федеральное государственное автономное образовательное учреждение высшего образования
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
кафедра вычислительной техники

Отчёт по лабораторной работе №2
Вариант: метод Симпсона

Выполнил Ощепков Артём,
группа Р3202
Преподаватель Исаев И.В.

Санкт-Петербург 2017

Текст задания

Пользователь выбирает функцию, интеграл которой он хочет вычислить (3-5 функций), из тех, которые предлагает программа.

Пользователь задает пределы интегрирования и точность.

В результате должны получить:

- значение интеграла
- количество разбиений, на которое пришлось разбить
- полученную погрешность

Для оценки погрешности использовать оценку Рунге.

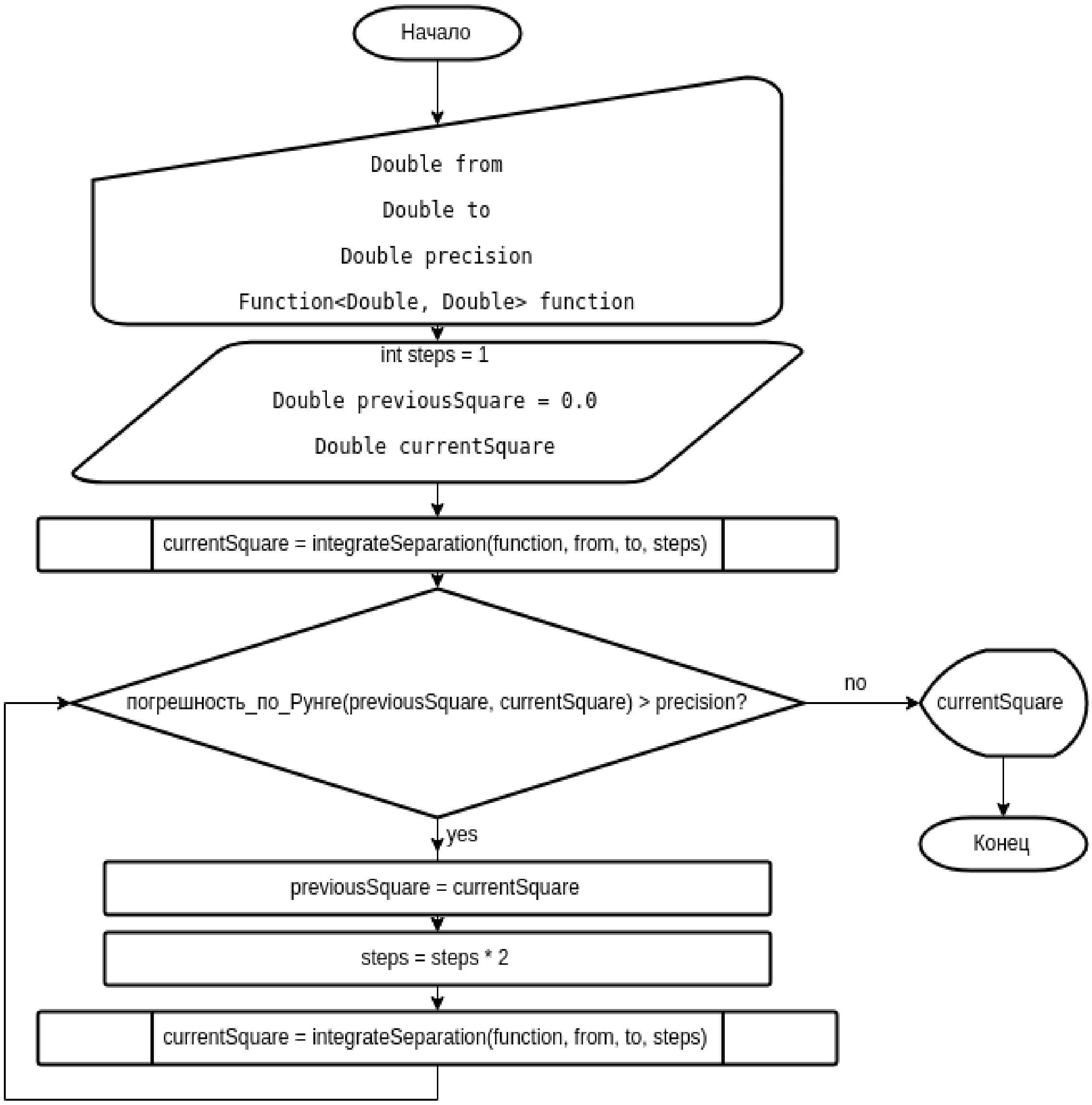
Описание метода

Интегрируем функцию, используя определение определённого интеграла и конечное фиксированное количество разбиений. Однако элементарная интегральная сумма («площадь одного столбика») заменяется параболой и считается с помощью формулы Симпсона:

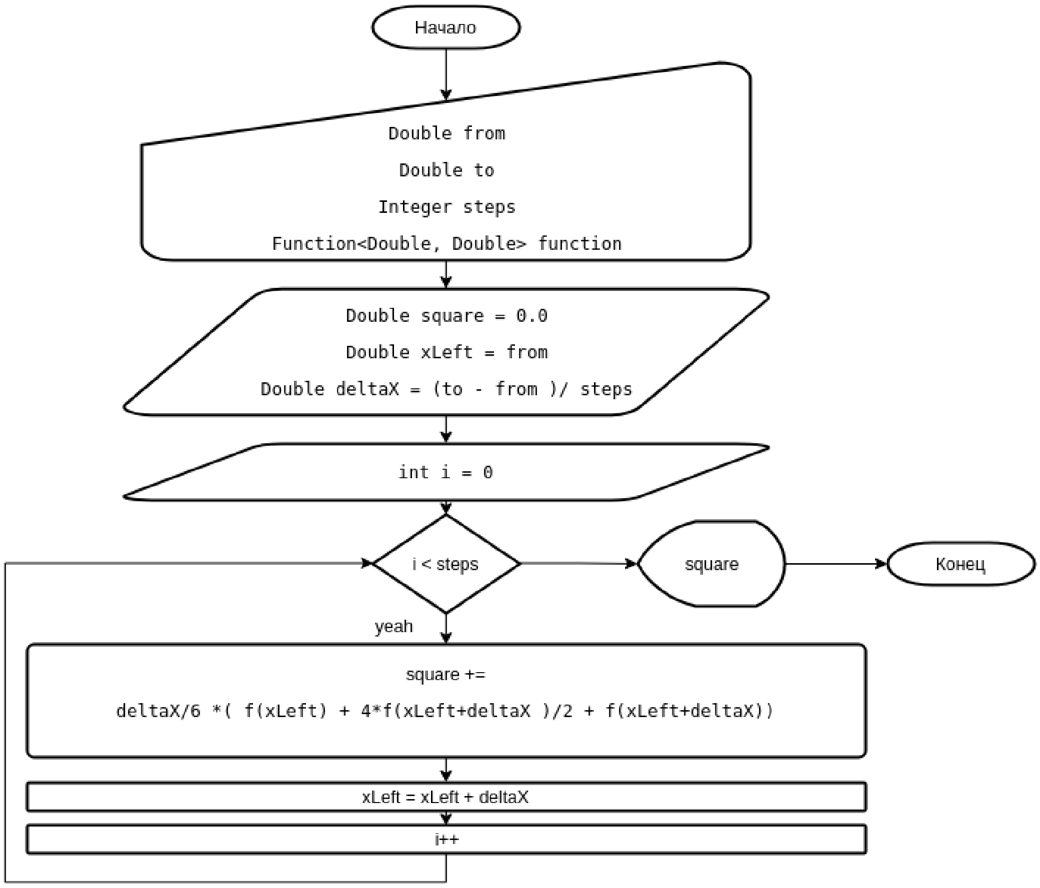
$$\int_a^b f(x)dx \approx \int_a^b p_2(x)dx = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right),$$

Блок-схема

Интеграл с указанной точностью:



Интеграл с указанным количеством разбиений:



Текст разработанной программы

```
public abstract class Integral implements Integrator {
    public Double integrate(Function<Double, Double> function,
                            Double from, Double to, Double precision) {
        // The number of the separations
        Integer steps = 1;
        // We need to do this to use the Runge's rule  (This rule is based on a difference between
            // integration with n separations and 2n separation)
         Double previousSquare = 0.0;
        Double currentSquare = integrateSeparation(function, from, to, steps);
        // As the theory require, we're integrating the function until
         // The infelicity is greater than precision that an user set
           while (infelicityRunge(previousSquare,currentSquare) > precision){
                 previousSquare = currentSquare;
                  steps *= 2;
            currentSquare = integrateSeparation(function, from, to, steps);
```

```
        }
      this.lastIntegrateInfelicity = infelicityRunge(previousSquare,currentSquare);
      this.lastIntegrateSteps = steps;
      return currentSquare;
  }
  // Integrate the function clearly setting the numbers of the separations
  public Double integrateSeparation(
                    Function<Double, Double> function, Double from, Double to, Integer steps){
        Double square = 0.0; // The undef integral is a square under the plot
        Double xLeft = from; // Current point to get function value
        Double deltaX = (to - from )/ steps.doubleValue(); // The offset to get the second point
          for (int i = 0; i < steps; i++ ) {
                    // The current simple piece's square is getting from an any integration method
                   square += getAtomSquare(function, xLeft, xLeft + deltaX);
                   // The primitive is the square on the every single step
                   primitiveValues.add(new Pair<>(xLeft+deltaX, square));
                    xLeft += deltaX;
            }
      return square;
  }
  protected abstract Double getAtomSquare(Function<Double,Double> function, Double x1,
Double x2); //
/* В классе — потомке находится реализация:
@Override
protected Double getAtomSquare(
                                  Function<Double, Double> function, Double x1, Double x2) {
  // Just the formula. Square under an approximate parabola
  return (x2-x1)/6.0 * (function.apply(x1) + 4.0*function.apply((x1+x2 )/ 2.0) +
function.apply(x2));
*/
}
  // Оценка погрешности правилом Рунге
  protected Double infelicityRunge(Double I1,Double I2){
      return Math.abs(I1 - I2)/rungeRuleCoeff;
  }
  protected Double rungeRuleCoeff = 15.0;
```

```
    // Сколько разбиений было совершено при последнем интегрировании?
    protected Integer lastIntegrateSteps;
}
```

Результаты тестирования

$\int_{1}^{2.718281828459045235360} 1/x\,dx\,,precision=0.25:$ 1.0078899408946135, infelicity: 0.067

$\int_{10}^{0} 10\,dx\,,precision=0.0025:$ -100.0, infelicity: 0.0

$\int_{1}^{2} \frac{\sin x}{x}\,dx\,,precision=0.0001:$ `0.6593312109452119,` inflecity: 1.32297706789064E-6

Выводы

Разрабатывая данную программу, я забавным образом решил задачу ввода функции пользователем, что меня порадовало и послужило мотивацией для усердной и, что самое важное, *творческой* работы над лабораторной.

Моё понимание работы JVM немного улучшилось. Также я научился пользоваться графиками в JavaFx и бояться Generic'ов, поупражнялся в разделении программы на модули.

Внезапно, код, отвечающий за математику, заработал с первого раза. Я очень удивился этому факту, но на следующий же день обнаружил две серьёзные ошибки.

В итоге, определённые интегралы, находясь под моим руководством, радостно считаются методом Симпсона.