**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 ИУ5-13Б |  | преподаватель каф. ИУ5 |
| Поляков Данила |  | Аксёнова М.В. |
| 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.

Для печати таблицы результатов использовать функцию

void PrintTabl(I\_print i\_prn[],int k), приведенную в приложении 2.

Здесь i\_prn[] – массив структур типа I\_print размерностью k.

Вид таблицы приведен в Приложении 1.

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

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

**Разработка алгоритма**

**Описание алгоритма**

Для вычисления значение интегральной суммы используется два метода – метод прямоугольников и метод трапеций. Оба метода нацелены на то, чтобы разделить пространство под графиком и представить его в виде соответствующих названиям методов фигур. Чем больше таких фигур получается, тем точнее точность суммы их площадей.

**Описание используемых переменных**

Функции рассчёта интеграла соответствующим способом:

double IntRect(TPF, double, int&);

double IntTrap(TPF, double, int&)

**Обе требуемые функции(IntRect и IntTrap) возвращают значение рассчитанного определённого интеграла. Используемые для этого переменные:**

TPF funq – указатель на функцию типа double (double), для которой производится расчёт

TPFEV exFunq – указатель на функцию типа double. Используется для расчёта точного значения

double eps – точность рассчитываемого значения

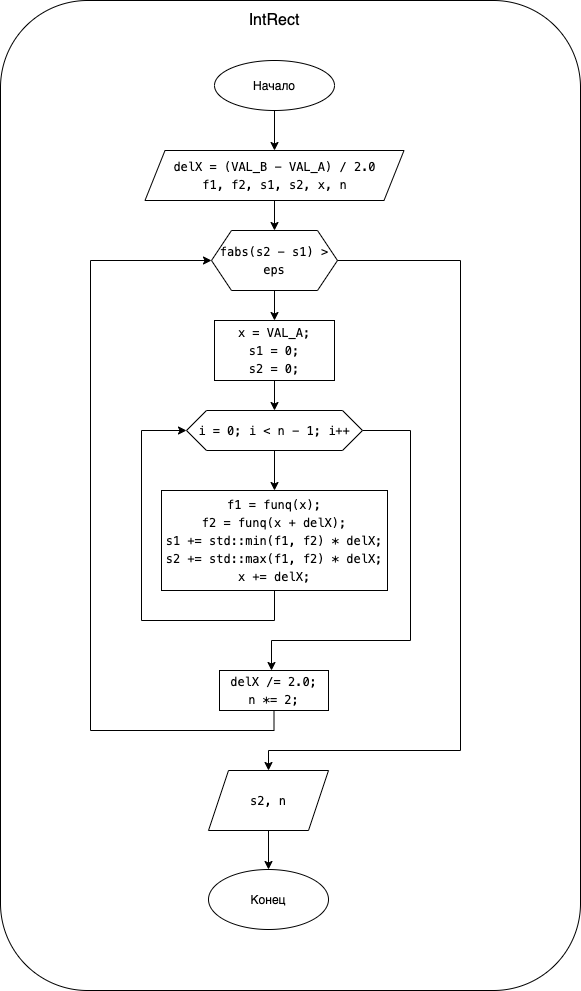
char \*functionName – название интегрируемой функции

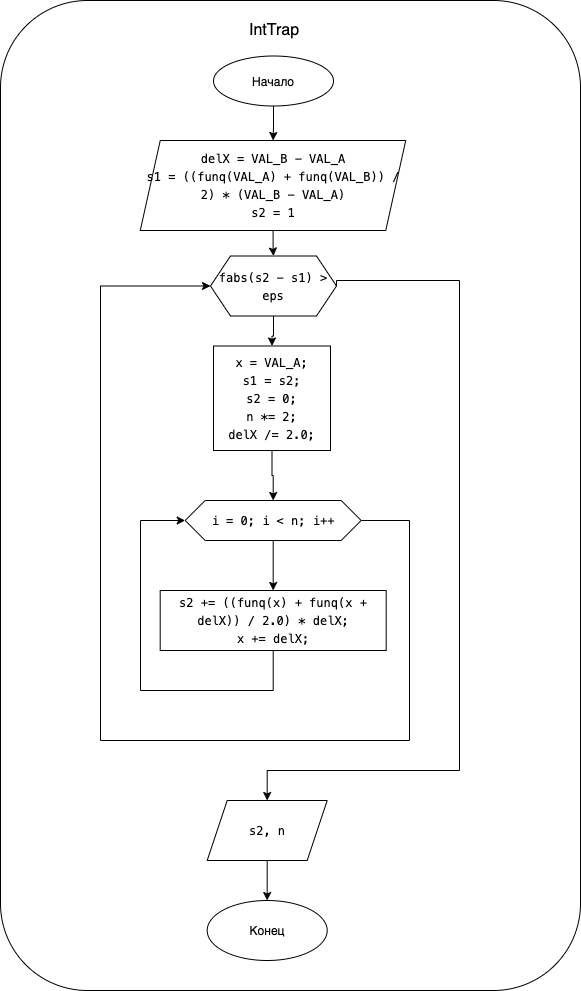
enum ABValues {

VAL\_A = 0,

VAL\_B = 1

}; - константы границ расчёта интеграла

**Схема алгоритма**

****

**Текст программы**

**Листинг кода программы:**

**functions.h**

#include <cmath>

#include <iostream>

#include <iomanip>

#include <cstring>

#define f1Name "y = x "

#define f2Name "y = sin(22x)"

#define f3Name "y = x^4 "

#define f4Name "y = arctg(x)"

#define textRect "методом прямоугольников"

#define textTrap "методом трапеций"

#define printInfoSize 4

typedef double(\*TPF)(double);

typedef double(\*TPFEV)();

enum ABValues {

VAL\_A = 0,

VAL\_B = 1

};

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

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

};

void printInfoBeforeTable(const char\*, double);

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

double IntRect(TPF, double, int&);

double IntTrap(TPF, double, int&);

double f1ExactValue();

double f2ExactValue();

double f3ExactValue();

double f4ExactValue();

double f1(double);

double f2(double);

double f3(double);

double f4(double);

**functions.cpp**

#include "functions.h"

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 f1ExactValue() {

return (VAL\_B \* VAL\_B - VAL\_A \* VAL\_A) / 2.0;

}

double f2ExactValue() {

return (cos(VAL\_A \* 22.0) - cos(VAL\_B \* 22.0)) / 22.0;

}

double f3ExactValue() {

return (VAL\_B \* VAL\_B \* VAL\_B \* VAL\_B \* VAL\_B - VAL\_A \* VAL\_A \* VAL\_A \* VAL\_A \* VAL\_A) / 5.0;

}

double f4ExactValue() {

return VAL\_B \* atan((double)VAL\_B) - VAL\_A \* atan((double)VAL\_A) - (log(VAL\_B \* VAL\_B + 1) - log(VAL\_A \* VAL\_A + 1)) / 2.0;

}

double IntRect(TPF funq, double eps, int& n) {

double delX = (VAL\_B - VAL\_A) / 2.0;

double x;

double f1, f2, s1, s2;

n = 2;

do {

x = VAL\_A;

s1 = 0;

s2 = 0;

for (int i = 0; i < n - 1; i++) {

f1 = funq(x);

f2 = funq(x + delX);

s1 += std::min(f1, f2) \* delX;

s2 += std::max(f1, f2) \* delX;

x += delX;

}

delX /= 2.0;

n \*= 2;

} while (fabs(s2 - s1) > eps);

return s2;

}

double IntTrap(TPF funq, double eps, int& n) {

double delX = (double) VAL\_B - VAL\_A;

double x;

double s1 = ((funq(VAL\_A) + funq(VAL\_B)) / 2) \* (VAL\_B - VAL\_A), s2 = s1;

n = 1;

do {

x = VAL\_A;

s1 = s2;

s2 = 0;

n \*= 2;

delX /= 2.0;

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

s2 += ((funq(x) + funq(x + delX)) / 2.0) \* delX;

x += delX;

}

} while (fabs(s2 - s1) > eps);

n/=2;

return s2;

}

void printInfoBeforeTable(const char \*text, double eps) {

std::cout << "\t\tТаблица рассчёта интеграла" << std::endl;

std::cout << "\t от " << VAL\_A << " до " << VAL\_B << " с точностью до " << std::scientific << eps << ' ' << text << std::endl;

}

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

const int m = 4;

int wn[m] = {16, 18, 18, 10};

char title[m][10] = {"Function", "Integral", "IntSum", "N "};

int size[m];

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

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

std::cout << '\_' << std::setfill('\_');

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

std::cout << std::setw(wn[j]) << '\_';

std::cout << std::setw(wn[m - 1]) << '\_' << std::endl;

std::cout << '|';

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

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

<< std::setw((wn[j] - size[j]) / 2) << '|';

std::cout << std::endl;

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

std::cout << '|' << std::fixed;

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

std::cout << std::setfill('\_') << std::setw(wn[j]) << '|';

std::cout << std::setw(wn[m - 1]) << '|' << std::setfill(' ') << std::endl;

std::cout << '|' << std::setw((wn[0] - strlen(i\_prn[i].name)) / 2) << ' ' << i\_prn[i].name

<< std::setw((wn[0] - strlen(i\_prn[i].name)) / 2) << '|';

std::cout << std::setw(wn[1] - 1) << std::setprecision(10) << i\_prn[i].i\_toch << '|'

<< std::setw(wn[2] - 1) << i\_prn[i].i\_sum << std::setprecision(6) << '|'

<< std::setw(wn[3] - 1) << i\_prn[i].n << '|' << std::endl;

}

std::cout << '|' << std::setfill('\_');

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

std::cout << std::setw(wn[j]) << '|';

std::cout << std::setw(wn[m - 1]) << '|' << std::setfill(' ') << std::endl;

}

**lab6.cpp**

#include "functions.h"

int main() {

I\_print printInfo[printInfoSize];

TPF functions[] = {f1,f2,f3,f4};

TPFEV functionsExactValues [] = {f1ExactValue,f2ExactValue,f3ExactValue,f4ExactValue};

char\* functionsNames[] = {f1Name,f2Name,f3Name,f4Name};

double eps = 0.01;

for (size\_t i = 0; i < printInfoSize; i++)

{

printInfo[i].name = functionsNames[i];

printInfo[i].i\_toch = functionsExactValues[i]();

}

int n;

while (eps > 0.000001) {

for (size\_t i = 0; i < printInfoSize; i++)

{

printInfo[i].i\_sum = IntRect(functions[i], eps, n);

printInfo[i].n = n;

}

eps \*= 0.1;

printInfoBeforeTable(textRect, eps);

PrintTabl(printInfo, printInfoSize);

}

eps = 0.01;

while (eps > 0.000001) {

for (size\_t i = 0; i < printInfoSize; i++)

{

printInfo[i].i\_sum = IntTrap(functions[i], eps, n);

printInfo[i].n = n;

}

eps \*= 0.1;

printInfoBeforeTable(textTrap, eps);

PrintTabl(printInfo, printInfoSize);

}

return 0;

}

**Анализ результатов**

