Практическое занятие №2

Численные методы решения ОДУ

Вариант 8

Формулировка задания: реализовать на любом языке программирования программу для численного интегрирования дифференциальных уравнений в соответствие с вариантом методом Эйлера, усовершенствованным методом Эйлера и методом Рунге-Кутта. Начальные условия задаются с клавиатуры. Выходными данными является набор пар значений X и Y для функции в заданном отрезке.

Функции:

Метод Эйлера - y' = x + cos(y/sqrt(2))

Усовершенствованный метод Эйлера - y' = 0,145(x^2+cos(0,5x))+0,842y

Метод Рунге-Кутта - y' = 1 - sin(x+y)+0,5y/(x+2)

Ход решения

Было принято решение реализовать решения всех трех уравнений как части одной программы, последовательно выполняющей решения разными методами. В связи с тем, что набор входных данных не изменяется (меняются только значения, набор переменных неизменен), было принято решения вынести считывание начальных условий в отдельную функцию. Ее словесный алгоритм приведен ниже:

1) Запросить у пользователя начало отрезка
2) Считать начало отрезка
3) Запросить у пользователя конец отрезка
4) Считать конец отрезка
5) Запросить у пользователя шаг отрезка
6) Считать шаг отрезка
7) Запросить у пользователя начальное значение Y0
8) Считать начальное значение Y0

Для печати результатов использовалась функция со следующим алгоритмом:

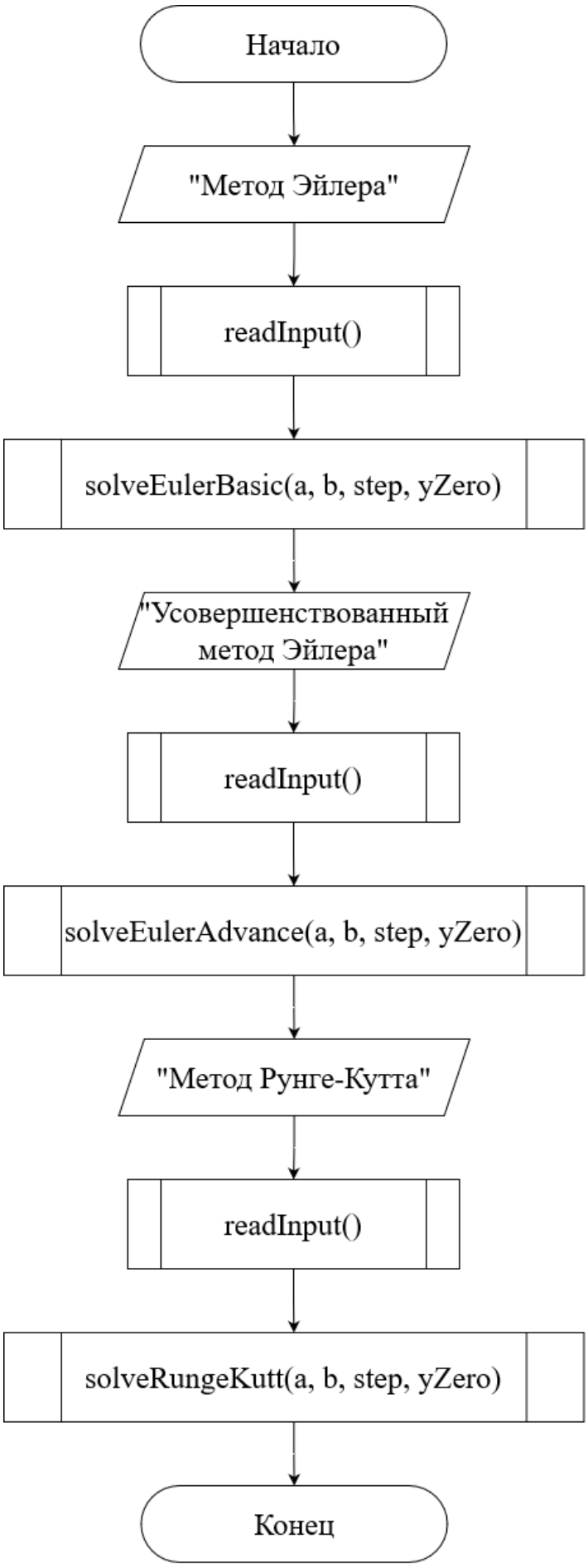
1) Открыть цикл по всем элементам списка значений
2) Напечатать пару значений X и Y, отформатированную до 4 знака после запятой

3) Закрыть цикл

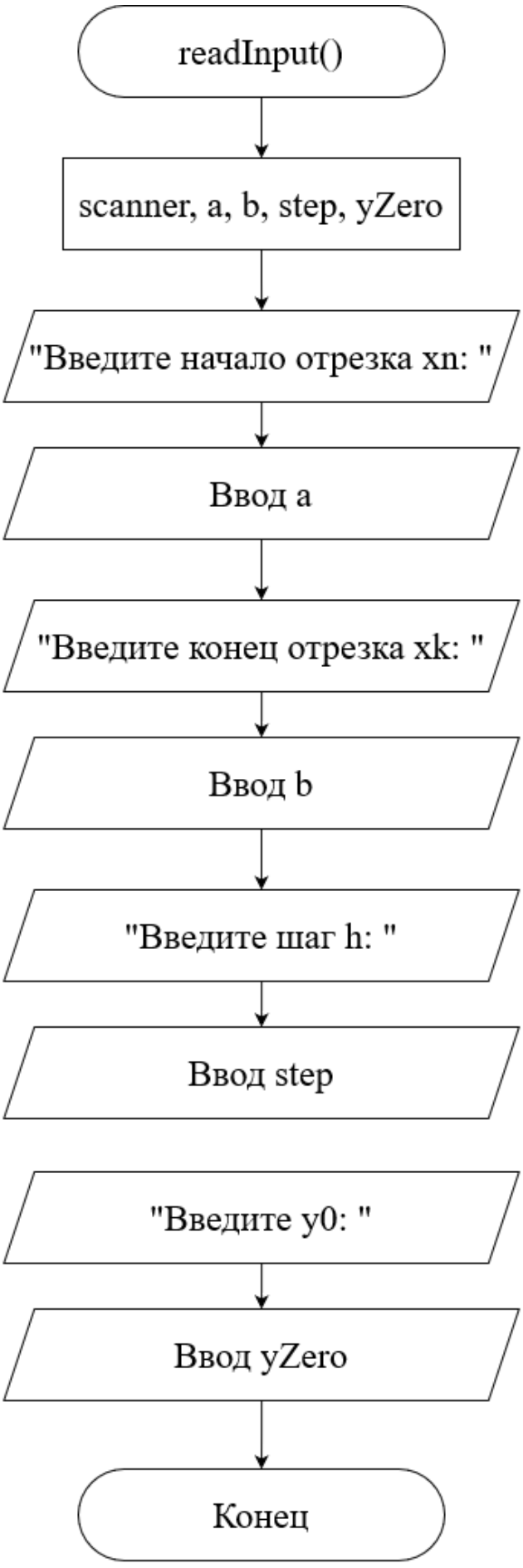
Алгоритмы методов решения не был принципиально изменен, за исключением того, что начальные значения передаются в функции напрямую, а не запрашиваются у пользователя индивидуально для каждого прохода. Также результаты работы выводятся после окончания расчетов. В связи с этим словесное описание опущено, блок схемы приведены ниже.

Блок-схемы функций программы

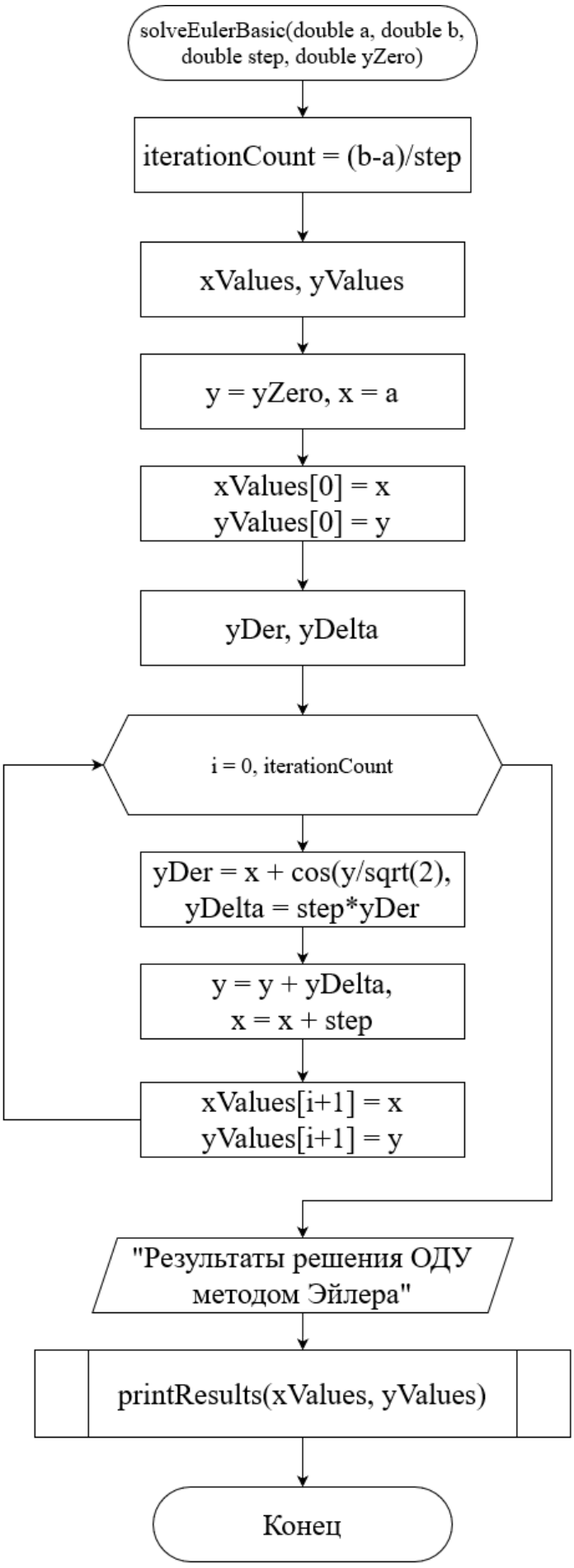
Основная функция


Начало
"Метод Эйлера"
readInput()
solveEulerBasic(a, b, step, yZero)
"Усовершенствованный метод Эйлера"
readInput()
solveEulerAdvance(a, b, step, yZero)
"Метод Рунге-Кутта"
readInput()
solveRungeKutt(a, b, step, yZero)
Конец

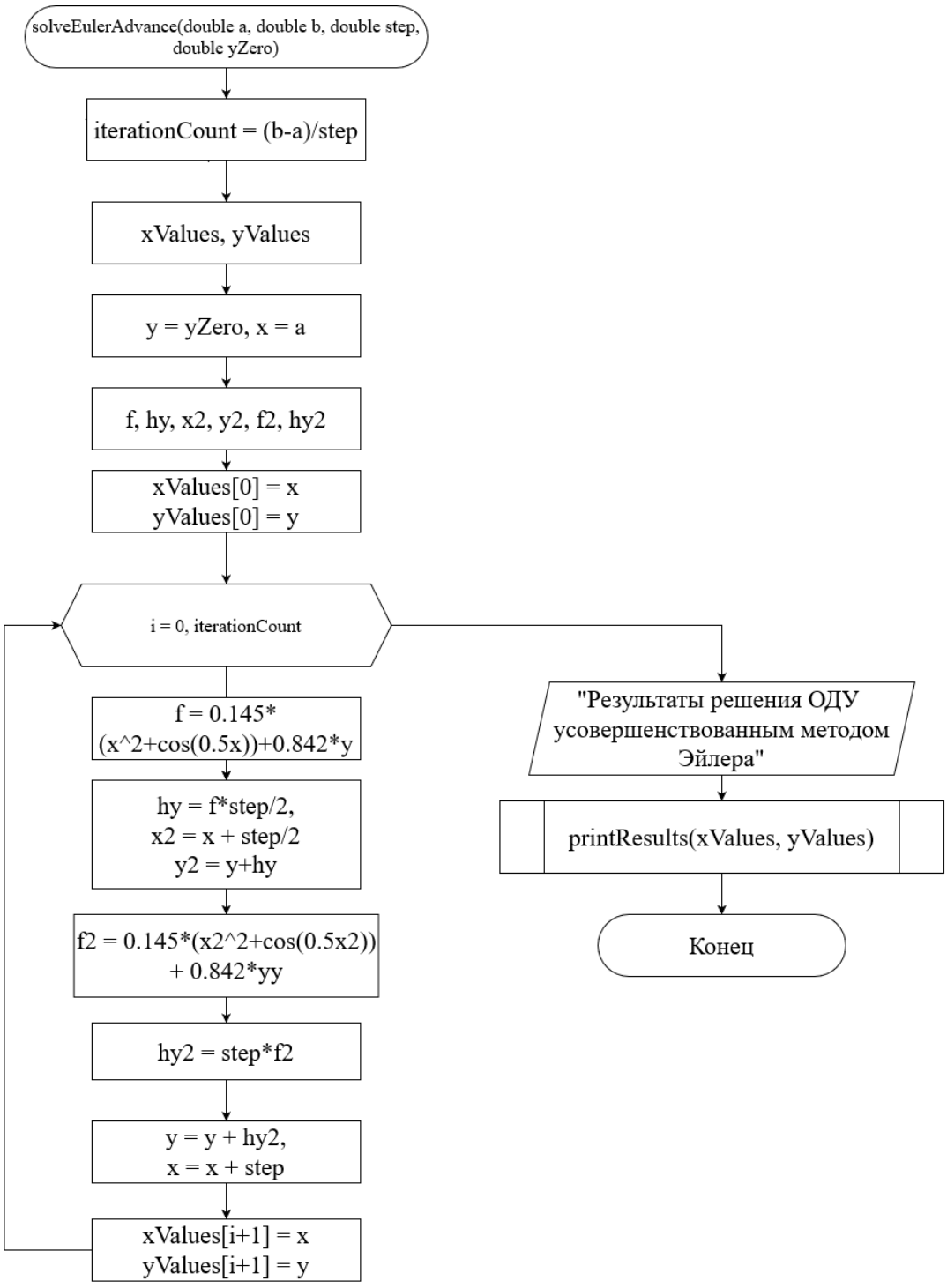
Функция считывания начальных значений


readInput()
scanner, a, b, step, yZero
"Введите начало отрезка xn: "
Ввод a
"Введите конец отрезка xk: "
Ввод b
"Введите шаг h: "
Ввод step
"Введите y0: "
Ввод yZero
Конец

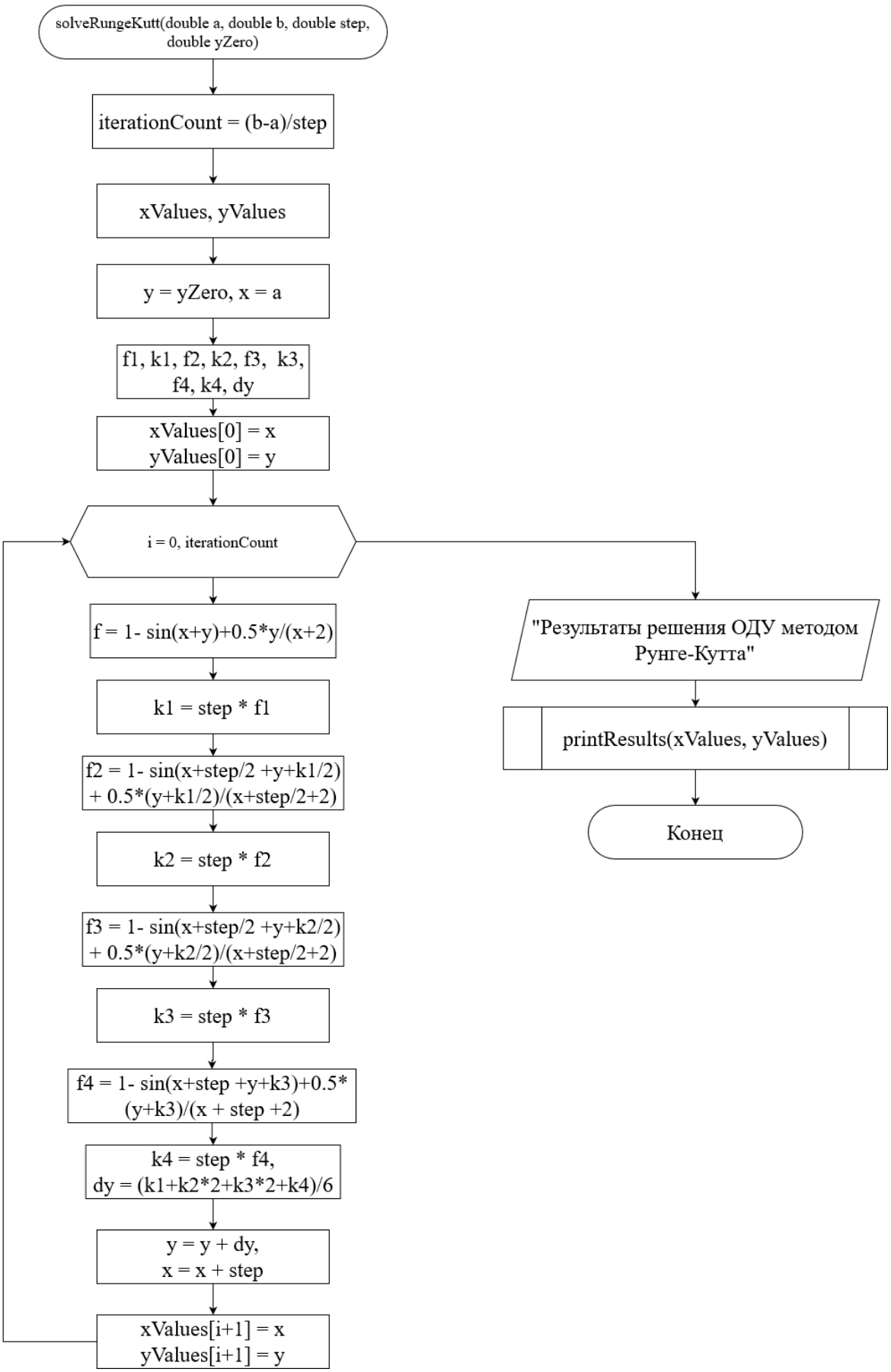
Функция решения уравнения методом Эйлера


solveEulerBasic(double a, double b, double step, double yZero)
iterationCount = (b-a)/step
xValues, yValues
y = yZero, x = a
xValues[0] = x
yValues[0] = y
yDer, yDelta
i = 0, iterationCount
yDer = x + cos(y/sqrt(2),
yDelta = step*yDer
y = y + yDelta,
x = x + step
xValues[i+1] = x
yValues[i+1] = y
"Результаты решения ОДУ методом Эйлера"
printResults(xValues, yValues)
Конец

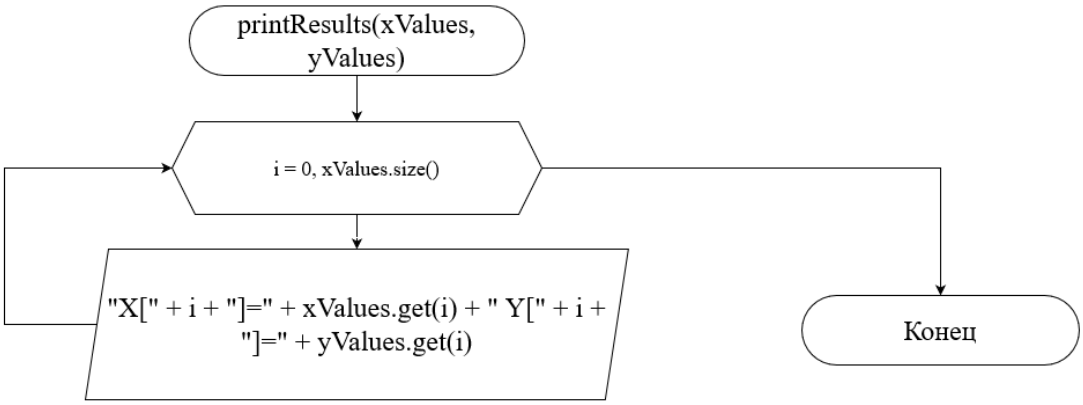
Функция решения уравнения усовершенствованным методом Эйлера



Функция решения уравнения методом Рунге-Кутта


solveRungeKutt(double a, double b, double step, double yZero)
iterationCount = (b-a)/step
xValues, yValues
y = yZero, x = a
f1, k1, f2, k2, f3, k3, f4, k4, dy
xValues[0] = x
yValues[0] = y
i = 0, iterationCount
f = 1- sin(x+y)+0.5*y/(x+2)
k1 = step * f1
f2 = 1- sin(x+step/2 +y+k1/2) + 0.5*(y+k1/2)/(x+step/2+2)
k2 = step * f2
f3 = 1- sin(x+step/2 +y+k2/2) + 0.5*(y+k2/2)/(x+step/2+2)
k3 = step * f3
f4 = 1- sin(x+step +y+k3)+0.5* (y+k3)/(x + step +2)
k4 = step * f4, dy = (k1+k2*2+k3*2+k4)/6
y = y + dy, x = x + step
xValues[i+1] = x
yValues[i+1] = y
"Результаты решения ОДУ методом Рунге-Кутта"
printResults(xValues, yValues)
Конец

Функция печати результатов



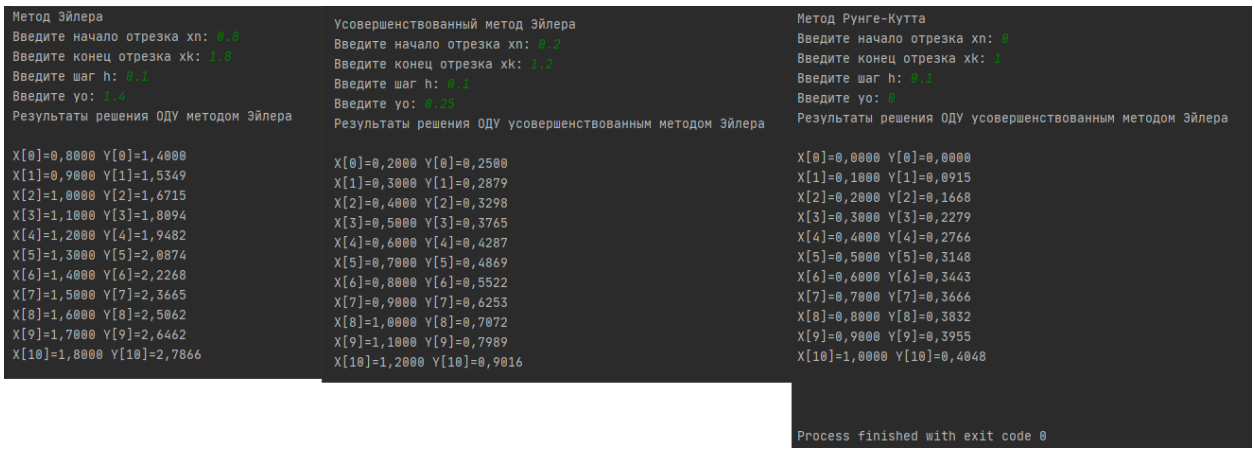
Текст программы

```
public class Main {

    public static void main(String[] args) {
        Solver s = new Solver();
        s.execute();
    }

}
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

import static java.lang.Math.*;

public class Solver {

    private double a, b, step, yZero;
    NumberFormat formatter = new DecimalFormat("#0.0000");

    public Solver() {

    }

    public void execute() {


        System.out.println("Метод Эйлера");
        readInput();
        solveEulerBasic(a, b, step, yZero);

        System.out.println("Усовершенствованный метод Эйлера");
        readInput();
        solveEulerAdvance(a, b, step, yZero);

        System.out.println("Метод Рунге-Кутта");
```

```
        readInput();
        solveRungeKutt(a, b, step, yZero);

    }

    private void readInput() {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Введите начало отрезка xn: ");
        a = Double.parseDouble(scanner.nextLine());
        System.out.print("Введите конец отрезка xk: ");
        b = Double.parseDouble(scanner.nextLine());
        System.out.print("Введите шаг h: ");
        step = Double.parseDouble(scanner.nextLine());
        System.out.print("Введите yo: ");
        yZero = Double.parseDouble(scanner.nextLine());
    }

    private void printResults(List<Double> xValues, List<Double> yValues) {

        for(int i = 0; i < xValues.size(); i++) {
            System.out.println("X[" + i + "]=" + formatter.format(xValues.get(i)) + " Y[" + i + "]=" +
formatter.format(yValues.get(i)));
        }
        System.out.println("\n");
    }

    private void solveEulerBasic(double a, double b, double step, double yZero) {
        int iterationCount =  (int)(ceil(b-a)/step);
        List<Double> xValues = new ArrayList<>();
        List<Double> yValues = new ArrayList<>();
        double y = yZero;
        double x = a;
        xValues.add(x);
        yValues.add(y);
        double yDer, yDelta;
        for(int i = 0; i < iterationCount; i++) {
            yDer = x + cos(y/sqrt(2));
            yDelta = step*yDer;
            y += yDelta;
            x += step;
            xValues.add(x);
            yValues.add(y);
        }
        System.out.println("Результаты решения ОДУ методом Эйлера\n");
        printResults(xValues, yValues);
    }

    private void solveEulerAdvance(double a, double b, double step, double yZero) {
        int iterationCount =  (int)(ceil(b-a)/step);
        List<Double> xValues = new ArrayList<>();
        List<Double> yValues = new ArrayList<>();
        double y = yZero;
        double x = a;
        double f, hy, x2, y2, f2, hy2;
        xValues.add(x);
        yValues.add(y);
        for(int i = 0; i < iterationCount; i++) {
            f = euAdvFunc(x, y);
            hy = f*step/2;
```

```
            x2=x+step/2;
            y2=y+hy;
            f2 = euAdvFunc(x2, y2);
            hy2 = step*f2;

            y += hy2;
            x += step;
            xValues.add(x);
            yValues.add(y);
        }
        System.out.println("Результаты решения ОДУ усовершенствованным методом Эйлера\n");
        printResults(xValues, yValues);
    }

    private double euAdvFunc(double x, double y) {
        return 0.145*(pow(x, 2) + cos(0.5*x))+0.842*y;
    }

    private void solveRungeKutt(double a, double b, double step, double yZero) {
        int iterationCount =  (int)(ceil(b-a)/step);
        List<Double> xValues = new ArrayList<>();
        List<Double> yValues = new ArrayList<>();
        double y = yZero;
        double x = a;
        double f1, k1, f2, k2, f3, k3, f4, k4, dy;
        xValues.add(x);
        yValues.add(y);
        for(int i = 0; i < iterationCount; i++) {
            f1 = rungeKuttFunc(x, y);
            k1 = step * f1;

            f2 = rungeKuttFunc(x+step/2, y+k1/2);
            k2 = step * f2;

            f3 = rungeKuttFunc(x+step/2, y+k2/2);
            k3 = step * f3;

            f4 = rungeKuttFunc(x+step, y+k3);
            k4 = step * f4;
            dy = (k1+k2*2+k3*2+k4)/6;

            y += dy;
            x += step;
            xValues.add(x);
            yValues.add(y);
        }
        System.out.println("Результаты решения ОДУ усовершенствованным методом Эйлера \n");
        printResults(xValues, yValues);
    }

    private double rungeKuttFunc(double x, double y) {
        return 1-sin(x+y)+0.5*y/(x+2);
    }

}
```

Скрины работы



Вывод

В ходе работы были изучены три численных метода решения дифференциальных уравнений первого порядка. Они были реализованы на языке программирования Java.