**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Основы информатики»

Отчет по лабораторной работе №6

«Численное интегрирование функции»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-11 |  | преподаватель каф. ИУ5 |
| Михалёв Ярослав |  | Козлов А.Д. |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2021 г.

Постановка задачи

1. Численное интегрирование функции с заданной точностью методом прямоугольников.

Вычислить определённый интеграл в пределах от ***a*** до ***b*** для четырех функций f1 = x, f2 = sin( 22 \* x ), f3 = x4 и f4 = arctg(x).

Вычисление интеграла оформить в виде функции IntRect.

Вычисления выполнить для пяти значений точности: 0.01, 0.001, 0.0001, 0.00001 и 0.000001.

Исследовать быстродействие алгоритма в зависимости от подынтегральной функции и требуемой точности (быстродействие алгоритма можно оценить числом элементарных прямоугольников ***n***).

Результаты представить в виде 5 таблиц, по одной таблице для каждого значения точности. В каждой таблице выводить данные для всех четырех функций.

2. Выполнить п.1, используя для интегрирования методом трапеций. Вычисление интеграла оформить в виде функции IntTrap.

Для печати таблиц результатов использовать ту же функцию, что и в методе прямоугольников.

Разработка алгоритма

Входные данные

* double a – левая граница интегрирования
* double b – правая граница интегрирования
* double epss[5] – пять значений точности
* const char\* functions[4] – четыре функции
* double Int[4] – четыре формулы для вычисления точного значения интеграла
* double IntSum[4] – четыре функции для вычисления приближённого значения интеграла
* int N[4] – четыре переменные, хранящие в себе количество итераций
* I\_print printData[4]

IntRect

Вычисляет определённый интеграл методом прямоугольников

Входные данные

* TPF f – функция
* double a – левая граница интегрирования
* double b – правая граница интегрирования
* int n – количество разбиений

Выходные данные

double sum – приближённое значение интеграла

IntTrap

Вычисляет определённый интеграл методом трапеций

Входные данные

* TPF f – функция
* double a – левая граница интегрирования
* double b – правая граница интегрирования
* int n – количество разбиений

Выходные данные

* double sum – приближённое значение интеграла

Integrate

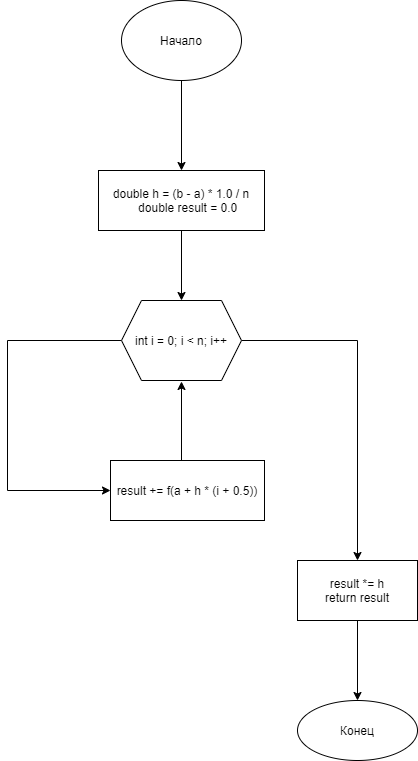
Вычисляет определённый интеграл с переданной точностью

Входные данные

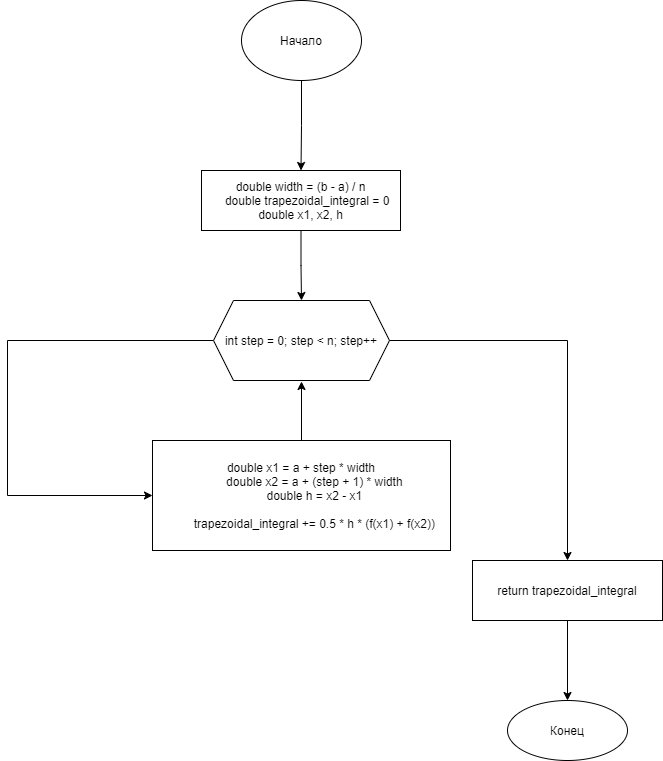
* TPF f – функция
* TPM method – метод прямоугольников / трапеций
* double a – левая граница интегрирования
* double b – правая граница интегрирования
* double eps – требуемая точность
* int &n – количество разбиений

Схема алгоритма

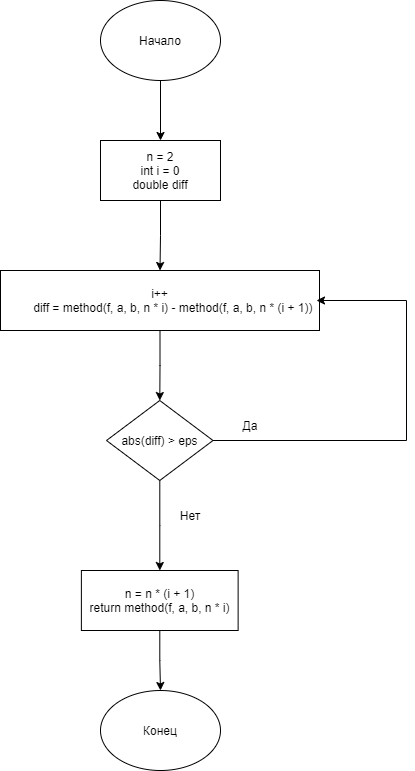
IntRect



IntTrap



Integrate



Текст программы

Main.cpp

#include <iostream>

#include <iomanip>

using namespace std;

typedef double (\*TPF)(double);

typedef double (\*TPM)(TPF f, double a, double b, int n);

struct I\_print { //данные для печати результатов интегрирования

const char\* name; //название функции

double i\_sum; //значение интегральной суммы

double i\_toch; //точное значение интеграла

int n; //число разбиений области интегрирования, при котором достигнута требуемая точность

};

int PrintTabl(I\_print i\_prn[], int k)

{

const int m = 4; //число столбцов таблицы

int wn[m] = { 12,18,18,10 }; //ширина столбцов таблицы

const char\* title[m] = { "Function","Integral","IntSum","N " };

int size[m];

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

size[i] = strlen(title[i]);

//шапка таблицы

cout << char(218) << setfill(char(196));

for (int j = 0; j < m - 1; j++)

cout << setw(wn[j]) << char(194);

cout << setw(wn[m - 1]) << char(191) << endl;

cout << char(179);

for (int j = 0; j < m; j++)

cout << setw((wn[j] - size[j]) / 2) << setfill(' ') << ' ' << title[j]

<< setw((wn[j] - size[j]) / 2) << char(179);

cout << endl;

//заполнение таблицы

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

{

cout << char(195) << fixed;

for (int j = 0; j < m - 1; j++)

cout << setfill(char(196)) << setw(wn[j]) << char(197);

cout << setw(wn[m - 1]) << char(180) << setfill(' ') << endl;

cout << char(179) << setw((wn[0] - strlen(i\_prn[i].name)) / 2) << ' ' << i\_prn[i].name

<< setw((wn[0] - strlen(i\_prn[i].name)) / 2) << char(179);

cout << setw(wn[1] - 1) << setprecision(10) << i\_prn[i].i\_toch << char(179)

<< setw(wn[2] - 1) << i\_prn[i].i\_sum << setprecision(6) << char(179)

<< setw(wn[3] - 1) << i\_prn[i].n << char(179) << endl;

}

//низ таблицы

cout << char(192) << setfill(char(196));

for (int j = 0; j < m - 1; j++)

cout << setw(wn[j]) << char(193);

cout << setw(wn[m - 1]) << char(217) << setfill(' ') << endl;

return 0;

}

void h1(const char\* s)

{

cout << setw(38) << s << "\n\n\n";

}

double f1(double x)

{

return x;

}

double f2(double x)

{

return sin(22 \* x);

}

double f3(double x)

{

return pow(x, 4);

}

double f4(double x)

{

return atan(x);

}

double IntRect(TPF f, double a, double b, int n)

{

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

double sum = 0.0;

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

sum += h \* f(a + i \* h);

return sum;

}

double IntTrap(TPF f, double a, double b, int n)

{

double width = (b - a) \* 1.0 / n;

double trapezoidal\_integral = 0;

double x1, x2, h;

for (int step = 0; step < n; step++) {

x1 = a + step \* width;

x2 = a + (step + 1) \* width;

h = x2 - x1;

trapezoidal\_integral += 0.5 \* h \* (f(x1) + f(x2));

}

return trapezoidal\_integral;

}

double Integrate(TPF f, TPM method, double a, double b, double eps, int &n)

{

n = 10;

int i = 0;

double diff;

do {

i++;

diff = method(f, a, b, n \* i) - method(f, a, b, n \* (i + 1));

} while (abs(diff) > eps);

n = n \* (i + 1);

return method(f, a, b, n \* i);

}

void Do(TPM method)

{

double a = -1;

double b = 3;

double epss[5] = { 0.01, 0.001, 0.0001, 0.00001, 0.000001 };

for (int epsIndex = 0; epsIndex < 5; epsIndex++)

{

double eps = epss[epsIndex];

const char\* functions[4] = {

"y=x",

"y=sin(22\*x)",

"y=x^4",

"y=arctg(x)"

};

double Int[4] = {

(b \* b - a \* a) / 2.0,

(cos(a \* 22.0) - cos(b \* 22.0)) / 22.0,

(b \* b \* b \* b \* b - a \* a \* a \* a \* a) / 5.0,

(b \* atan(b) - a \* atan(a) - (log(b \* b + 1) - log(a \* a + 1)) / 2.0)

};

int n0, n1, n2, n3;

double IntSum[4] = {

Integrate(f1, method, a, b, eps, n0),

Integrate(f2, method, a, b, eps, n1),

Integrate(f3, method, a, b, eps, n2),

Integrate(f4, method, a, b, eps, n3)

};

int N[4] = {

n0,

n1,

n2,

n3

};

I\_print printData[4];

for (int i = 0; i < 4; i++) {

printData[i] = { functions[i], Int[i], IntSum[i], N[i] };

}

cout << setw(34) << "Precision: " << eps << "\n";

PrintTabl(printData, 4);

cout << "\n\n\n";

}

}

int main()

{

h1("Rectangle method");

Do(IntRect);

h1("Trapeze method");

Do(IntTrap);

return 0;

}

Анализ результатов

Метод прямоугольников

Точность: 0.01

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Интеграл | Точное значение | N |
| y=x | 49.500000 | 49.500000 | 20 |
| y=sin(22\*x) | -0.091003 | -0.090729 | 100 |
| y=x^4 | 20000.199857 | 20000.200000 | 350 |
| y=arctg(x) | 11.965578 | 11.964892 | 30 |

Точность: 0.001

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Интеграл | Точное значение | N |
| y=x | 49.500000 | 49.500000 | 20 |
| y=sin(22\*x) | -0.090753 | -0.090729 | 180 |
| y=x^4 | 20000.199993 | 20000.200000 | 750 |
| y=arctg(x) | 11.964954 | 11.964892 | 50 |

Точность: 0.0001

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Интеграл | Точное значение | N |
| y=x | 49.500000 | 49.500000 | 20 |
| y=sin(22\*x) | -0.090731 | -0.090729 | 370 |
| y=x^4 | 20000.200000 | 20000.200000 | 1600 |
| y=arctg(x) | 11.964897 | 11.964892 | 90 |

Точность: 0.00001

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Интеграл | Точное значение | N |
| y=x | 49.500000 | 49.500000 | 20 |
| y=sin(22\*x) | -0.090729 | -0.090729 | 770 |
| y=x^4 | 20000.200000 | 20000.200000 | 3440 |
| y=arctg(x) | 11.964892 | 11.964892 | 180 |

Точность: 0.000001

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Интеграл | Точное значение | N |
| y=x | 49.500000 | 49.500000 | 20 |
| y=sin(22\*x) | -0.090729 | -0.090729 | 1650 |
| y=x^4 | 20000.200000 | 20000.200000 | 7400 |
| y=arctg(x) | 11.964892 | 11.964892 | 380 |

Метод трапеций

Точность: 0.01

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Интеграл | Точное значение | N |
| y=x | 49.500000 | 49.500000 | 20 |
| y=sin(22\*x) | -0.090363 | -0.090729 | 110 |
| y=x^4 | 20000.200113 | 20000.200000 | 440 |
| y=arctg(x) | 11.963520 | 11.964892 | 30 |

Точность: 0.001

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Интеграл | Точное значение | N |
| y=x | 49.500000 | 49.500000 | 20 |
| y=sin(22\*x) | -0.090709 | -0.090729 | 220 |
| y=x^4 | 20000.200005 | 20000.200000 | 940 |
| y=arctg(x) | 11.964837 | 11.964892 | 60 |

Точность: 0.0001

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Интеграл | Точное значение | N |
| y=x | 49.500000 | 49.500000 | 20 |
| y=sin(22\*x) | -0.090728 | -0.090729 | 460 |
| y=x^4 | 20000.200000 | 20000.200000 | 2020 |
| y=arctg(x) | 11.964888 | 11.964892 | 110 |

Точность: 0.00001

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Интеграл | Точное значение | N |
| y=x | 49.500000 | 49.500000 | 20 |
| y=sin(22\*x) | -0.090729 | -0.090729 | 970 |
| y=x^4 | 20000.200000 | 20000.200000 | 4330 |
| y=arctg(x) | 11.964892 | 11.964892 | 220 |

Точность: 0.000001

|  |  |  |  |
| --- | --- | --- | --- |
| Функция | Интеграл | Точное значение | N |
| y=x | 49.500000 | 49.500000 | 20 |
| y=sin(22\*x) | -0.090729 | -0.090729 | 2080 |
| y=x^4 | 20000.200000 | 20000.200000 | 9320 |
| y=arctg(x) | 11.964892 | 11.964892 | 470 |

Вывод

Проанализировав результаты программы, можно сказать, что у метода трапеций лучшая точность по сравнению с методом прямоугольников

Я научился использовать typedef и struct