МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп'ютерні інформаційні технології»

**Звіт**

**з навчальної практики**

Виконав:

студент гр.ПЗ1911

Сафонов Д.Є.

Прийняла:

Шаповал І.В.

Дніпро, 2020

# **Постановка завдання**

Ознайомитись з методами чисельного інтегрування функцій. Провести програмну реалізацію методів у відповідності до обраного рівня складності та оформити звіт.

Програма повинна забезпечувати введення наступних даних: границь проміжку інтегрування функції, кількість відрізків для обчислення інтеграла, вибір підінтегральної функції та метода її інтегрування. Програма обчислює значення інтегралу; значення виводяться з точністю до шести знаків. В доповнення може будуватись графік функції та графічне представлення методу інтегрування.

Перелік методів чисельного інтегрування функцій:

метод лівих прямокутників;

метод правих прямокутників;

метод середніх прямокутників;

метод трапецій;

метод парабол (метод Сімпсона).

Рівні складності завдань визначаються наступним чином:

|  |  |  |
| --- | --- | --- |
| Рівень | Вимоги до оформлення програми | Вимоги до змісту завдання |
| А | В незалежності від типу інтерфейсу обов’язковий контроль вхідної інформації з реалізацією методів і функцій в окремих файлах. Підінтегральна функція повинна передаватись в якості параметра (вказівник на функцію). | Для набору мінімум з трьох різних функцій повинні бути реалізовані всі методи. Графік функції та графічне представлення методу інтегрування (окрім методу Сімпсона). |

# **Зовнішні специфікації**

2.1. Формат вхідних даних:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва | Умовне позначення | Вимоги | Приклад |
| 1 | Вибір методу інтегрування | method | Ціле число від нуля до 4 включно(обирається через listBox) | 3 |
| 2 | Вибір функції для інтегрування | f | Ціле число від нуля до 4 включно(обирається через listBox) | 2 |
| 3 | Лівий ліміт інтегрування | a | Дійсне число менше ніж значення правого ліміту | 0 |
| 4 | Правий ліміт інтегрування | b | Дійсне число більше ніж значення лівого ліміту | 3.141592653 |
| 5 | Кількість фрагментів | N | Ціле число | 50 |

2.2. Формат вихідних даних:

- Графік функції з фігурами фрагментів, координатними осями

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва | Умовне позначення | Вимоги | Приклад |
| 1 | Інтеграл функції | integral | Дійсне число | 2 |

2.3. Функціональні вимоги до програми:

Програма повинна забезпечувати введення наступних даних: границь проміжку інтегрування функції, кількість відрізків для обчислення інтеграла, вибір підінтегральної функції та метода її інтегрування. Програма обчислює значення інтегралу; значення виводяться з точністю до шести знаків. В доповнення може будуватись графік функції та графічне представлення методу інтегрування.

# **Вибір методу рішення задачи**

Були реалізовані усі запропоновані методи наближеного інтегрування без використання першообразних. Методи прямокутників відрізняються тільки тим, що висота дорівнює значенню функції на лівій границі прямокутника, правій, та посередині відповідно. Інтеграл в усіх методах дорівнює сумі площин прямокутників:

; Де m = 0, для лівих прямокутників, =1 для середніх, =2 для правих.

Метод трапецій дуже схожий на групу попередніх методів. Але у цьому методі інтеграл дорівнює сумі площин трапецій:

Метод Сімпсона полягає в апроксимації функції многочленом другого порядку на відрізку:

Для виведення графіку функції потрібно переводити Декартові(далі реальні) координати в піксельні(далі графічні).

Перша різниця це напрямки базисних векторів, в декартовій системі координат горизонтальний вектор(абсциса) направлений вправо, в графічній також вправо. А вертикальні вектори мають різні напрямки ордината в декартовій системі координат – вверх, а в графічній вниз.

Але сама головна різниця – обмеження обох координат в графічних координатах. Покладемо hg – висота в пікселях, wg – ширина в пікселях, a та b – ліва та права границя графіку відповідно. Для виведення графіку функції f(x) знайдемо формули переведення координат з реальної системи до графічної:

; довжина відрізку на якому малюється графік функції у реальних координатах

Для знаходження висоти у реальних координатах потрібно найти локальний мінімум та максимум на відрізку [a; b], будь-яким методом, наприклад – перебору, бісекції, або інш. Покладемо, що мі їх знайшли, позначимо найменше значення – h\_lowest, найбільше – h\_highest.

; висота у реальних координатах

; відношення реальної довжини до графічної

відношення реальної висоти до графічної

Але графічні координати, окрім перерахованого вище, не мають від’ємних координат, тож потрібно не тільки множити на коефіцієнт, а ще й модифікувати значення після цього. Значення ординати потрібно помножити на -1 та відняти від максимального значення(hg), бо графічні координати не мають від’ємних значень.

З цього получимо формули:

; формула приведення реального значення абсциси до графічного

; формула приведення реального значення ординати до графічного

# **Розробка алгоритмів рішення програми**

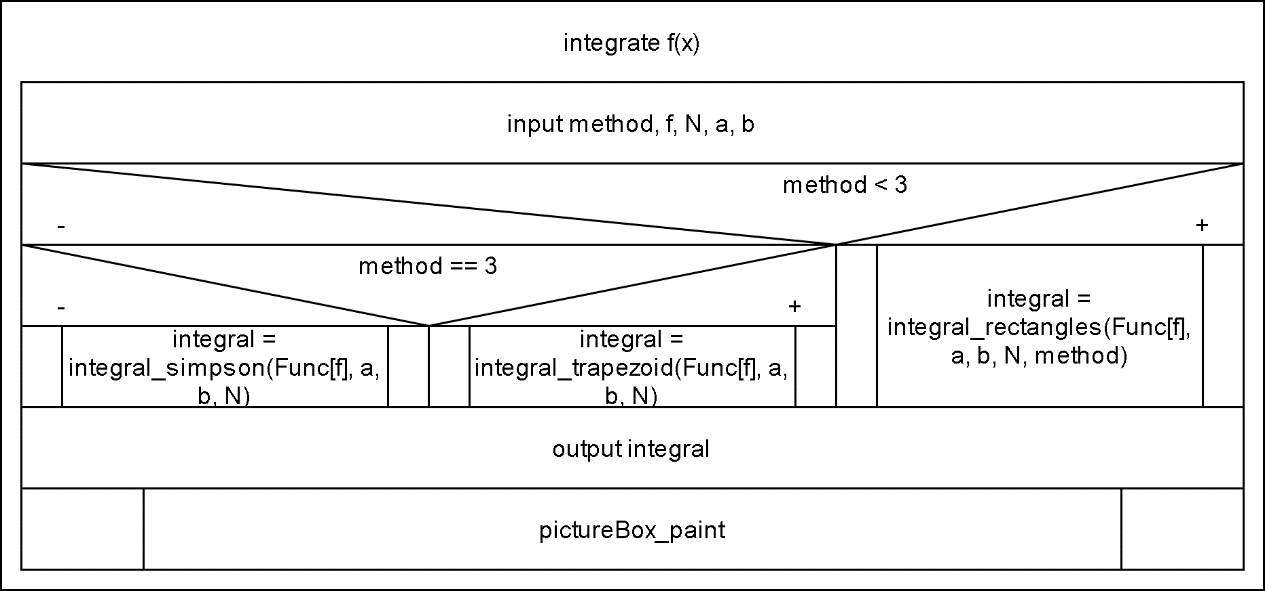


Рисунок 1

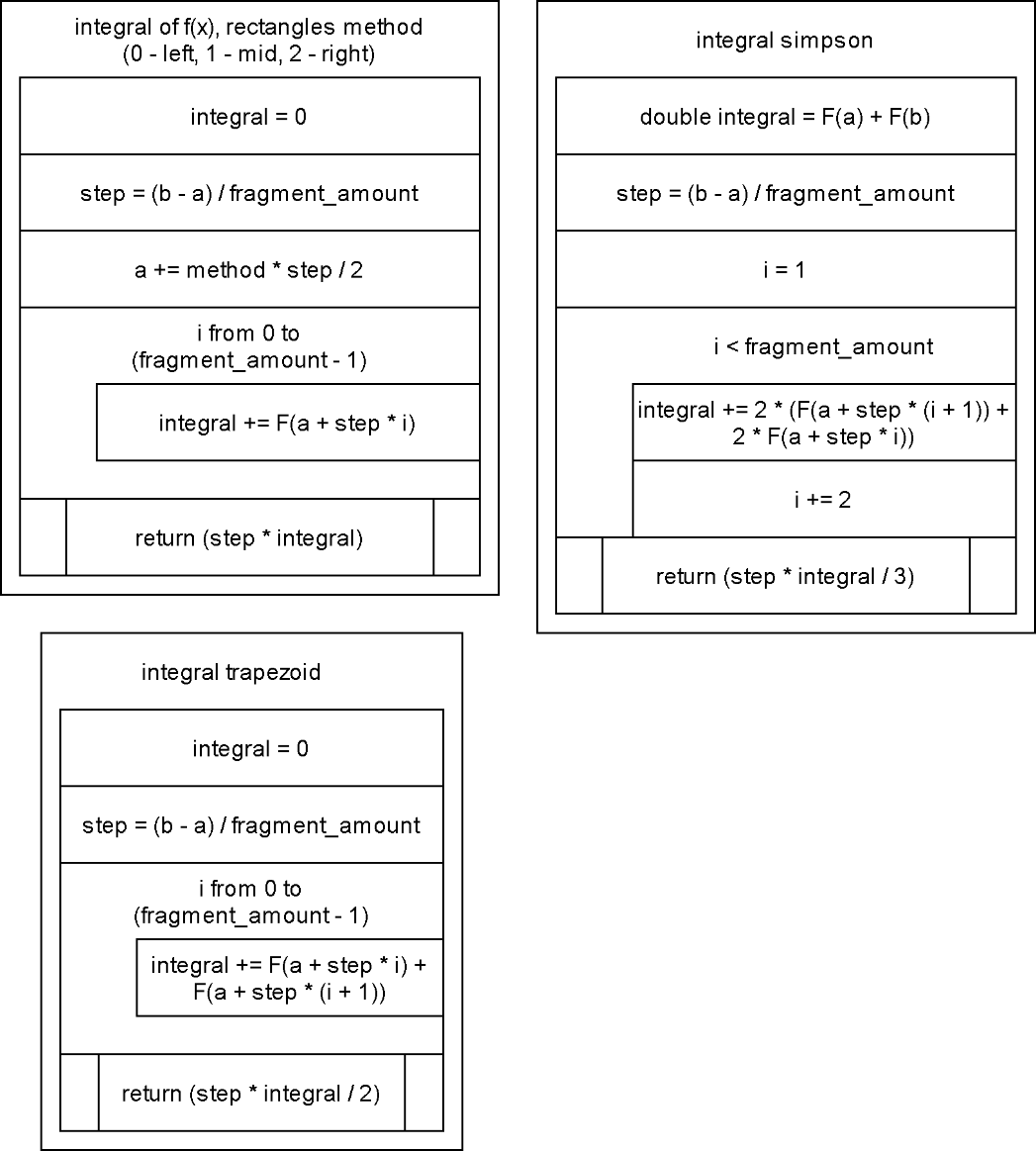
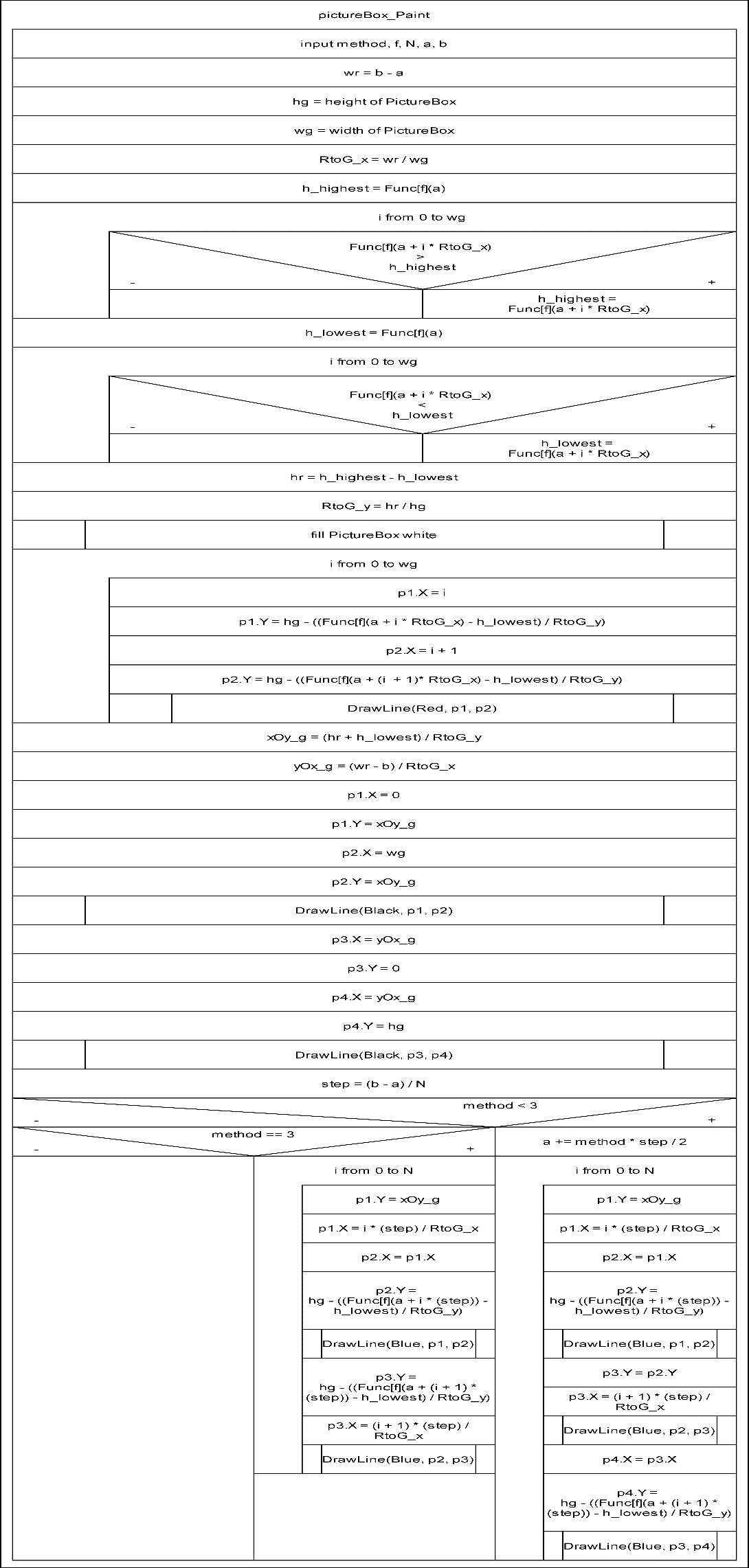


Рисунок 2



# **Текст програми**

“MyForm.h”

#include "integral\_methods.h"

#include <cstdlib>

#include <ctime>

#include <cmath>

private: System::Void button1\_Click(System::Object^ sender, System::EventArgs^ e) {

pointFunc2d Func[5]{f\_x, f\_x0, f\_x1, f\_x2, f\_x3};

double integral;

int method = listBox1->SelectedIndex;

int f = listBox2->SelectedIndex;

int N = Convert::ToInt16(maskedTextBox4->Text);

double a = Convert::ToDouble(maskedTextBox1->Text);

double b = Convert::ToDouble(maskedTextBox2->Text);

if (method < 3)

integral = integral\_rectangles(Func[f], a, b, N, method);

else if(method == 3)

integral = integral\_trapezoid(Func[f], a, b, N);

else

integral = integral\_simpson(Func[f], a, b, N);

textBox1->Text = Convert::ToString(integral);

System::Drawing::Graphics^ graph = this->pictureBox1->CreateGraphics();

System::Drawing::Rectangle rect;

System::Windows::Forms::PaintEventArgs^ e1 = gcnew System::Windows::Forms::PaintEventArgs(graph, rect);

this->pictureBox1\_Paint(sender, e1);

}

private: System::Void pictureBox1\_Paint(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) {

pointFunc2d Func[5]{ f\_x, f\_x0, f\_x1, f\_x2, f\_x3 };

double integral;

int method = listBox1->SelectedIndex;

int f = listBox2->SelectedIndex;

int N = Convert::ToInt16(maskedTextBox4->Text);

double a = Convert::ToDouble(maskedTextBox1->Text);

double b = Convert::ToDouble(maskedTextBox2->Text);

double wr = b - a;//real width

int wg = this->pictureBox1->Width;//graphical width

int hg = this->pictureBox1->Height;//graphical height

double RtoG\_x = wr / wg;//coefs

double h\_highest = Func[f](a);

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

if (Func[f](a + i \* RtoG\_x) > h\_highest)

h\_highest = Func[f](a + i \* RtoG\_x);

double h\_lowest = Func[f](a);

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

if (Func[f](a + i \* RtoG\_x) < h\_lowest)

h\_lowest = Func[f](a + i \* RtoG\_x);

double hr = h\_highest - h\_lowest;//real height

double RtoG\_y = hr / hg;//coefs

Point ltp(0, 0);Point rbp(wg - 1, hg - 1);

e->Graphics->FillRectangle(Brushes::White, ltp.X, ltp.Y, rbp.X - ltp.X, rbp.Y - ltp.Y);

e->Graphics->DrawRectangle(Pens::Black, ltp.X, ltp.Y, rbp.X- ltp.X, rbp.Y - ltp.Y);

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

Point p1(i, hg - ((Func[f](a + i \* RtoG\_x) - h\_lowest) / RtoG\_y));

Point p2(i + 1, hg - ((Func[f](a + (i + 1) \* RtoG\_x) - h\_lowest) / RtoG\_y));

e->Graphics->DrawLine(Pens::Red, p1, p2);

}

int xOy\_g, yOx\_g;

xOy\_g = (hr + h\_lowest) / RtoG\_y;

yOx\_g = (wr - b) / RtoG\_x;

Point p1(0, xOy\_g);

Point p2(wg, xOy\_g);

e->Graphics->DrawLine(Pens::Black, p1, p2);//OY

Point p3(yOx\_g, 0);

Point p4(yOx\_g, hg);

e->Graphics->DrawLine(Pens::Black, p3, p4);//OX

double step = (b - a) / N;

if (method < 3) {

a += method \* step / 2;

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

p1.Y = xOy\_g;

p1.X = i \* (step) / RtoG\_x;

p2.X = p1.X;

p2.Y = hg - ((Func[f](a + i \* (step)) - h\_lowest) / RtoG\_y);

e->Graphics->DrawLine(Pens::Blue, p1, p2);

p3.Y = p2.Y;

p3.X = (i + 1) \* (step) / RtoG\_x;

e->Graphics->DrawLine(Pens::Blue, p2, p3);

p4.X = p3.X;

p4.Y = hg - ((Func[f](a + (i + 1) \* (step)) - h\_lowest) / RtoG\_y);

e->Graphics->DrawLine(Pens::Blue, p3, p4);

}

}

else if (method == 3) {

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

p1.Y = xOy\_g;

p1.X = i \* (step) / RtoG\_x;

p2.X = p1.X;

p2.Y = hg - ((Func[f](a + i \* (step)) - h\_lowest) / RtoG\_y);

e->Graphics->DrawLine(Pens::Blue, p1, p2);

p3.Y = hg - ((Func[f](a + (i + 1) \* (step)) - h\_lowest) / RtoG\_y);

p3.X = (i + 1) \* (step) / RtoG\_x;

e->Graphics->DrawLine(Pens::Blue, p2, p3);

}

}

else {

//simpson's?

}

}

private: System::Void MyForm\_Load(System::Object^ sender, System::EventArgs^ e) {

listBox1->SelectedIndex = 0;

listBox2->SelectedIndex = 0;

}

“integral\_methods.h”

typedef double (\*pointFunc2d)(double);

double f\_x(double x);

double f\_x0(double x);

double f\_x1(double x);

double f\_x2(double x);

double f\_x3(double x);

double integral\_rectangles(pointFunc2d F, double a, double b, double N, int method);

double integral\_trapezoid(pointFunc2d F, double a, double b, double fragment\_amount);

double integral\_simpson(pointFunc2d F, double a, double b, double fragment\_amount);

“integral\_methods.cpp”

#include <cmath>

#include "integral\_methods.h"

#include <iostream>

double f\_x(double x) {

return sin(x);

}

double f\_x0(double x) {

return pow(x, 1 / x);

}

double f\_x1(double x) {

return (-(x \* x) + 3 \* x + 7);

}

double f\_x2(double x) {

return (-(x \* x \* x) + (4 \* x \* x) - 6 \* x + 24);

}

double f\_x3(double x) {

return sin(x);

}

double integral\_rectangles(pointFunc2d F, double a, double b, double fragment\_amount, int method) {//method = 0,1,2 to choose left, middle or right rectangles

double integral = 0, step = (b - a) / fragment\_amount;

a += method \* step / 2;

for (int i = 0; i < fragment\_amount; i++)

integral += F(a + step \* i);

return step \* integral;

}

double integral\_trapezoid(pointFunc2d F, double a, double b, double fragment\_amount) {

double integral = 0, step = (b - a) / fragment\_amount;

for (int i = 0; i < fragment\_amount; i++)

integral += F(a + step \* i) + F(a + step \* (i + 1));

return step \* integral / 2;

}

double integral\_simpson(pointFunc2d F, double a, double b, double fragment\_amount) {//fragment\_amount % 2 = 0

double integral = F(a) + F(b), step = (b - a) / fragment\_amount;

for (int i = 1; i < fragment\_amount; i += 2)

integral += 2 \* (F(a + step \* (i + 1)) + 2 \* F(a + step \* i));

return step \* integral / 3;}

# **Контрольний приклад**

Метод лівих прямокутників:

Метод середніх прямокутників:

Метод правих прямокутників:

Метод трапецій:

Метод Сімпсона:

# **Тестування програми**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Вхідні дані | Очікуваний результат |
| 1 |  | a = 0  b = 3.141592653  N = 30 | Integral = 2 |
| 2 |  | a = 0.00001  b = 5  N = 30 | Integral = 5.89517… |
| 3 |  | a = -3  b = 6  N = 30 | Integral = 22.5 |
| 4 |  | a = -2  b = 5  N = 30 | Integral = 130.08(3) |

# **Аналіз результатів тестування і роботи програми**

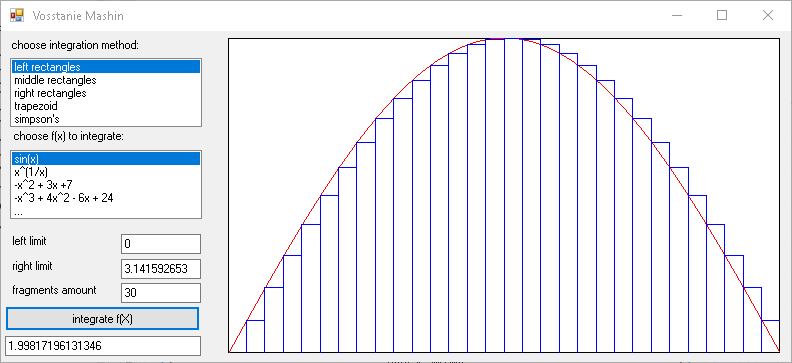


Рисунок 3(Тест1)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить.

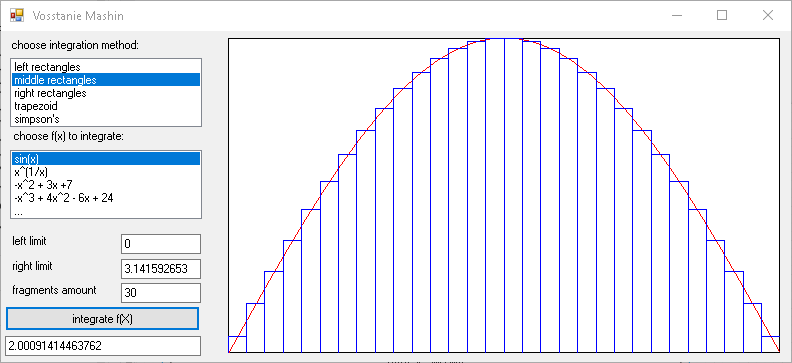


Рисунок 4(Тест1)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить.

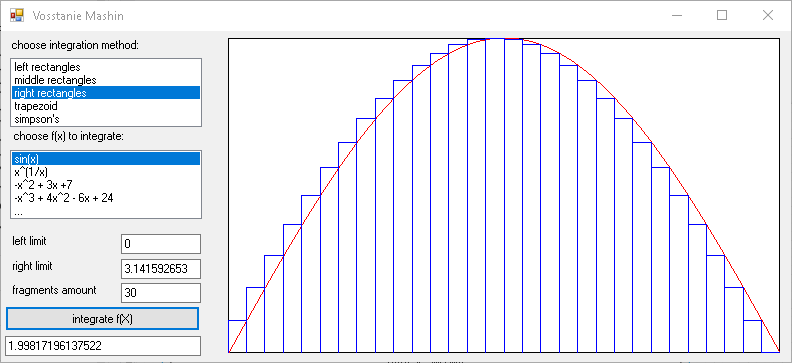


Рисунок 5(Тест1)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить.

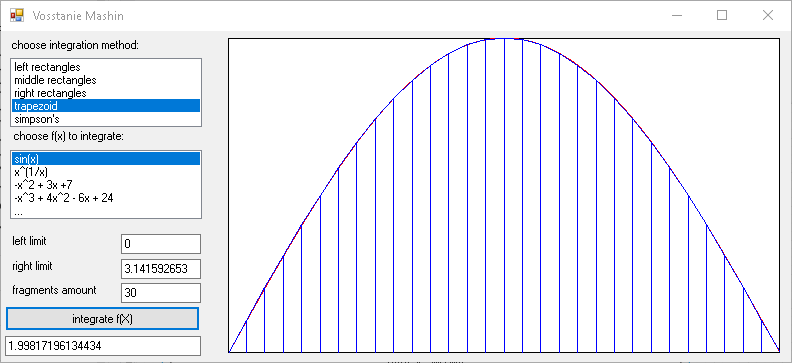


Рисунок 6(Тест1)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить.

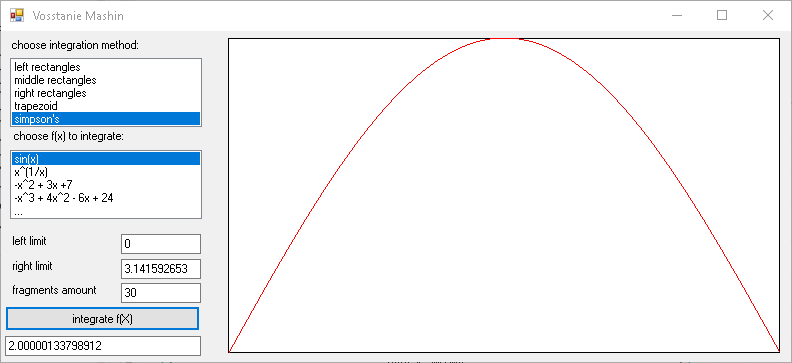


Рисунок 7(Тест1)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить.

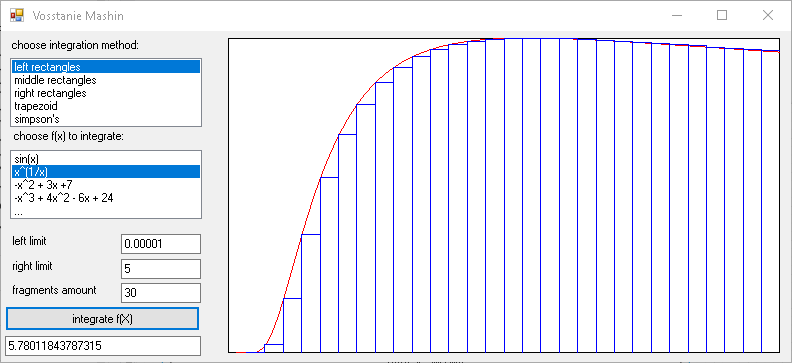


Рисунок 8(Тест2)

Результат відповідає очікуванням, присутня відносно велика похибка, зумовлена різким зростанням функції.

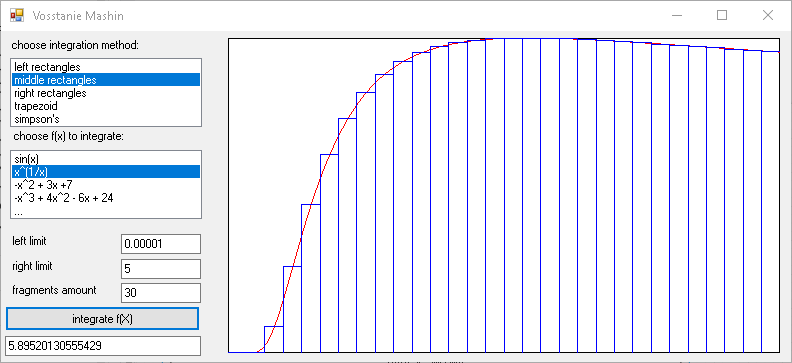


Рисунок 9(Тест2)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить.

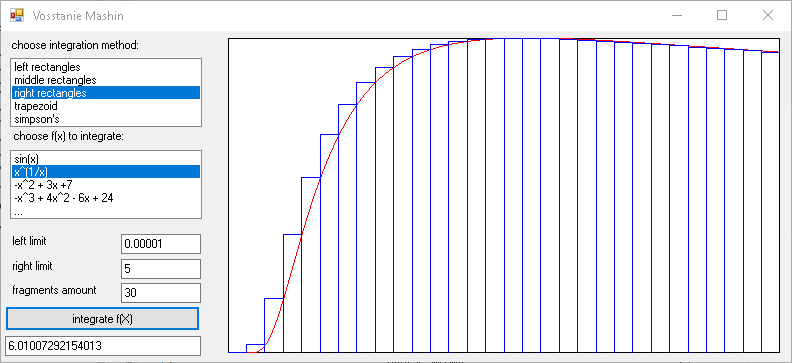


Рисунок 10(Тест2)

Результат відповідає очікуванням, присутня відносно велика похибка, зумовлена різким зростанням функції.

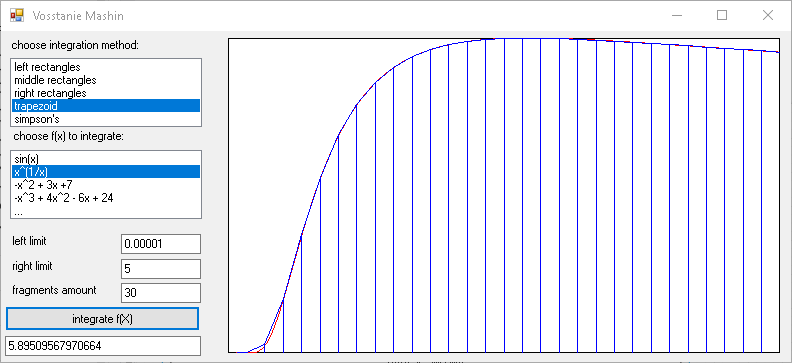


Рисунок 11(Тест2)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить.



Рисунок 12(Тест2)

Результат відповідає очікуванням, присутня відносно велика похибка, зумовлена різким зростанням функції.

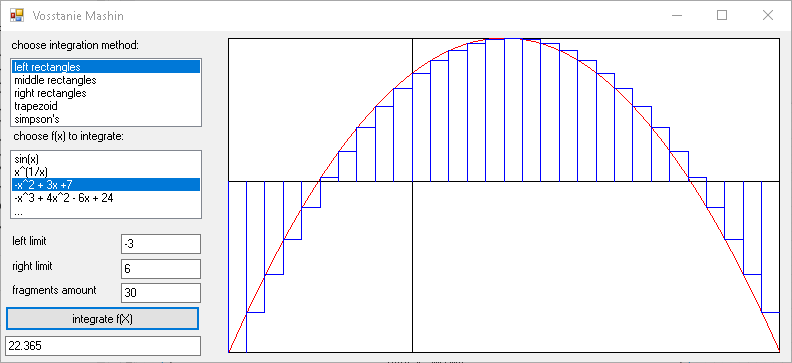


Рисунок 13(Тест3)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить.

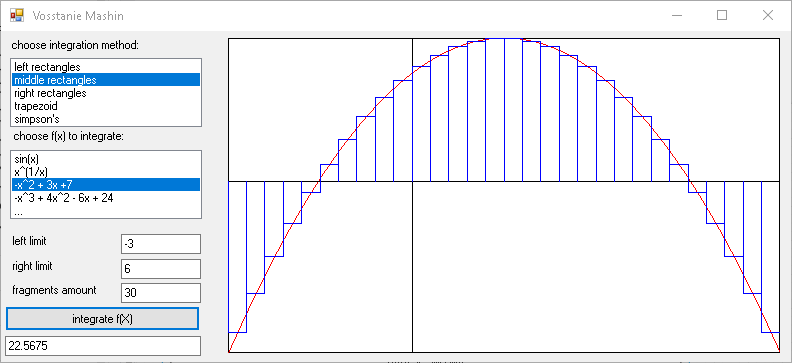


Рисунок 14(Тест3)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить.

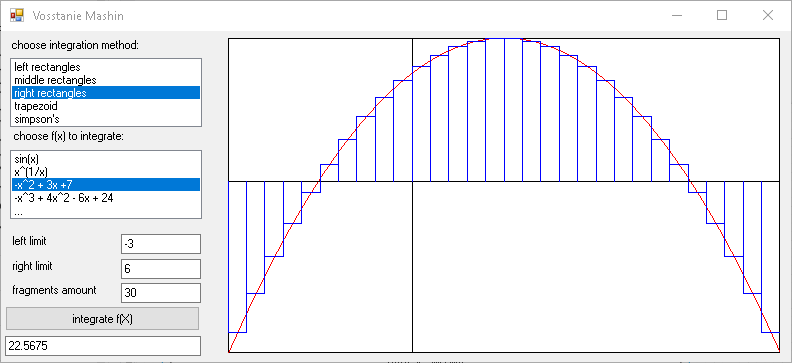


Рисунок 15(Тест3)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить.

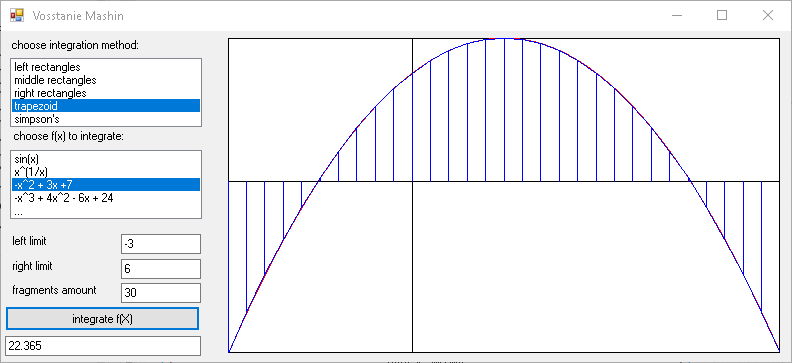


Рисунок 16(Тест3)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить.

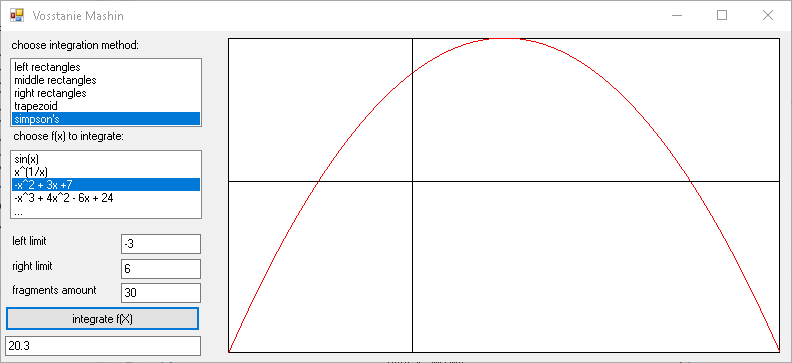


Рисунок 17(Тест3)

Результат не відповідає очікуванням, присутня відносно велика похибка, якої не має бути для парабол та функцій подібним їм за графіком.

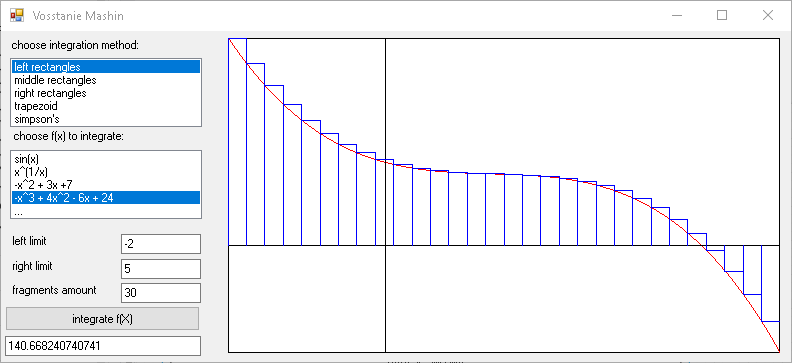


Рисунок 18(Тест4)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить, великими значеннями функції.

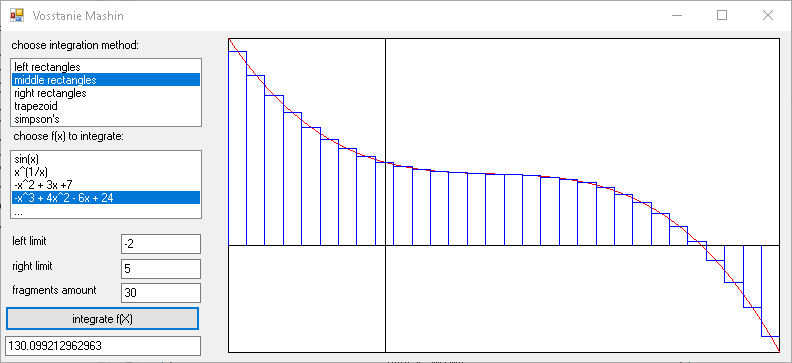
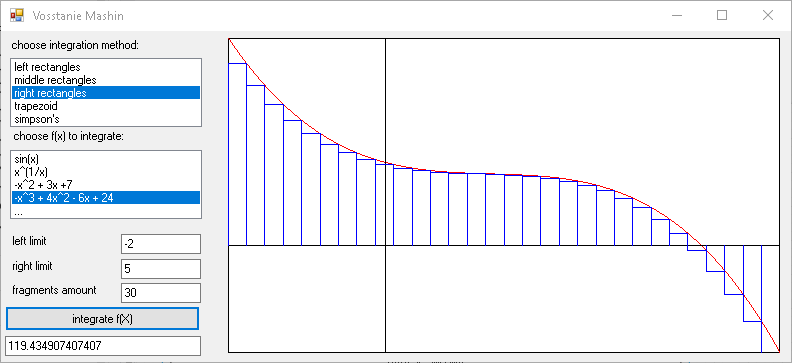


Рисунок 19(Тест4)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить.



Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить, великими значеннями функції.

Рисунок 20(Тест4)

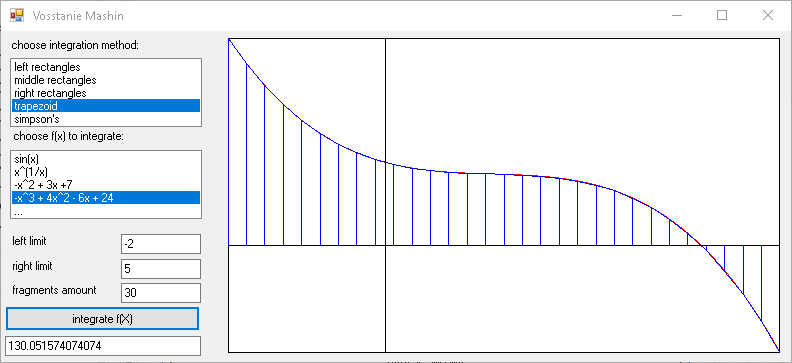


Рисунок 21(Тест4)

Результат відповідає очікуванням, присутня допустима похибка, зумовлена малою кількістю розбить.

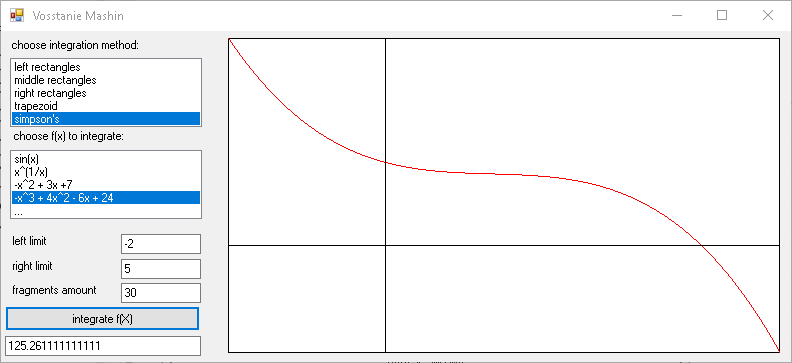


Рисунок 22(Тест4)

Результат не відповідає очікуванням, присутня відносно велика похибка, якої не має бути для парабол та функцій подібним їм за графіком.

Загалом, самий універсальний з реалізованих методів – метод трапецій, найточніший за правильних умов – метод Сімпсона; Далі йде метод середніх прямокутників, а останнє місце поділяють методи лівих та правих прямокутників. Це зумовлено тим, що в функціях, які зростають дуже різко, середні прямокутники компенсують похибку тим, що займають менше міста поза графіком, і більше в ньому; Ліві і праві прямокутники мають схожу похибку через те, що в обох випадках прямокутники або занадто вилазять за графік, або недостатньо заповнюють його. Метод Сімпсона програє в універсальності через те, що він добре інтерполює графіки параболічних функцій та схожі на них, але має відносно велику похибку в інших. Метод трапецій виграє в точності над прямокутниками через те, що займає менше місця поза графіком і більше в ньому.

# **Література**

//