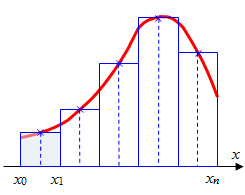
Некоторые математические задачи не всегда поддаются аналитическому решению, но в то же время для них могут существовать методы численного решения, которые позволяют получить ответ для заданных входных данных. К таким задачам можно отнести, например, задачу интегрирования, для которой не всегда можно найти неопределенный интеграл, но относительно легко найти значение на интервале. Для этого используется определение интеграла как площади фигуры, описанной кривой функции. Площадь в свою очередь можно найти как сумму площадей элементарных фигур, которые будут апроксимировать кривую. Например, для функции, показанной на рисунке ниже, приближенное значение интеграла будет состоять из суммы площадей прямоугольников, лежащих под кривой на интервале [x0;xn]. При длине отрезка → 0, погрешность будет минимальна.



*Рисунок 1 – Интегрирование методом прямоугольников*

Для представления кривой определим абстрактный класс «Уравнение», в котором определим метод, рассчитывающий значение функции в точке, после чего можно будет написать универсальный алгоритм интегрирования, использующий данный абстрактный класс. Добавляя производные от «Уравнения» классы, мы добьёмся того, что наша программа сможет интегрировать большой класс функций.

Опишем класс «Уравнение» следующим образом:

public abstract class Equation

{

public abstract double GetValue(double x);

}

Опишем класс интегрирования. Будем использовать метод прямоугольников:

public class Integrator

{

private readonly Equation equation;

/// <summary>

/// Конструктор класса "интегратор"

/// </summary>

/// <param name="equation">интегрируемое уравнение</param>

public Integrator( Equation equation)

{

//проверяем допустимость параметров:

if (equation == null) {

throw new ArgumentNullException();

}

this.equation = equation;

}

/// <summary>

/// Функция интегрирования

/// </summary>

/// <param name="x1">левая граница интегрирования</param>

/// <param name="x2">правая граница интегрирования</param>

public double Integrate(double x1,double x2)

{

//проверяем допустимость параметров:

if (x1 >= x2) {

throw new ArgumentException("Правая граница интегирования должны быть больше левой!");

}

/\* для интегирования разобъем исходный отрезок на 100 точек.

\* Считаем значение функции в точке, умножаем на ширину интервала.

\* Площадь полученного прямоугольника приблизительно равна значению интеграла на этом отрезке

\* суммируем значения площадей, получаем значение интеграла на отрезке [X1;X2]\*/

int N = 100; //количество интервалов разбиения

//определяем ширину интервала:

double h = (x2 - x1) / N;

double sum = 0; //"накопитель" для значения интеграла

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

sum = sum + equation.GetValue(x1 + i \* h) \* h;

}

return sum;

}

}

Осталось написать класс, являющийся наследником Equation и переопределить в нем функцию Value:

/// <summary>

/// Класс, представляющий квадратное уравнение

/// </summary>

public class QuadEquation : Equation

{

private readonly double a;

private readonly double b;

private readonly double c;

public QuadEquation(double a, double b, double c)

{

this.a = a;

this.b = b;

this.c = c;

}

public override double GetValue(double x)

{

return a \* x \* x + b \* x + c;

}

}

Пример использования разработанных классов:

Equation e = new QuadEquation(0, 20, 0); //создаем объект класса "кв. уравнение"

Integrator i1 = new Integrator(e); //создаем интегратор для этого уравнения

double integrValue = i1.Integrate(10,30); //вызываем интегрирование на интвервале 10;30

//создаем интегратор для другого уравнения:

Integrator i2 = new Integrator(new QuadEquation(-3, 0, 2.3));

integrValue = i2.Integrate(-2, 100); //вызываем ф-ю интегрирования

**Задание**

1. Реализуйте абстрактный класс Equation – определяющий поведение функции вида y = f(x) – функции одной переменной.
2. На базе класса Equation создайте 2 производных класса для уравнений, представленных ниже. Варианты уравнений согласуются с преподавателем.
   1. a\*x\*x + b\*x + c;
   2. sin(a\*x)/x;
   3. x\*x\*cos(x - a)/b;
   4. x\*|sin(a\*x)|, здесь | | - модуль числа;
   5. x\*cos(x)/a;
   6. a\*x\*| sin(x) |, здесь | | - модуль числа.
3. Напишите функцию для отображения графика функций на заданном отрезке [X1; X2]. Возможный прототип функции в случае, если вы используете Windows Forms:

void DrawFunction( double x1, double x2, Series series, Equation equation)

{

//код рисования графика функции...

}

1. Разработайте 2 класса -интегратора из вариантов, представленных ниже. Варианты заданий согласуются с преподавателем. Во всех вариантах должны настраиваться (задаваться пользователем) либо шаг разбиения h, либо количество разбиений N.
2. Интегрирование методом прямоугольников.

Формула для нахождения приближенного значения интеграла методом прямоугольников:

http://aco.ifmo.ru/el_books/numerical_methods/lectures/glava2/glava2_clip_image013_0000.png

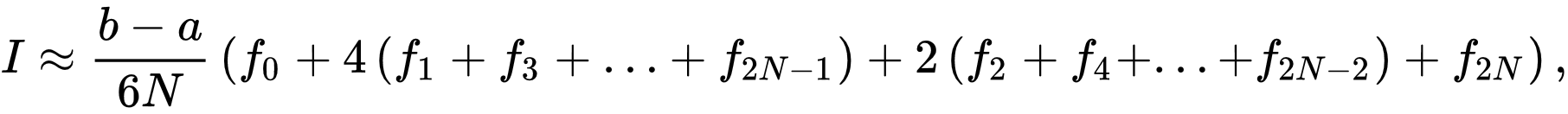
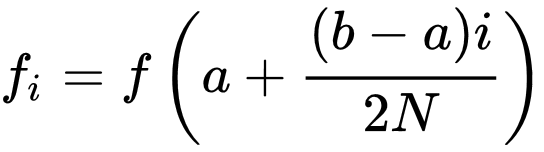
1. Интегрирование методом трапеций.

Формула для нахождения приближенного значения интеграла методом трапеций:

http://aco.ifmo.ru/el_books/numerical_methods/lectures/glava2/glava2_clip_image011_0001.png

1. Интегрирование методом парабол (метод Симпсона).

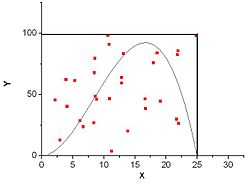
Формула для нахождения приближенного значения интеграла методом Симпсона:

1. Интегрирование методом Монте-Карло

Для определения площади под графиком функции можно использовать следующий стохастический алгоритм:

* ограничим функцию прямоугольником, площадь которого Spr можно легко вычислить;
* «набросаем» в этот прямоугольник некоторое количество точек ( N штук), координаты которых будем выбирать случайным образом;
* определим число точек ( K штук), которые попадут под график функции;
* площадь области, ограниченной функцией и осями координат, S вычисляется выражением S = Spr \* K / N.



*Рисунок 2 – Геометрическая интерпретация метода Монте-Карло*

1. Для разработанных классов интегрирования выделите базовый класс, добавьте метод или свойство, позволяющее пользователю узнать название метода, реализованного в данном классе.

**Дополнительное задание**

1. Решить задание, используя делегаты – вместо класса Equation в классе интегрирования использовать делегат, ссылающийся на вещественную функцию с одним вещественным параметром.