl;
    cout << (n + 1) << setw(20) << i << setw(20) << Trapezoid(a,b,n+1) <<
setw(20) << abs(i - Trapezoid(a,b,n+1)) << endl;
}

void Error_S(double a, double b, double i){//находим минимальное значение n при
котором погрешность будет меньше 0.001
    bool t = true;
    int n = 2;
    do{//проходим по циклу пока погрешность не будет меньше 0.001
        if(abs(i - Simpson(a,b,n)) > 0.001){
            n+=2;
        }
        else t = false;
    } while (t);
    cout << " Вычисление минимального значение N для метода Симпсона" << endl;
    cout << (n - 4) << setw(20) << i << setw(20) << Simpson(a,b,n-4) << setw(20)
<< abs(i - Simpson(a,b,n-2)) << endl;
    cout << (n - 2) << setw(20) << i << setw(20) << Simpson(a,b,n-2) << setw(20)
<< abs(i - Simpson(a,b,n-1)) << endl;
    cout << n << setw(20) << i << setw(20) << Simpson(a,b,n) << setw(20) <<
```

```
abs(i - Simpson(a,b,n)) << endl;
    cout << (n + 2) << setw(20) << i << setw(20) << Simpson(a,b,n+2) << setw(20)
<< abs(i - Simpson(a,b,n+2)) << endl;
}

int main() {
    setlocale(LC_ALL, "Russian");
    int n;
    double a, b, integral;
    a = -1;
    b = 1;
    n = 16;
    integral = 1.68294;
    cout << "Сравнение аналитического значение интеграла с приближенными" <<
endl;
    print(n, a, b, integral);
    print(2*n, a, b, integral);
    print(5*n, a, b, integral);
    print(10*n, a, b, integral);
    cout << endl;
    Error_L(a, b, integral);
    cout << endl;
    Error_R(a, b, integral);
    cout << endl;
    Error_T(a, b, integral);
    cout << endl;
    Error_S(a, b, integral);
    return 0;
}
```

Работа программы:



Вывод:

В ходе работы я реализовал на языке C++ 4 метода нахождение приближенного значения интеграла. Из пунктов 3-6 видно, что самым эффективным методом нахождение приближенного значения интеграла — это метод Симпсона, чуть менее эффективным является метод трапеций, и самым не эффективным методы левых и правых прямоугольников.