Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Інститут комп’ютерних наук та інформаційних технологій
Кафедра програмного забезпечення

Звіт
До лабораторної роботи №10

На тему: «Чисельні методи інтегрування»
З дисципліни: “Чисельні методи”

Лектор:
доц. каф. ПЗ
Мельник Н.Б.
Виконав:
ст. гр. ПЗ-18
Лук’янов Н.О.
Прийняв:
проф. каф. ПЗ
Гавриш В.І.
« … » … 2023 р.

∑ = ____________

Львів – 2023

Тема: чисельні методи інтегрування.
Мета: ознайомлення на практиці з методамми чисельного інтегрування.

Завдання

Скласти програму чисельного інтегрування відповідно до варіанта:

1) за методом лівих, правих та середніх прямокутників;
2) за методом трапецій;
3) за методом Сімпсона.

$$\int_0^{\frac{5}{6}} e^{-x}\,dx = 0{,}565402$$

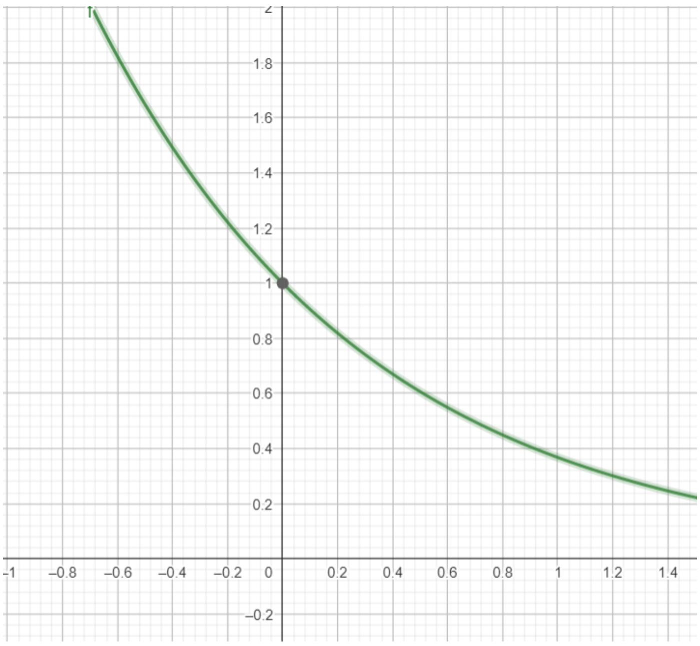

Рис. 1. Графік підінтегральної функції f(x) = e^{-x}

Метод прямокутників

Найпростішим методом наближеного обчислення інтеграла є метод прямокутників, суть якого зводиться до знаходження означеного інтеграла як суми площ n прямокутників висотою $f(x_i)$ та основою $h = \Delta x_i = x_{i+1} - x_i$, отриманих шляхом розбиття відрізка інтегрування [a,b] на n рівних частин.

Розбиття на прямокутники виконують зліва направо або справа наліво. При цьому висотою кожного елементарного прямокутника буде значення функції $y = f(x)$ у крайній лівій або крайній правій точці відповідно.

Для першого випадку отримуємо формулу **лівих прямокутників**

$$I_л = \int_a^b f(x)dx \approx h\left(f(x_0) + f(x_1) + \ldots + f(x_{n-1})\right) = h\sum_{i=0}^{n-1} f(x_i),$$

а для другого - формулу **правих прямокутників**

$$I_{пр} = \int_a^b f(x)dx \approx h\left(f(x_1) + f(x_2) + \ldots + f(x_n)\right) = h\sum_{i=1}^{n} f(x_i).$$

Тут крок інтегрування $h = \frac{b-a}{n}$. Якщо функція $f(x)$ монотонно зростає на відрізку $[a, b]$, то із використанням формул лівих і правих прямокутників отримують наближене значення інтеграла з недостачею та з надлишком відповідно.

На практиці застосовують точнішу розрахункову формулу **середніх (центральних) прямокутників**, у результаті чого отримують точніше значення інтеграла

$$I_{сер} = \int_a^b f(x)dx \approx h\left(f\left(x_0 + \frac{h}{2}\right) + f\left(x_1 + \frac{h}{2}\right) + \ldots + f\left(x_{n-1} + \frac{h}{2}\right)\right) =$$
$$= h\sum_{i=0}^{n-1} f\left(x_i + \frac{h}{2}\right).$$

Основні етапи обчислювального алгоритму, реалізованого у програмному продукті мовою С :

1) введення даних користувачем для ініціалізації змінних, що відповідають за зберігання верхньої, нижньої межі інтегрування;
2) обчислення інтегралу згідно з варіантом, за формулами лівих, правих та середніх прямокутників, за допомогою функцій MethodOfLeftRectangles() (рис. 2), MethodOfRightRectangles() (рис. 3) та MethodOfAverageRectangles() (рис. 4);
3) знаходження похибки обчислення інтеграла, за методом прямокутників, за допомогою функції CalculateEpsilon() (рис. 5);
4) результат виконання програми (рис. 6).

```
double MethodOfLeftRectangles(double upBorder, double downBorder, double n, double* epsilon)
{
    double startBorder = downBorder, h = (upBorder - downBorder) / n, result = 0, maxValue = 0, derivateAtPoint = 0;

    maxValue = fabs(CalculateDerivateAtPoint(startBorder));

    for (int i = 0; i < n - 1; i++, startBorder += h)
    {
        result += CalculateFucntion(startBorder);

        derivateAtPoint = CalculateDerivateAtPoint(startBorder);

        if (maxValue < derivateAtPoint)
        {
            maxValue = derivateAtPoint;
        }
    }

    result *= h;

    *epsilon = CalculateEpsilon(upBorder, downBorder, n, maxValue, 0);

    return result;
}
```

Рис. 2. Функція MethodOfLeftRectangles

```
double MethodOfRightRectangles(double upBorder, double downBorder, double n, double* epsilon)
{
    double startBorder = downBorder, h = (upBorder - downBorder) / n, result = 0, maxValue = 0, derivateAtPoint = 0;

    startBorder += h;

    maxValue = fabs(CalculateDerivateAtPoint(startBorder));

    for (int i = 0; i < n - 1; i++, startBorder += h)
    {
        result += CalculateFucntion(startBorder);

        derivateAtPoint = fabs(CalculateDerivateAtPoint(startBorder));

        if (maxValue < derivateAtPoint)
        {
            maxValue = derivateAtPoint;
        }
    }

    result *= h;

    *epsilon = CalculateEpsilon(upBorder, downBorder, n, maxValue, 0);

    return result;
}
```

Рис. 3. Функція MethodOfRightRectangles

```
double MethodOfAverageRectangles(double upBorder, double downBorder, double n, double* epsilon)
{
    double startBorder = downBorder, h = (upBorder - downBorder) / n, result = 0, maxValue = 0, derivateAtPoint = 0;

    maxValue = fabs(CalculateFucntion(startBorder + (h / 2.0)));

    for (int i = 0; i < n - 1; i++, startBorder += h)
    {
        result += CalculateFucntion(startBorder + (h / 2.0));

        derivateAtPoint = CalculateFucntion(startBorder + (h / 2.0));

        if (maxValue < derivateAtPoint)
        {
            maxValue = derivateAtPoint;
        }
    }

    result *= h;

    *epsilon = CalculateEpsilon(upBorder, downBorder, n, maxValue, 1);

    return result;
}
```

Рис. 4. Функція MethodOfAverageRectangles

```
double CalculateEpsilon(double upBorder, double downBorder, double n, double maxValue, int numberOfMethod)
{
    switch (numberOfMethod)
    {
    case 0:
        return pow((upBorder - downBorder), 2) / (2.0 * n) * maxValue;
    case 1:
        return pow((upBorder - downBorder), 3.0) / (24 * n * n) * maxValue;
    case 2:
        return pow((upBorder - downBorder), 3.0) / (12 * n * n) * maxValue;
    case 3:
        return pow((upBorder - downBorder), 5.0) / (180 * pow(n, 4.0)) * maxValue;
    default:
        return 0;
    }
}
```

Рис. 5. Функція CalculateEpsilon

```
Microsoft Visual Studio Debug Console
The function: e^(-x)
--------------------------------------------------
Enter the limits of integration:
UpBorder - 0.8333
DownBorder - 0
Enter the number of steps - 100
--------------------------------------------------
Choose the action:
1)Method of rectangles
2)Method of Trapeze
3)Method of Simpson
Your choice - 1
--------------------------------------------------
You have choosed the method of rectangles!
Choose the method:
1)Left rectangles
2)Right rectangles
3)Average rectangles
4)Do all types method
Your choice - 4
--------------------------------------------------
You have choosed all types of method:
        The result - 0.56409
        Epsilon - 0.0034719444500000018
--------------------------------------------------
You have choosed the right rectangles method:
        The result - 0.55941
        Epsilon - 0.0034431329467461625 9
--------------------------------------------------
You have choosed the average rectangles method:
        The result - 0.56175
        Epsilon - 0.0000024009516578381 4
--------------------------------------------------
```

Рис. 6. Результат виконання програми

Метод трапецій

Метод трапецій полягає в тому, що відрізок інтегрування [a, b] розбивають на n рівних відрізків, а криву, описану підінтегральну функцією $f(x)$, замінюють на кожному із цих відрізків кусково-лінійною функцією $\varphi(x)$, отриманою стягуванням хорд, які проходять через точки $(x_{i-1}, f(x_{i-1}))$ та $(x_i, f(x_i))$ $(i = \overline{1, n})$.

Значення інтеграла знаходять як суму площ S_i $(i = \overline{0, n})$ прямокутних трапецій з висотою $h = \frac{b-a}{n}$.

Площу кожної i-ої елементарної трапеції визначають за формулою

$$S_i = h\,\frac{f(x_i) + f(x_{i+1})}{2}\ .$$

Відповідно на всьому відрізку інтегрування $[a, b]$ площу складеної фігури визначають сумою площ усіх елементарних трапецій. У результаті отримують таку формулу

$$I_{mp} = \int_a^b f(x)dx \approx h\left(\frac{f(x_0)+f(x_1)}{2} + \frac{f(x_1)+f(x_2)}{2} + \ldots + \frac{f(x_{n-1})+f(x_n)}{2}\right) = \\ = h\sum_{i=0}^{n-1}\frac{f(x_i)+f(x_{i+1})}{2}.$$

Оскільки в наведеній формулі під знаком суми величини $f(x_i)$, $(i = \overline{1, n-1})$ зустрічаються двічі, то перепишемо її у вигляді

$$I_{mp} = \int_a^b f(x)dx \approx h\left(\frac{f(x_0)}{2} + f(x_1) + f(x_2) + \ldots + f(x_{n-1}) + \frac{f(x_n)}{2}\right) = \\ = h\left(\frac{f(x_0)+f(x_n)}{2} + \sum_{i=0}^{n-1} f(x_i)\right).$$

Основні етапи обчислювального алгоритму, реалізованого у програмному продукті мовою С :

1) введення даних користувачем для ініціалізації змінних, що відповідають за зберігання верхньої, нижньої межі інтегрування;
2) обчислення інтегралу згідно з варіантом, за формулою трапецій, за допомогою функції MethodOfTrapeze() (рис. 7);
3) знаходження похибки обчислення інтеграла, за методом трапецій, за допомогою функції CalculateEpsilon();
4) результат виконання програми (рис. 8).

```
double MethodOfTrapeze(double upBorder, double downBorder, double n, double* epsilon)
{
    double startBorder = downBorder, h = (upBorder - downBorder) / n, result = 0, maxValue = 0, derivateAtPoint = 0;

    maxValue = fabs(CalculateFucntion(startBorder));

    for (int i = 0; i < n - 1; i++, startBorder += h)
    {
        result += (CalculateFucntion(startBorder) + CalculateFucntion(startBorder + h)) / 2.0;

        derivateAtPoint = CalculateFucntion(startBorder);

        if (maxValue < derivateAtPoint)
        {
            maxValue = derivateAtPoint;
        }
    }

    result *= h;

    *epsilon = CalculateEpsilon(upBorder, downBorder, n, maxValue, 2);

    return result;
}
```

Рис. 7. Функція MethodOfTrapeze

```
Microsoft Visual Studio Debug Console
The function: e^(-x)
--------------------------------------------------
Enter the limits of integration:
UpBorder - 0.8333
DownBorder - 0
Enter the number of steps - 100
--------------------------------------------------
Choose the action:
1)Method of rectangles
2)Method of Trapeze
3)Method of Simpson
Your choice - 2
--------------------------------------------------
You have choosed the method of Trapeze:
        The result - 0.56175
        Epsilon - 0.00000482195218364167
--------------------------------------------------
```

Рис. 8. Результат виконання програми

Метод Сімпсона

Даний метод полягає в тому, що криву, описану підінтегральною функцією $f(x)$, на елементарних відрізках заміняють параболою. Поділимо відрізок інтегрування $[a, b]$ на парну кількість n рівних частин з кроком $h = \frac{b-a}{n}$. На кожному елементарному відрізку $[x_0, x_2]$, $[x_2, x_4]$, … , $[x_{i-1}, x_{i+1}]$, … , $[x_{n-2}, x_n]$ підінтегральну функцію $f(x)$ замінимо інтерполяційним поліномом другого степеня (квадратичною параболою). Тоді обчислення означеного інтеграла зводиться до обчислення суми площ S_i, ($\mathrm{i} = \overline{1, n}$) криволінійних трапецій.

Площу S_i кожної елементарної криволінійної трапеції визначають за формулою Сімпсона

$$S_i = \frac{h}{3}\left(f(x_i) + 4f(x_{i+1}) + f(x_{i+2})\right).$$

Послідовно обчислюємо за формулою площі n криволінійних трапецій S_i ($i = \overline{1,n}$)

$$S_1 = \int_{x_0}^{x_2} f(x)dx = \frac{h}{3}\left(f(x_0) + 4f(x_1) + f(x_2)\right),$$

$$S_2 = \int_{x_2}^{x_4} f(x)dx \approx \frac{h}{3}\left(f(x_2) + 4f(x_3) + f(x_4)\right),$$

...

$$S_n = \int_{x_{2n-2}}^{x_{2n}} f(x)dx \approx \frac{h}{3}\left(f(x_{2n-2}) + 4f(x_{2n-1}) + f(x_{2n})\right).$$

Знайдемо суму площ всіх криволінійних трапецій.

$$\sum_{i=1}^{n} S_i = \frac{h}{3}\Big(f(x_0) + f(x_{2n}) + 4\big(f(x_1) + \ \dots \ + f(x_{2n-1})\big) + $$
$$+ 2\big(f(x_2) + \ \dots \ + f(x_{2n-2})\big)\Big).$$

Тоді розрахункова формула методу Сімпсона набуде такого вигляду

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left(f(x_0) + f(x_{2n}) + 4\sum_{i=1}^{n} f(x_{2i-1}) + 2\sum_{i=1}^{n-1} f(x_{2i})\right)$$

Основні етапи обчислювального алгоритму, реалізованого у програмному продукті мовою С :

1) введення даних користувачем для ініціалізації змінних, що відповідають за зберігання верхньої, нижньої межі інтегрування;
2) обчислення інтегралу згідно з варіантом, за формулою Сімпсона, за допомогою функції MethodOfSimpson() (рис. 9);
3) знаходження похибки обчислення інтеграла, за методом трапецій, за допомогою функції CalculateEpsilon();
4) результат виконання програми (рис. 10).

```c
double MethodOfSimpson(double upBorder, double downBorder, double n, double* epsilon)
{
    double startBorder = downBorder, h = (upBorder - downBorder) / n, result = 0, maxValue = 0, derivateAtPoint = 0, sumOfEven = 0, sumOfOdd = 0;

    maxValue = fabs(CalculateFucntion(startBorder));
    result += CalculateFucntion(downBorder) + CalculateFucntion(upBorder);

    for (int i = 1; i < n; i += 2)
    {
        double x = downBorder + i * h;
        result += 4 * CalculateFucntion(x);
    }

    for (int i = 2; i < n; i += 2)
    {
        double x = downBorder + i * h;
        result += 2 * CalculateFucntion(x);
    }

    for (int i = 1; i < n; i++, startBorder += h)
    {

        derivateAtPoint = fabs(CalculateFucntion(startBorder));

        if (maxValue < derivateAtPoint)
        {
            maxValue = derivateAtPoint;
        }
    }

    result *= (h / 3.0);

    *epsilon = CalculateEpsilon(upBorder, downBorder, n, maxValue, 3);

    return result;
}
```

Рис. 9. Функція MethodOfSimpson

```
Microsoft Visual Studio Debug Console
The function: e^(-x)
--------------------------------------------------
Enter the limits of integration:
UpBorder - 0.8333
DownBorder - 0
Enter the number of steps - 100
--------------------------------------------------
Choose the action:
1)Method of rectangles
2)Method of Trapeze
3)Method of Simpson
Your choice - 3
--------------------------------------------------
You have choosed the method of Simpson:
        The result - 0.56539
        Epsilon - 0.00000000002232206683
--------------------------------------------------
```

Рис. 10. Результат виконання програми

Висновки

У результаті виконання лабораторної роботи, розроблено обчислюваний алгоритм, реалізований мовою програмування C чисельного інтегрування, а саме за методом лівих, правих та середніх прямокутників, за методом трапецій та за методом Сімпсона. Визначено похибки обчислення інтеграла за методом лівих прямокутників – $3{,}4719 * 10^{-3}$, за методом правих прямокутників – $3{,}4431 * 10^{-3}$, за методом середніх прямокутників – $2{,}4009 * 10^{-6}$, за методом трапецій -

$4{,}8219 * 10^{-6}$, а за методом Сімпсона – $2{,}2322 * 10^{-11}$. Програмний продукт розроблений у середовищі Microsoft Visual Studio мовою програмування С.

Додаток

Назва файлу: **Lab10.c**

```
#include <stdio.h>
#include <math.h>

double MethodOfLeftRectangles(double, double, double, double*);
double CalculateEpsilon(double, double, double, double, int);
double MethodOfRightRectangles(double, double, double, double*);
double MethodOfAverageRectangles(double, double, double, double*);
double MethodOfTrapeze(double, double, double, double*);
double MethodOfSimpson(double, double, double, double*);
double CalculateDerivateAtPoint(double);
double CalculateFucntion(double);
void PrintLine(int, char);

int main()
{
	double epsilon = 0.0, result = 0.0, upBorder = 0.0, downBorder = 0.0;
	int choice = 0, countOfSteps = 0.0;

	printf("The function: e^(-x)\n");
	PrintLine(50, '-');

	printf("Enter the limits of integration:\nUpBorder - ");
	scanf_s("%lf", &upBorder);
	printf("DownBorder - ");
	scanf_s("%lf", &downBorder);
	printf("Enter the number of steps - ");
	scanf_s("%d", &countOfSteps);
	PrintLine(50, '-');

	printf("Choose the action:\n1)Method of rectangles\n2)Method of Trapeze\n3)Method of 
Simpson\nYour choice - ");
	scanf_s("%d", &choice);
	PrintLine(50, '-');

	switch (choice)
	{
	case 1:
		printf("You have choosed the method of rectangles!\nChoose the method:\n1)Left 
rectangles\n2)Right rectangles\n3)Average rectangles\n4)Do all types method\nYour choice - 
");
		scanf_s("%d", &choice);
		PrintLine(50, '-');

		switch (choice)
		{
		case 1:
			printf("You have choosed the left rectangles method:\n");
			result = MethodOfLeftRectangles(upBorder, downBorder, countOfSteps, 
&epsilon);
			printf("\tThe result - %.5f\n\tEpsilon - %.20f\n", result, epsilon);
			PrintLine(50, '-');
			break;
		case 2:
			printf("You have choosed the right rectangles method:\n");
			result = MethodOfRightRectangles(upBorder, downBorder, countOfSteps, 
&epsilon);
			printf("\tThe result - %.5f\n\tEpsilon - %.20f\n", result, epsilon);
			PrintLine(50, '-');
			break;
		case 3:
			printf("You have choosed the average rectangles method:\n");
			result = MethodOfAverageRectangles(upBorder, downBorder, countOfSteps, 
&epsilon);
			printf("\tThe result - %.5f\n\tEpsilon - %.20f\n", result, epsilon);
```

```
                        PrintLine(50, '-');
                        break;
                case 4:
                        printf("You have choosed all types of method:\n");
                        result = MethodOfLeftRectangles(upBorder, downBorder, countOfSteps,
&epsilon);
                        printf("\tThe result - %.5f\n\tEpsilon - %.20f\n", result, epsilon);
                        PrintLine(50, '-');

                        printf("You have choosed the right rectangles method:\n");
                        result = MethodOfRightRectangles(upBorder, downBorder, countOfSteps,
&epsilon);
                        printf("\tThe result - %.5f\n\tEpsilon - %.20f\n", result, epsilon);
                        PrintLine(50, '-');

                        printf("You have choosed the average rectangles method:\n");
                        result = MethodOfAverageRectangles(upBorder, downBorder, countOfSteps,
&epsilon);
                        printf("\tThe result - %.5f\n\tEpsilon - %.20f\n", result, epsilon);
                        PrintLine(50, '-');
                        break;
                default:
                        printf("Ooops! The incorrect input....\n");
                        break;
                }
                break;
        case 2:
                printf("You have choosed the method of Trapeze:\n");
                result = MethodOfTrapeze(upBorder, downBorder, countOfSteps, &epsilon);
                printf("\tThe result - %.5f\n\tEpsilon - %.20f\n", result, epsilon);
                PrintLine(50, '-');
                break;
        case 3:
                printf("You have choosed the method of Simpson:\n");
                result = MethodOfSimpson(upBorder, downBorder, countOfSteps, &epsilon);
                printf("\tThe result - %.5f\n\tEpsilon - %.20f\n", result, epsilon);
                PrintLine(50, '-');
                break;
        default:
                printf("Ooops! The incorrect input....\n");
                break;
        }
}

double CalculateEpsilon(double upBorder, double downBorder, double n, double maxValue, int
numberOfMethod)
{
        switch (numberOfMethod)
        {
        case 0:
                return pow((upBorder - downBorder), 2) / (2.0 * n) * maxValue;
        case 1:
                return pow((upBorder - downBorder), 3.0) / (24 * n * n) * maxValue;
        case 2:
                return pow((upBorder - downBorder), 3.0) / (12 * n * n) * maxValue;
        case 3:
                return pow((upBorder - downBorder), 5.0) / (180 * pow(n, 4.0)) * maxValue;
        default:
                return 0;
        }
}

double CalculateDerivateAtPoint(double x)
{
        return -(exp(-x));
}

double CalculateFucntion(double x)
```

```
{
	return (exp(-x));
}

double MethodOfRightRectangles(double upBorder, double downBorder, double n, double*
epsilon)
{
	double startBorder = downBorder, h = (upBorder - downBorder) / n, result = 0,
maxValue = 0, derivateAtPoint = 0;

	startBorder += h;

	maxValue = fabs(CalculateDerivateAtPoint(startBorder));

	for (int i = 0; i < n - 1; i++, startBorder += h)
	{
		result += CalculateFucntion(startBorder);

		derivateAtPoint = fabs(CalculateDerivateAtPoint(startBorder));

		if (maxValue < derivateAtPoint)
		{
			maxValue = derivateAtPoint;
		}
	}

	result *= h;

	*epsilon = CalculateEpsilon(upBorder, downBorder, n, maxValue, 0);

	return result;
}

double MethodOfLeftRectangles(double upBorder, double downBorder, double n, double* epsilon)
{
	double startBorder = downBorder, h = (upBorder - downBorder) / n, result = 0,
maxValue = 0, derivateAtPoint = 0;

	maxValue = fabs(CalculateDerivateAtPoint(startBorder));

	for (int i = 0; i < n - 1; i++, startBorder += h)
	{
		result += CalculateFucntion(startBorder);

		derivateAtPoint = CalculateDerivateAtPoint(startBorder);

		if (maxValue < derivateAtPoint)
		{
			maxValue = derivateAtPoint;
		}
	}

	result *= h;

	*epsilon = CalculateEpsilon(upBorder, downBorder, n, maxValue, 0);

	return result;
}

double MethodOfAverageRectangles(double upBorder, double downBorder, double n, double*
epsilon)
{
	double startBorder = downBorder, h = (upBorder - downBorder) / n, result = 0,
maxValue = 0, derivateAtPoint = 0;

	maxValue = fabs(CalculateFucntion(startBorder + (h / 2.0)));

	for (int i = 0; i < n - 1; i++, startBorder += h)
```

```
	{
		result += CalculateFucntion(startBorder + (h / 2.0));

		derivateAtPoint = CalculateFucntion(startBorder + (h / 2.0));

		if (maxValue < derivateAtPoint)
		{
			maxValue = derivateAtPoint;
		}
	}

	result *= h;

	*epsilon = CalculateEpsilon(upBorder, downBorder, n, maxValue, 1);

	return result;
}

double MethodOfTrapeze(double upBorder, double downBorder, double n, double* epsilon)
{
	double startBorder = downBorder, h = (upBorder - downBorder) / n, result = 0,
maxValue = 0, derivateAtPoint = 0;

	maxValue = fabs(CalculateFucntion(startBorder));

	for (int i = 0; i < n - 1; i++, startBorder += h)
	{
		result += (CalculateFucntion(startBorder) + CalculateFucntion(startBorder +
h)) / 2.0;

		derivateAtPoint = CalculateFucntion(startBorder);

		if (maxValue < derivateAtPoint)
		{
			maxValue = derivateAtPoint;
		}
	}

	result *= h;

	*epsilon = CalculateEpsilon(upBorder, downBorder, n, maxValue, 2);

	return result;
}

double MethodOfSimpson(double upBorder, double downBorder, double n, double* epsilon)
{
	double startBorder = downBorder, h = (upBorder - downBorder) / n, result = 0,
maxValue = 0, derivateAtPoint = 0, sumOfEven = 0, sumOfOdd = 0;

	maxValue = fabs(CalculateFucntion(startBorder));
	result += CalculateFucntion(downBorder) + CalculateFucntion(upBorder);

	for (int i = 1; i < n; i += 2)
	{
		double x = downBorder + i * h;
		result += 4 * CalculateFucntion(x);
	}

	for (int i = 2; i < n; i += 2)
	{
	 double x = downBorder + i * h;
	 result += 2 * CalculateFucntion(x);
    }

	for (int i = 1; i < n; i++, startBorder += h)
	{
```

```
			derivateAtPoint = fabs(CalculateFucntion(startBorder));

			if (maxValue < derivateAtPoint)
			{
				maxValue = derivateAtPoint;
			}
		}

		result *= (h / 3.0);

		*epsilon = CalculateEpsilon(upBorder, downBorder, n, maxValue, 3);

		return result;
}

void PrintLine(int number, char symbol)
{
		while (number > 0)
		{
			printf("%c", symbol);
			number--;
		}
		printf("\n");
}
```