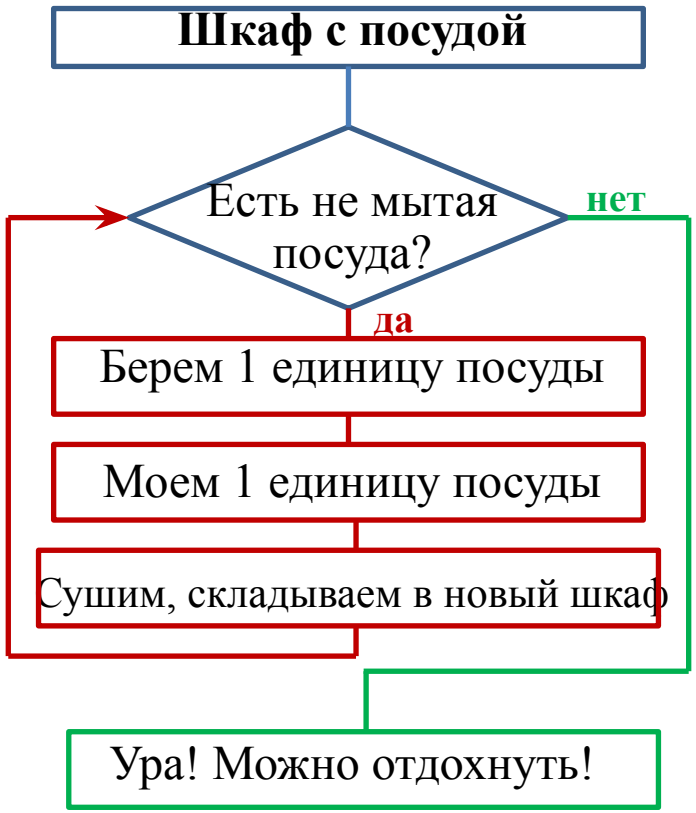
Информатика
Лекция 6
Программирование:
Основы языка программирования JAVA
Циклы

Доцент каф. ВММБ, к.т.н. Каменских Анна Александровна
Адрес кафедры ВММБ:108 к. Г (ул. Пр. Поздеева),
Тел. кафедры ВММБ: +7(342) 239-15-64
Электронная почта преподавателя: anna_kamenskih@mail.ru

Циклы (итерационные процедуры)

Цикл с неизвестным количеством операций:

У бабушки есть шкаф с посудой, которую нужно перемыть (сколько посуды в шкафу не известно даже бабушке) и переложить в новый шкаф.



В данном случае количество операций не

Цикл с известным количеством операций:

При переезде было собрано 98 коробок, которые нужно погрузить и перевезти в новый дом.

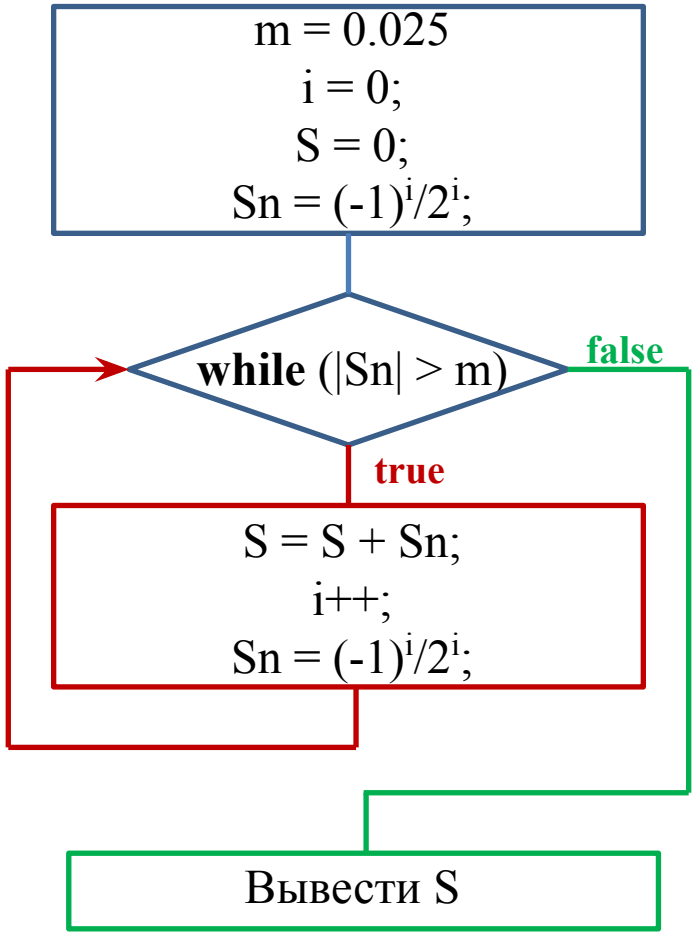
В данном случае известно сколько раз нужно выполнить операции!

Циклы

Постановка задачи:

Вычислить сумму членов ряда, абсолютное значение которых больше чем число m. Ряд:

$$S = 1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{8} + \ldots + (-1)^n \frac{1}{2^n}$$

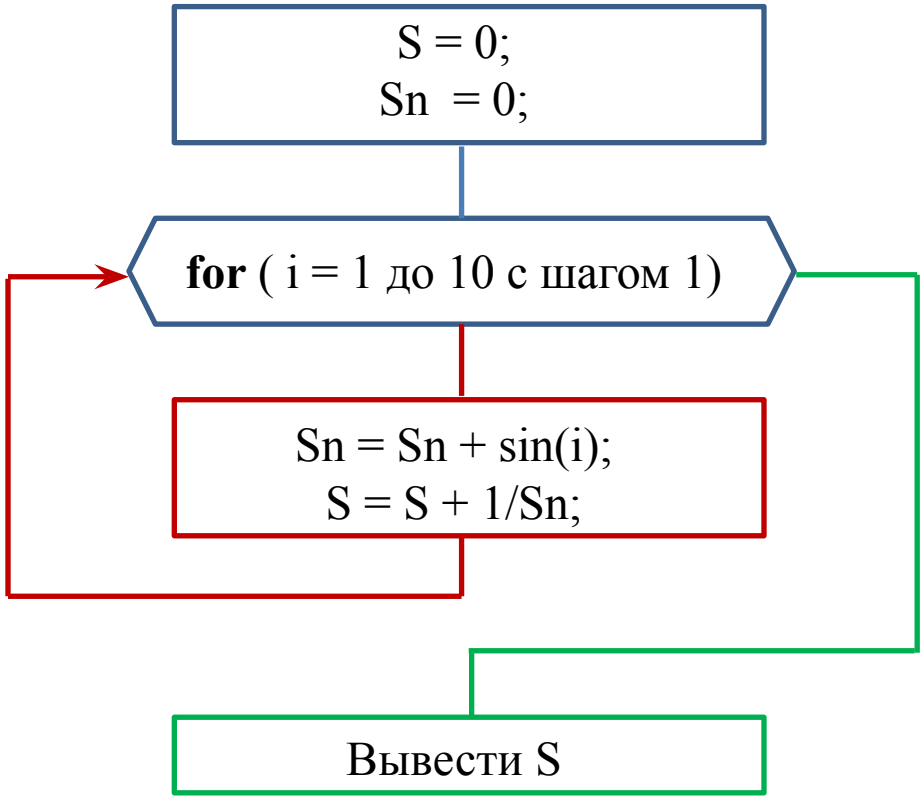


Постановка задачи:

Вычислить сумму первых 10 членов ряда.

Ряд:

$$S = \frac{1}{\sin(1)} + \frac{1}{\sin(1)+\sin(2)} + \ldots + \frac{1}{\sin(1)+\sin(2)+\ldots+\sin(n)}$$



Цикл While

Условие остановки итерационной процедуры

```
while (логическое выражение) {
    Оператор 1;
    Оператор 2;
}
```

Тело цикла

Постановка задачи:

Вычислить сумму членов ряда, абсолютное значение которых больше чем число m. Ряд:

$$S = 1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{8} + \ldots + (-1)^n \frac{1}{2^n}$$

Решение:

Если m = 0,025

$$S = 1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{8} + \frac{1}{16} - \frac{1}{32} + \frac{1}{64} - \ldots.$$

```
import static java.lang.System.out;
import java.util.Scanner;
public class CiclWhile {
  public static void main(String args[]) {
    Scanner scan = new Scanner(System.in);
    out.println("Введите m");
    double m = scan.nextDouble();
    int i = 0;
    double S = 0;
    double Sn = Math.pow(-1, i)/Math.pow(2, i);
    while (Math.abs(Sn) > m) {
      S = S + Sn;
      i++;
      Sn = Math.pow(-1, i)/Math.pow(2, i);
    }
    out.printf("Сумма %d членов ряда S = %f", i,
S);
  }
}
```

```
Введите m
0,025
Сумма 6 членов ряда S = 0,656250
```

$$S = 1 - 0,5 + 0,25 - 0,125 + 0,0625 - 0,03125 + 0,015625 - \ldots. = 0,65625$$

Цикл While

Цикл с неизвестным количеством операций:

Мама пытается накормить ребенка блинами. С каждой итерацией мама сокращает порцию в два раза. Посчитать сколько ребенок съест блинов до того, как наестся.

```
import java.util.Scanner;
public class Mama {
public static void main(String args[]) {
  Scanner scan = new Scanner(System.in);
  double Summa_blinov = 0;

  System.out.println("Введите  размер порции блинов в штуках");
  double start_porciy_blinov=scan.nextDouble();

  System.out.print("Введите  1 - если ребенок хочет есть, ");
  System.out.print("0  - если ребенок не хочет  есть \n ");
  int hochu_blinov=scan.nextInt();

  if (hochu_blinov<0 || hochu_blinov>1) {
    while(hochu_blinov<0 || hochu_blinov>1) {
      System.out.println("Вы  ввели информцию в неправильном формате");
      System.out.print("Введите  1 - если ребенок хочет  есть, ");
      System.out.print("0  - если ребенок не хочет  есть \n ");
      hochu_blinov=scan.nextInt();
    }
}
```

Ввод данных хочет ли ребенок есть

Проверка правильно ли пользователь ввел данные и цикл до того момента пока пользователь не введет данные верно ему будет предложено ввести данные.

Продолжение

Цикл While

Цикл с неизвестным количеством операций:

Мама пытается накормить ребенка блинами. С каждой итерацией мама сокращает порцию в два раза. Посчитать сколько ребенок съест блинов до того, как наестся.

```
int sch = 1;   // счетчик чтобы понять сколько итераций потребовалось, чтобы накормить чадо
if (hochu_blinov==1) {   // если надо кормить ребенка то параметр равен 1 и ребенка надо кормить
  while(hochu_blinov==1) {  // пока параметр равен 1  мы выполняем тело цикла и кормим чадо

    if (sch>1)  // если  мы даем есть ребенку более 1 раза, тогда порция с каждой итерацией уменьшается в 2 раза
       start_porciy_blinov  /=2;

    Summa_blinov += start_porciy_blinov;  // подсчет сколько всего слопал ребенок
    sch++;

    System.out.print("Введите  1 - если ребенок хочет есть, ");
    System.out.print("0  - если ребенок не хочет есть \n ");
    hochu_blinov=scan.nextInt();
    if (hochu_blinov<0 || hochu_blinov>1) {
      while(hochu_blinov<0 || hochu_blinov>1) {
        System.out.println("Вы  ввели информцию в неправильном формате");
        System.out.print("Введите  1 - если ребенок хочет есть, ");
        System.out.print("0 - если ребенок не хочет есть \n ");
        hochu_blinov=scan.nextInt();
      }
    }
  }
  System.out.printf("Ребенок слопал %.15f блинов, до того как наелся \n",Summa_blinov);
  System.out.println("Потребовалось " + (sch-1) + " итераций до насыщения ребенка");
} else {
  System.out.printf("Ребенок не хочет есть, вы зря выбирали размер порции блинов");
}
}
}
```

Блок ввода данных о необходимости питания и проверка данных выполняется в программе 2 раз

Вывод сколько слопал ребенок, если мы его кормили и за сколько итераций

Вывод о том, что чадо мы не кормили

Сценарии работы программы

```
Введите размер порции блинов в штуках
5
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
0
Ребенок не хочет есть, вы зря выбирали размер порции блинов
```

```
Введите размер порции блинов в штуках
10
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
0
Ребенок слопал 19,687500000000000 блинов, до того как наелся
Потребовалось 6 итераций до насыщения ребенка
```

	A	B	C
1	*№ итерации*	*Порция блинов*	*Сумма слопаных блинов*
2	1	10	10
3	2	5	15
4	3	2,5	17,5
5	4	1,25	18,75
6	5	0,625	19,375000000000
7	6	0,3125	19,687500000000
8	7	0,15625	19,843750000000
9	8	0,078125	19,921875000000
10	9	0,0390625	19,960937500000
11	10	0,01953125	19,980468750000
12	11	0,009765625	19,990234375000
13	12	0,004882813	19,995117187500
14	13	0,002441406	19,997558593750
15	14	0,001220703	19,998779296875
16	15	0,000610352	19,999389648438
17	16	0,000305176	19,999694824219
18	17	0,000152588	19,999847412109
19	18	7,62939E-05	19,999923706055
20	19	3,8147E-05	19,999961853027
21	20	1,90735E-05	19,999980926514

Проверка показала, что программа работает верно!

Сценарий работы программы с неправильным вводом данных

```
Введите размер порции блинов в штуках
5
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
5
Вы ввели информцию в неправильном формате
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
-1
Вы ввели информцию в неправильном формате
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
3
Вы ввели информцию в неправильном формате
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
2
Вы ввели информцию в неправильном формате
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
1
Введите 1 - если ребенок хочет есть, 0 - если ребенок не хочет есть
0
Ребенок слопал 9,921875000000000 блинов, до того как наелся
Потребовалось 7 итераций до насыщения ребенка
```

	A	B	C
1	*№ итерации*	*Порция блинов*	*Сумма слопаных блинов*
2	1	5	5
3	2	2,5	7,5
4	3	1,25	8,75
5	4	0,625	9,375
6	5	0,3125	9,687500000000
7	6	0,15625	9,843750000000
8	7	0,078125	9,921875000000
9	8	0,0390625	9,960937500000
10	9	0,01953125	9,980468750000
11	10	0,009765625	9,990234375000
12	11	0,004882813	9,995117187500
13	12	0,002441406	9,997558593750
14	13	0,001220703	9,998779296875
15	14	0,000610352	9,999389648438
16	15	0,000305176	9,999694824219
17	16	0,000152588	9,999847412109
18	17	7,62939E-05	9,999923706055
19	18	3,8147E-05	9,999961853027
20	19	1,90735E-05	9,999980926514
21	20	9,53674E-06	9,999990463257

Проверка показала, что программа работает верно!

Программа с выносом повторяющегося блока в метод

```
import java.util.Scanner;
public class Mama {
public static Scanner scan = new Scanner(System.in);
public static void main(String args[]) {
 Scanner scan = new Scanner(System.in);
 double Summa_blinov = 0;
 System.out.println("Введите размер порции блинов в штуках");
 double start_porciy_blinov=scan.nextDouble();

 int hochu_blinov = kormit_yes_or_not();

 int sch = 1;
 if (hochu_blinov==1) {
    while(hochu_blinov==1) {
       if (sch>1)
          start_porciy_blinov /=2;
       Summa_blinov += start_porciy_blinov;
       sch++;
       hochu_blinov = kormit_yes_or_not();
    }
    System.out.printf("Ребенок слопал %.15f блинов, до того как наелся\n",Summa_blinov);
    System.out.println("Потребовалось " + (sch-1) + " итераций до насыщения ребенка");
 } else {
    System.out.printf("Ребенок не хочет есть, вы зря выбирали размер порции блинов");
 }
}
```

Программа с выносом повторяющегося блока в метод

Метод kormit_yes_or_not

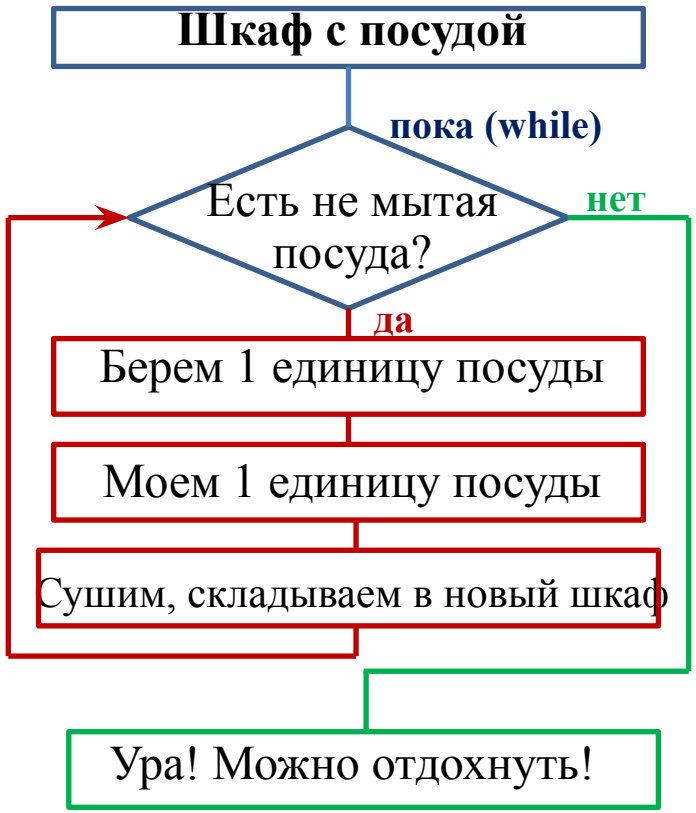
```
public static int kormit_yes_or_not(){
   System.out.println("Введите 1 - если ребенок хочет есть, "
                              + "0 - если ребенок не хочет есть");
   System.out.print("0 - если ребенок не хочет есть \n ");
   hochu_blinov=scan.nextInt();
   if (hochu_blinov<0 || hochu_blinov>1) {
      while(hochu_blinov<0 || hochu_blinov>1) {
          System.out.println("Вы ввели информцию в неправильном формате");
          System.out.print("Введите 1 - если ребенок хочет есть, "
                                 + "0 - если ребенок не хочет есть");
          hochu_blinov=scan.nextInt();
      }
   }
    return hochu_blinov;
}
}
```

Программа работает точно так же, только стала короче!

Циклы (итерационные процедуры)

Цикл с неизвестным количеством операций:

У бабушки есть шкаф с посудой, которую нужно перемыть (сколько посуды в шкафу не известно даже бабушке) и переложить в новый шкаф.



В данном случае количество операций не

Цикл с неизвестным количеством операций:

У бабушки есть шкаф с посудой, которую нужно перемыть (сколько посуды в шкафу не известно даже бабушке) и переложить в новый шкаф.

В данном случае количество операций так же не известно! Но тело цикла выполниться

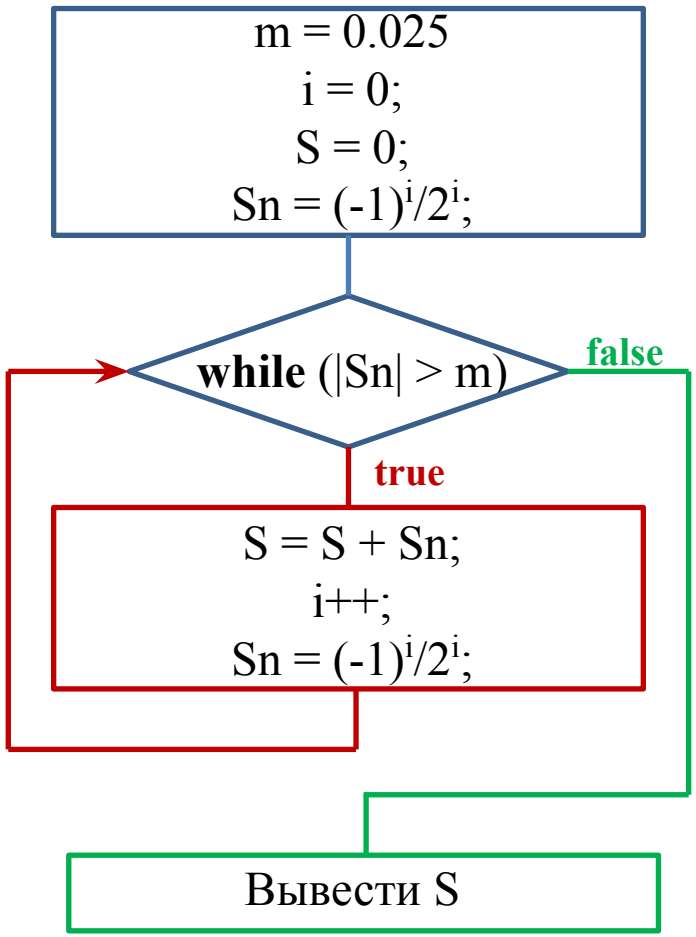
Циклы While и Do

Постановка задачи:

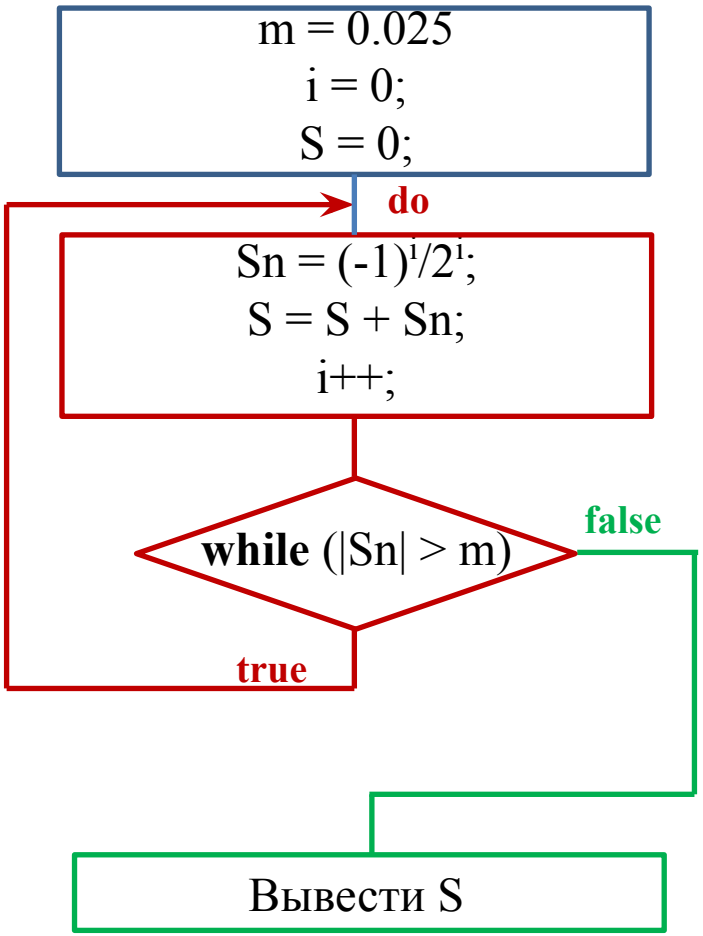
Вычислить сумму членов ряда, абсолютное значение которых больше чем число m. Ряд:

$$S = 1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{8} + \ldots + (-1)^n \frac{1}{2^n}$$

Цикл While



Цикл Do … While



Цикл Do (Do ... While)

```
do {
    Оператор 1;
    Оператор 2;
} while (логическое выражение);
```

Тело цикла

Условие остановки итерационной процедуры

Постановка задачи:

Вычислить сумму членов ряда, абсолютное значение которых больше чем число m. Ряд:

$$S = 1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{8} + \ldots + (-1)^n \frac{1}{2^n}$$

Решение:

Если m = 0,025

$$S = 1 - \frac{1}{2} + \frac{1}{4} - \frac{1}{8} + \frac{1}{16} - \frac{1}{32} + \frac{1}{64} - \ldots.$$

```
import static java.lang.System.out;
import java.util.Scanner;
public class CiclDoWhile {
   public static void main(String args[]) {
      Scanner scan = new Scanner(System.in);
      out.println("Введите m");
      double m = scan.nextDouble();
      int i = 0;
      double S = 0, Sn;
      do {
         Sn = Math.pow(-1, i)/Math.pow(2, i);
         if (Math.abs(Sn) > m)
               S = S + Sn;
         i++;
      } while (Math.abs(Sn) > m);
       out.printf("Сумма %d членов ряда S = %f", i-1,
S);
   }
}
```

```
CВведите m
 0,025
 Сумма 6 членов ряда S = 0,656250
```

$$S = 1 - 0,5 + 0,25 - 0,125 + 0,0625 - 0,03125 + 0,015625 - \ldots. = 0,65625$$

Цикл Do While

Цикл с неизвестным количеством операций:

Пользователь вводит целое число и хочет узнать сколько в нем цифр.

```
import java.util.Scanner;
public class Chislo {

public static void main(String args[]) {
   Scanner scan = new Scanner(System.in);
   System.out.println("Введите целочисленное число");
   int x = scan.nextInt();
   System.out.println("В числе " + kolichestvo_cifr(x) + " цифр" );
}
```

Основной метод класса

```
public static int kolichestvo_cifr(int x) {
    int sch = 0;
    do{
        x = x/10;
        sch++;
    } while(x!=0);
    return sch;
}
```

Метод класса для определения количества цифр в числе

```
}
```

Результаты работы программы

```
Введите целочисленное число
0
В числе 1 цифр
```

```
Введите целочисленное число
123456789
В числе 9 цифр
```

```
Введите целочисленное число
-123
В числе 3 цифр
```

```
Введите целочисленное число
11111111111
Exception in thread "main" java.util.InputMismatchException: For input string: "11111111111"
        at java.base/java.util.Scanner.nextInt(Scanner.java:2264)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
        at Chislo.main(Chislo.java:6)
```

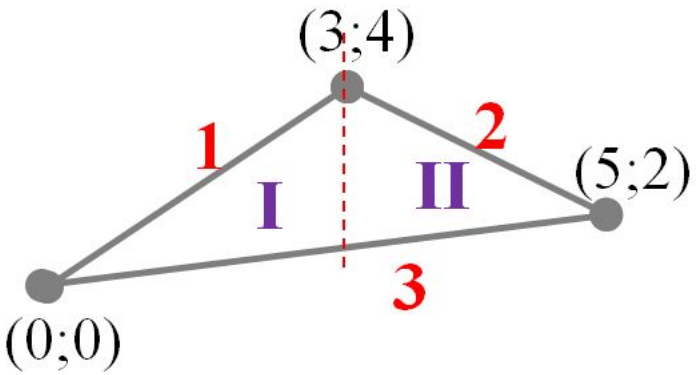
Цифр 11, а integer включает только 10 цифр по размерности

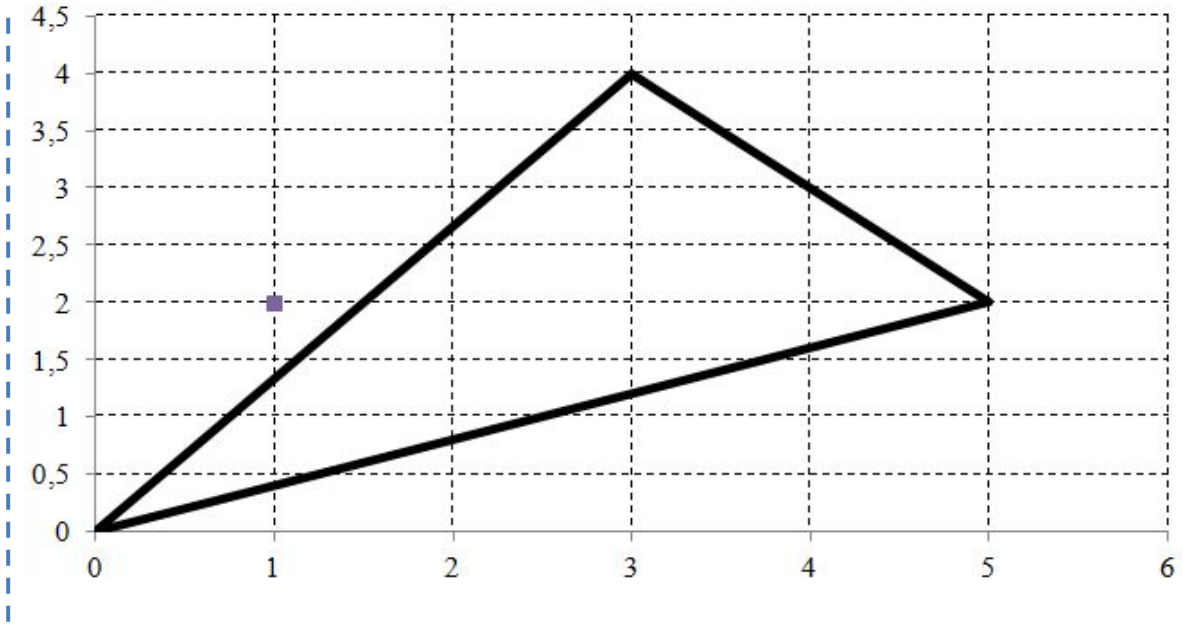
Для того чтобы снять ограничение на 10 цифр в числе достаточно переменную x сделать long, тогда поменяется всего 2 строчки кода:
Строчка 6 – long x = scan.nextLong();
Строчка 9 – public static int kolichestvo_cifr(long x) {

```
Введите целочисленное число
1234567890123456
В числе 16 цифр
```

```
Введите целочисленное число
-1234567890123
В числе 13 цифр
```

Проверка попадания точки в область в цикле





Математическая постановка задачи

$y_1 = 4/3\,x$ *Уравнение прямой 1*

$y_2 = -x + 7$ *Уравнение прямой 2*

$y_3 = 2/5\,x$ *Уравнение прямой 3*

Класс, в который входит проверка попадания точки в область

```
public class Tr_ik {
    public static boolean Treugolnik(double x, double y){
        if ((x>=0 & x<=3 & y<=4/3f*x & y>=2/5f*x)||(x>=3 & x<=5 & y<=-x+7 & y>=2/5f*x))
            return true;
        else
            return false;
    }
}
```

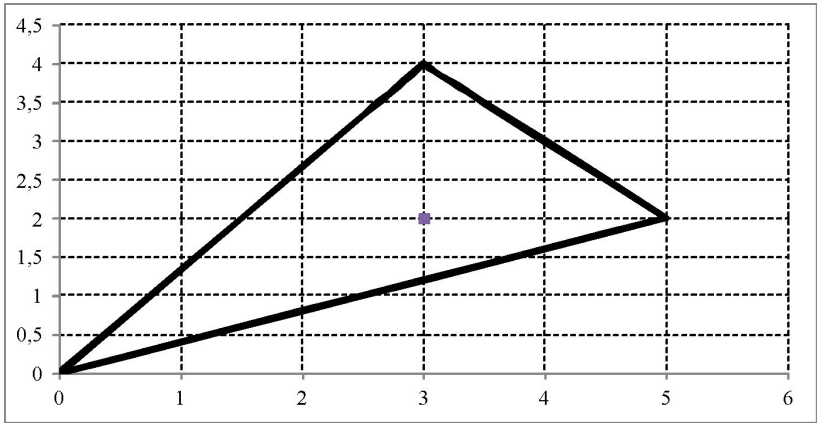
! В основной программе мы будем использовать метода данного класса (наследовать его)

Программа для пользователя с проверкой попадания точки

```
import java.util.Scanner;
public class Popadan_v_treugolnik extends Tr_ik { //Наследование методов класса Tr_ik

	public static void main(String args[]) {
		Scanner scan = new Scanner(System.in);

		System.out.println("Хотите ли вы проверить попала точка в треугольник?");
		boolean hochy = scan.nextBoolean();

		if (hochy) { // Если пользователь хочет проверить точку

			while (hochy) { // Цикл до того момента пока пользователь хочет проверять точки
				System.out.println("Введите координату точки x");
				double x = scan.nextDouble();
				System.out.println("Введите координату точки y");
				double y = scan.nextDouble();

				if (Treugolnik(x,y)) // Для поверки точки используется метод класса Tr_ik
					System.out.println("Точка попала в треугольник");
				else
					System.out.println("Точка не попала в треугольник");

				// Хочет ли пользователь дальше проверять точки
				System.out.println("Хотите ли вы проверить еще точку?");
				hochy = scan.nextBoolean();
				if (hochy)
					System.out.println("Проверяем еще одну точку");
				else
					System.out.println("Программа завершена");
			}

		} else // Если пользователь изначально не хочет проверять точки
			System.out.println("Очень жаль!");
	}
}
```

Сценарии работы программы

```
Хотите ли вы проверить попала точка в треугольник?
true
Введите координату точки x
0
Введите координату точки y
0
Точка попала в треугольник
Хотите ли вы проверить еще точку?
true
Проверяем еще одну точку
Введите координату точки x
3
Введите координату точки y
2
Точка попала в треугольник
Хотите ли вы проверить еще точку?
false
Программа завершена
```




```
Хотите ли вы проверить попала точка в треугольник?
false
Очень жаль!
```

Цикл For

```
for ( инициализация; условие; инкремент){
    Оператор 1;
    Оператор 2;
}
```
Тело цикла

- ***Инициализация*** – это присвоение счетчику начального значения. Всегда выполняется только один раз, когда программа начинает выполнять инструкцию for .
- ***Условие*** оценивается многократно перед каждой итерацией цикла .
- ***Инкремент*** счетчика выполняется многократно после каждой итерации цикла.

for (i=0; i <10;) – счетчик можно увеличивать в теле цикла.

for (; ;) – цикл будет выполнятся вечно.

В теле цикла можно использовать оператор break,
тогда программа немедленно выходит из цикла и

Постановка задачи

Вывести в цикле как изменяется «Значение счетчика».

Пример записи for

```
for ( int i = 1; i <= 10; i++) {
   out.print("Значение счетчика равно ");
   out.print(i);
   out.println(".");
}
```

Реализация аналогичного тела цикла с использованием while

```
i = 1;
while ( i <= 10) {
   out.print("Значение счетчика равно ");
   out.print(i);
   out.println(".");
   i++;
}
```

Пример использования цикла For

Постановка задачи:

Вычислить сумму первых 10 членов ряда.

Ряд: $S = \frac{1}{\sin(1)} + \frac{1}{\sin(1)+\sin(2)} + \ldots + \frac{1}{\sin(1)+\sin(2)+\ldots+\sin(n)}$

1) $1/\sin(1) = 1{,}1883951057781212$

2) $1/(\sin(1)+\sin(2)) = 0{,}5711777716316783$

3) $1/(\sin(1)+\sin(2)+\sin(3)) = 0{,}5285723986629383$

4) $1/(\sin(1)+\sin(2)+\sin(3)+\sin(4)) = 0{,}8809905739438656$

5) 5,676604423130382 5

6) -9,684869036403398

7) 1,8059253305182494

8) 0,6480499218978222

9) 0,51145414808554

10) 0,7086226193437877 → Сложив все 10 компонентов получим такой же ответ.

```
import static java.lang.System.out;
public class CiclFor {
  public static void main(String args[]) {
    double S = 0, Sn = 0;
    for (int i = 1; i <= 10; i++) {
      Sn = Sn + Math.sin(i);
      S = S + 1/Sn;
    }
     out.println("Сумма 10 членов ряда S = " +
S);
  }
}
```

```
Сумма 10 членов ряда S = 2.8349232565889864
```

Цикл For

Постановка задачи:

Пользователь вводит целое двоичное число, найти его значение в 10-ой системе счисления с использованием цикла for?

```
import java.util.Scanner;
public class Dvoichnoe_chislo{
    public static void main(String args[]) {
        Scanner scan = new Scanner(System.in);
        Chislo ch = new Chislo(); // для взаимосвязи с классом Chislo
        System.out.println("Введите целое число в 2-ой системе счисления");
        long x = scan.nextLong(), Chislo10 = 0;
        if(ch.proverka_na_2ost(x)) { //в скобках обращение к методу класса Chislo, который
                                     делает проверку является ли введенное число 2-ым
            int n = ch.kolichestvo_cifr(x); //обращение к методу возвращающему количество цифр класса
Chislo

            for(int i = 0; i < n; i++){
                Chislo10 += (int)(x%10*Math.pow(2, i));
                x = x/10;
            } // цикл for для перевода числа из 2-ой системы счисления в 10-ю

            System.out.println("Число в 10-ой системе счисления = " + Chislo10);
        } else {
            System.out.println("Введено число не в 2-ой системе счисления");
        }
    }
}
```

Измененный класс Chislo

```
public class Chislo {
```

```
public static int kolichestvo_cifr(long x) {
    int sch = 0;
    do{
        x = x/10;
        sch++;
    } while(x!=0);
    return sch;
}
```

Метод делит число, которое получил на 10, до того момента пока оно не станет равным 0. Т.е. не останется целой части. При этом подсчитывается сколько чисел входит в число, через счетчик. Метод возвращает количество цифр в числе.

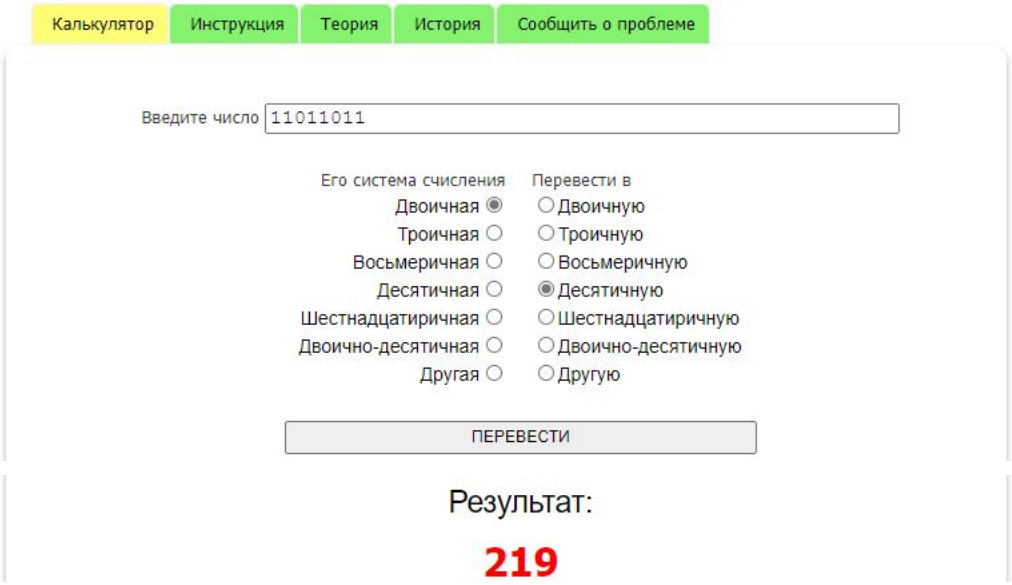
```
public static boolean proverka_na_2ost(long x) {
    int n = kolichestvo_cifr(x), sch = 0;
    for(int i = 1; i <= n; i++) {
        if(x%10 == 1 || x%10 == 0)
            sch++;
            x=x/10;
     }
     if (sch == n)
         return true;
     else
         return false;
 }
```

Метод обращается к 1 методу класса для организации цикла for. Затем в цикле for: 1) отделяется каждое число через остаток от деления на 10 и сравнивается с 0 и 1, при равенстве части числа 0 или 1 счетчик наращивается на 1; 2) число уменьшается на уже проверенную цифру. Пункты 1 и 2 выполняются пока мы не проверим все цифры числа. Затем если количество цифр числа совпало с числом числами равными 0 или 1, тогда число в двоичной системе счисления и метод возвращает ИСТИНА, иначе ЛОЖЬ.

```
}
```

Сценарии работы программы

```
Введите целое число в 2-ой системе счисления
11011011
Число в 10-ой системе счисления = 219
```




```
Введите целое число в 2-ой системе счисления
120101301
Введено число не в 2-ой системе счисления
```

Но можно сделать и другой вариант связи между классами Dvoichnoe_chislo и Chislo

НАСЛЕДОВАНИЕ

Тогда в заголовке класса Dvoichnoe_chislo надо добавить служебное слово extends и имя класса от которого мы наследуем методы Chislo

Другой вид класса Dvoichnoe_chislo

```
import java.util.Scanner;
public class Dvoichnoe_chislo extends Chislo {
   public static void main(String args[]) {
      Scanner scan = new Scanner(System.in);
      System.out.println("Введите целое число в 2-ой системе счисления");
      long x = scan.nextLong(), Chislo10 = 0;
      if(proverka_na_2ost(x)) { //в скобках обращение к методу класса Chislo, который
                                делает проверку является ли введенное число 2-ым
         int n = kolichestvo_cifr(x); //обращение к методу возвращающему количество цифр класса
Chislo

         for(int i = 0; i < n; i++){
            Chislo10 += (int)(x%10*Math.pow(2, i));
            x = x/10;
         } // цикл for для перевода числа из 2-ой системы счисления в 10-ю

         System.out.println("Число в 10-ой системе счисления = " + Chislo10);
      } else {
         System.out.println("Введено число не в 2-ой системе счисления");
      }
   }
}
```

Тогда к методам класса Chislo мы обращаемся просто как и обращались бы ко своим методам класса, так как мы их наследовали от класса Chislo!

Много кратное использование цикла For

Постановка задачи:

Вычислить значение выражения:

$$\sum_{i=1}^{8}\prod_{j=1}^{3}(j+i)$$

For i = 1 до 8 с шагом 1
For j = 1 до 3 с шагом 1
i = 1 → пробегаем весь цикл j
Sj = (1+1)*(2+1)*(3+1) = 24
вышел из j прибавил Sj к Si, приравнял Sj =1
i = 2 → пробегаем весь цикл j
Sj = (1+2)*(2+2)*(3+2) = 60
вышел из j прибавил Sj к Si, приравнял Sj =1…

i\j	1	2	3	Произведение
1	2	3	4	24
2	3	4	5	60
3	4	5	6	120
4	5	6	7	210
5	6	7	8	336
6	7	8	9	504
7	8	9	10	720
8	9	10	11	990
Сумма произведения				2964

```
import static java.lang.System.out;
public class ForFor {
   public static void main(String args[]) {
      double Si = 0, Sj = 1;
      for (int i = 1; i <= 8; i++) {
         for (int j = 1; j <= 3; j++) {
            Sj = Sj*(j+i);
         }
      Si = Si + Sj;
      Sj = 1;
     }
      out.println("Сумма произведения = " +
Si);
   }
}
```

```
Сумма произведения = 2964.0
```

Сочетание циклов For и While

Постановка задачи:

Вычислить значение выражения:

$$\sum_{i=1}^{10}\sum_{j=1}^{i}(2i+j)$$

For i = 1 до 10 с шагом 1

Пока j <= i тогда

i = 1 (при входе в цикл j = 1) → j <= 1

Sj = (2*1+1)=3 вышел из цикла Пока

прибавил Sj к Si, приравнял Sj = 0,

i = 2 (при входе в цикл j = 1) → j <= 1

Sj = (2*2+1)+(2*2+2) = 11 вышел из цикла Пока

прибавил Si к Si, приравнял Si = 0

```
import static java.lang.System.out;
public class CiclForWhile {
  public static void main(String args[]) {
    double Si = 0, Sj = 0;
    int j;
    for (int i = 1; i <= 10; i++) {
      j = 1;
      while (j<=i) {
        Sj = Sj + (2*i+j);
        j++;
      }
    Si = Si + Sj;
    Sj = 0;
    }
    out.println("Сумма сумм = " + Si);
  }
}
```

i/j	1	2	3	4	5	6	7	8	9	10	Сумма
1	3	0	0	0	0	0	0	0	0	0	3
2	5	6	0	0	0	0	0	0	0	0	11
3	7	8	9	0	0	0	0	0	0	0	24
4	9	10	11	12	0	0	0	0	0	0	42
5	11	12	13	14	15	0	0	0	0	0	65
6	13	14	15	16	17	18	0	0	0	0	93
7	15	16	17	18	19	20	21	0	0	0	126
8	17	18	19	20	21	22	23	24	0	0	164
9	19	20	21	22	23	24	25	26	27	0	207
10	21	22	23	24	25	26	27	28	29	30	255
									Сумма сумм		990

Ответ:

```
Сумма сумм = 990.0
```

Как сделать проверку задач с вычислениями типа ΣΣ,ΣΠ,ΠΣ,ΠΠ

Вычислить значение выражения:

$$\sum_{i=1}^{8}\prod_{j=1}^{3}(j+i)$$

	A	B	C	D	E	F
1						
2		i/j	1	2	3	Произведение
3		1	2	3	4	24
4		2	3	4	5	60
5		3	4	5	6	120
6		4	5	6	7	210
7		5	6	7	8	336
8		6	7	8	9	504
9		7	8	9	10	720
10		8	9	10	11	990
11					Сумма	2964

- В **C3** вводится формула **«=$B3+C$2»**, т.е. суммируются индексы *i* и *j*, которые заданны в столбце **B** и строчке **2** соответственно, формула растягивается на весь диапазон *i* и *j*;

-В ячейку **F3** вводится формула **«=ПРОИЗВЕД(C3:E3)»**, т.е. идет перемножение слагаемых при одинаковых значениях *i*, что соответствует заданной формулой, затем формула растягивается на весь диапазон *i*;

- В ячейка **F11** формируется итоговая сумма произведений при помощи формулы

Вычислить значение выражения:

$$\sum_{i=1}^{10}\sum_{j=1}^{i}(2i+j)$$

	A	B	C	D	E	F	G	H	I	J	K	L
1												
2		i/j	1	2	3	4	5	6	7	8	9	10
3		1	3	0	0	0	0	0	0	0	0	0
4		2	5	6	0	0	0	0	0	0	0	0
5		3	7	8	9	0	0	0	0	0	0	0
6		4	9	10	11	12	0	0	0	0	0	0
7		5	11	12	13	14	15	0	0	0	0	0
8		6	13	14	15	16	17	18	0	0	0	0
9		7	15	16	17	18	19	20	21	0	0	0
10		8	17	18	19	20	21	22	23	24	0	0
11		9	19	20	21	22	23	24	25	26	27	0
12		10	21	22	23	24	25	26	27	28	29	30
13											Итоговая сумма	990

-В **C3** вводится формула **«=ЕСЛИ (C$2<=$B3;2*$B3+C$2;0)»**, т.е. если номер *j* <= *i* , тогда находится сумма *2*i+j*, иначе 0, формула растягивается на весь диапазон *i* и *j*;

! При суммировании 0 задаются компоненты, которых не существует для данной задачи, при произведении компоненты которые не существуют для данной задачи задаются 1.

-В ячейка **L11** формируется итоговая сумма ссумм формулой **«=СУММ(C3:L12)»**.

! Внимательно следите за логикой математических операций чтобы избежать

Пример реализации задачи с использованием циклов

Постановка задачи: Ежемесячная стипендия студента составляет А руб., студент откладывает стипендию на покупку ноутбука, который стоит В руб. Стипендия индексируется на 6 % каждые три месяца, при этом стоимость ноутбука увеличивается раз в пол года на 10 %. Определить через сколько месяцев студент сможет купить ноутбук.

```
import java.util.Scanner;
  public class Cicl {
      public static void main(String args[]) {
      Scanner scn = new Scanner(System.in);
      System.out.println("Введите сумму стипендии, которую получае студент");
      double A = scn.nextDouble();
      System.out.println("Введите стоимость компьютера, на который студент копит");
      double B = scn.nextDouble();
      int i = 0;
      double S = 0;
      while (S<B) {
            if (i%3==0 & i!=0)
                  A = A+0.06*A;
            if (i%6==0 & i!=0)
                  B = B+0.1*B;
            S +=A;
            i++;
      }
      System.out.printf("Стуенту понадобиться %d месяцев копить на компьютер \n", i);
      System.out.printf("Текущий размер стипендии = %.4f руб. \n", A);
      System.out.printf("Текущая стоимость компьютера = %.4f руб.", B);
  }
}
```

Ответ

```
<terminated> Cicl [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\
Введите сумму стипендии, которую получае студент
2000
Введите стоимость компьютера, на который студент копит
20000
Стуенту понадобиться 11 месяцев копить на компьютер
Текущий размер стипендии = 2382,0320 руб.
Текущая стоимость компьютера = 22000,0000 руб.
```

Вычисление корня p-й степени в рамках итерационной процедуры

$y = \sqrt[p]{x}$ – функция, приближенное значение которой мы ищем

Итерационная процедура

$$y_{n+1} = \frac{1}{p}\left((p-1)y_n + \frac{x}{y_n^{p-1}}\right) \quad n = 0,1,2\ldots$$

$$y_0 < e^{\frac{\ln\left(x(p+1)\right)}{p}}$$

Начальное значение для итерационной процедуры

$$\left|y_{n+1} - y_n\right| \le \varepsilon$$

Остановка итерационной процедуры

Метод для отыскания корней нелинейных уравнений

$$f(x)=0$$

f(x) – нелинейное уравнение, которое от -∞ до + ∞, может обладать большим количеством корней.

Метод

Аналитические

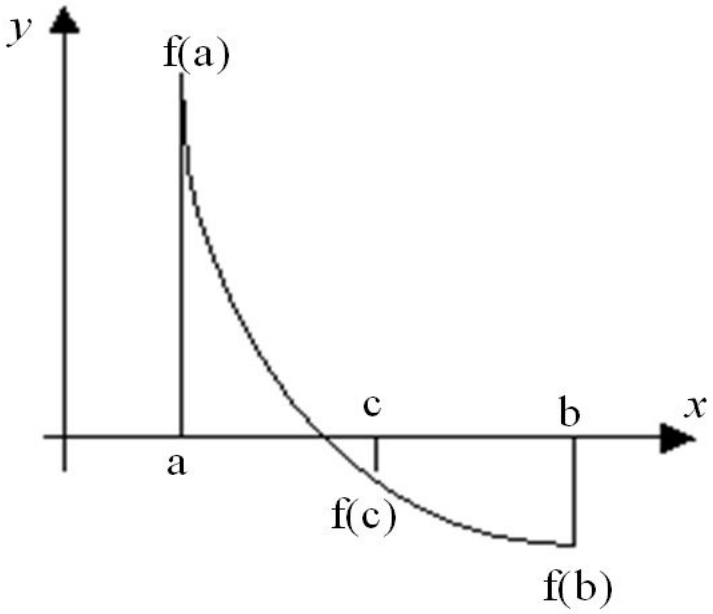
Графические

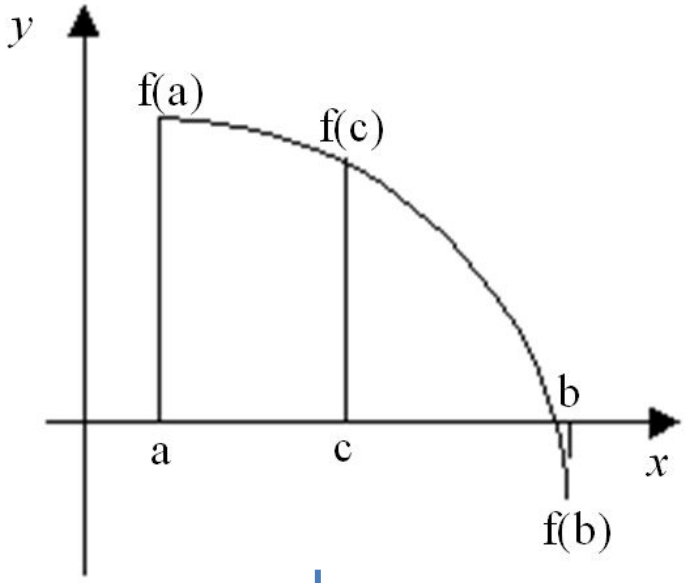
С использованием прикладных программ

Численные

Половинного деления
Простых итераций
Ньютона
Хорд
и т.д.

Метод половинного деления





Половина отрезка

$$c = \frac{a+b}{2}$$

Остановка итерационной процедуры

$$|b-a| < \varepsilon$$

Разделим отрезок **[a,b]** пополам точкой **c**. Если f(c) ≠ 0 (что практически наиболее вероятно), то возможны два случая:

- f(x) меняет знак на **[a,c]**, т.е. **f(a)*f(c) < 0**. Тогда отрезок **[a,b]** сокращается в два раза, это значит, что мы отсекаем отрезок **[c,b]** или **b=c**.

- f(x) меняет знак на **[c,b]**, т.е. **f(c)*f(b) < 0**. Тогда отрезок **[a,b]** сокращается в два раза, это значит, что мы отсекаем отрезок **[a,c]** или **a=c**.

ε – точность вычисления

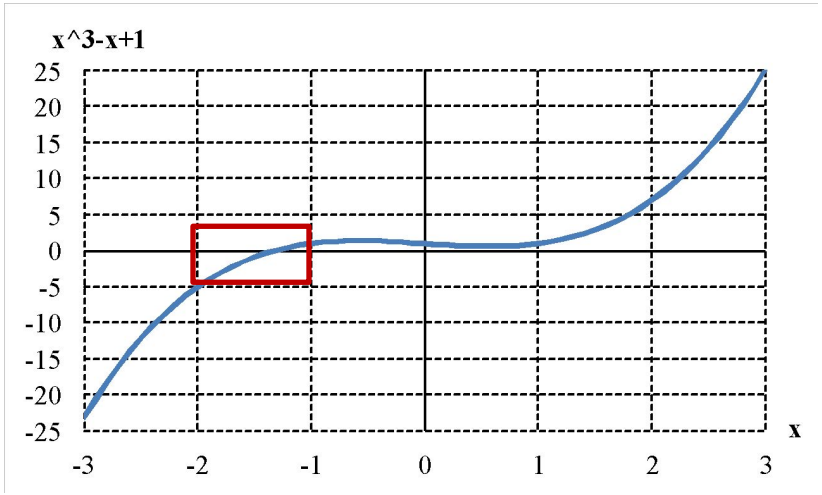
Условие проверки есть ли корень на отрезке, который ввел пользователь

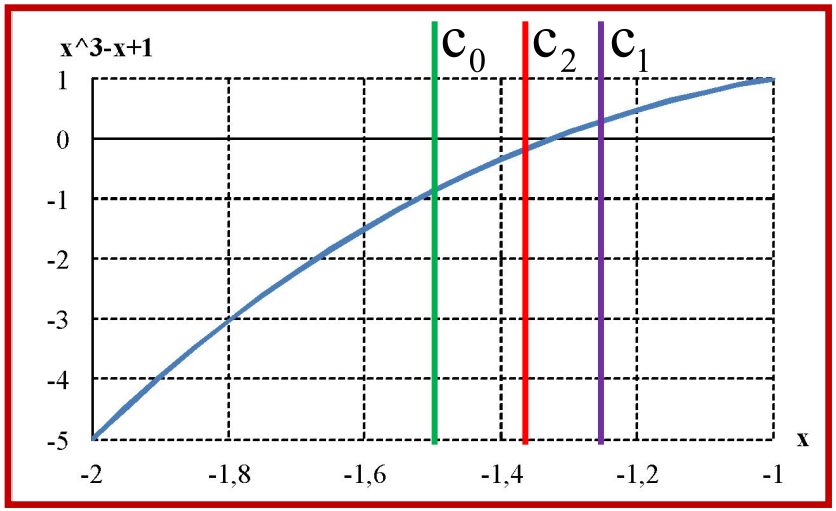
$$f(a) * f(b) < 0$$

Условие проверки есть ли корень на половине отрезка (необходимо для реализации метода)

$$f(a) * f(\ \ ) < 0$$

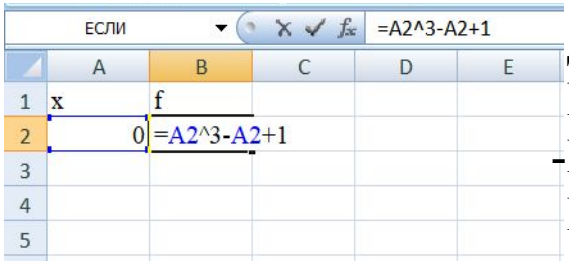
Найти корень уравнения $x^3 - x + 1 = 0$ методом половинного деления с точностью ε=0,01 и ε=0,0001 .



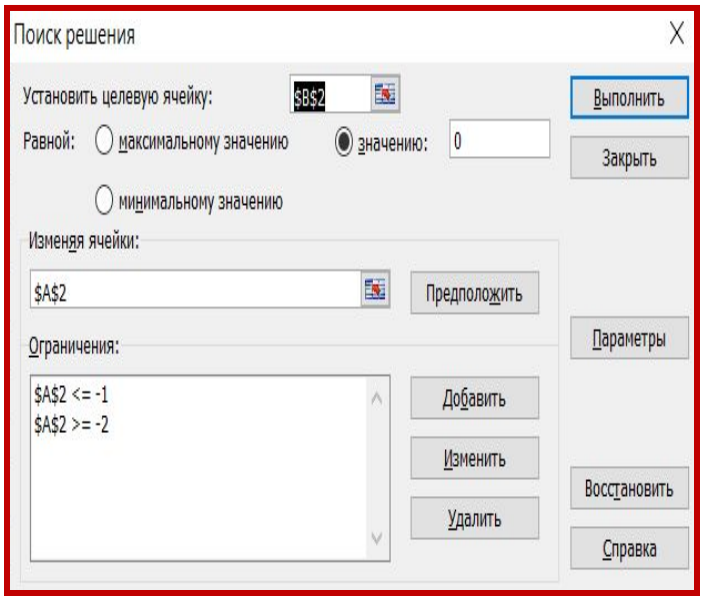


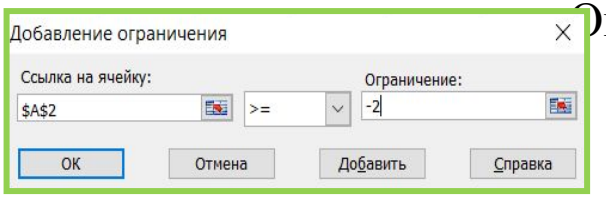
№ итер	a	b	c	f(a)	f(c)	f(a)*f(c)	abs(b-a)	
0	-2,00000000	-1,00000000	-1,50000000	-5,00000000	-0,87500000	4,37500000	1	
1	-1,50000000	-1,00000000	-1,25000000	-0,87500000	0,29687500	-0,25976563	0,5	
2	-1,50000000	-1,25000000	-1,37500000	-0,87500000	-0,22460938	0,19653320	0,25	
3	-1,37500000	-1,25000000	-1,31250000	-0,22460938	0,05151367	-0,01157045	0,125	
4	-1,37500000	-1,31250000	-1,34375000	-0,22460938	-0,08261108	0,01855522	0,0625	
5	-1,34375000	-1,31250000	-1,32812500	-0,08261108	-0,01457596	0,00120414	0,03125	
6	-1,32812500	-1,31250000	-1,32031250	-0,01457596	0,01871061	-0,00027273	0,015625	
7	-1,32812500	-1,32031250	-1,32421875	-0,01457596	0,00212795	-0,00003102	0,0078125	**ε<0,01 x = -1,32421875**
8	-1,32812500	-1,32421875	-1,32617188	-0,01457596	-0,00620883	0,00009050	0,00390625	
9	-1,32617188	-1,32421875	-1,32519531	-0,00620883	-0,00203665	0,00001265	0,001953125	
10	-1,32519531	-1,32421875	-1,32470703	-0,00203665	0,00004659	-0,00000009	0,000976563	
11	-1,32519531	-1,32470703	-1,32495117	-0,00203665	-0,00099479	0,00000203	0,000488281	
12	-1,32495117	-1,32470703	-1,32482910	-0,00099479	-0,00047404	0,00000047	0,000244141	
13	-1,32482910	-1,32470703	-1,32476807	-0,00047404	-0,00021371	0,00000010	0,00012207	
14	-1,32476807	-1,32470703	-1,32473755	-0,00021371	-0,00008355	0,00000002	6,10352E-05	**ε<0,0001 x = -1,32473755**
15	-1,32473755	-1,32470703	-1,32472229	-0,00008355	-0,00001848	0,00000000	3,05176E-05	
16	-1,32472229	-1,32470703	-1,32471466	-0,00001848	0,00001406	0,00000000	1,52588E-05	
17	-1,32472229	-1,32471466	-1,32471848	-0,00001848	-0,00000221	0,00000000	7,62939E-06	
18	-1,32471848	-1,32471466	-1,32471657	-0,00000221	0,00000592	0,00000000	3,8147E-06	
19	-1,32471848	-1,32471657	-1,32471752	-0,00000221	0,00000186	0,00000000	1,90735E-06	
20	-1,32471848	-1,32471752	-1,32471800	-0,00000221	-0,00000018	0,00000000	9,53674E-07	

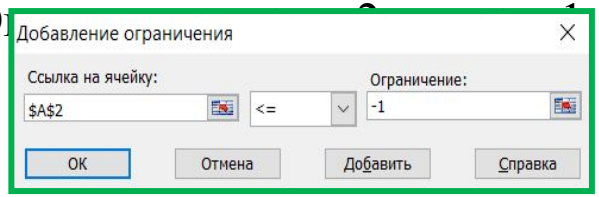
Проверка корня уравнения с помощью «Поиск решения»



Данные -> Поиск решения
Настраиваем поиск решения:
-Целевая ячейка это функция B2 (f)
- Равной: 0
-Изменяя ячейки A2 (x)







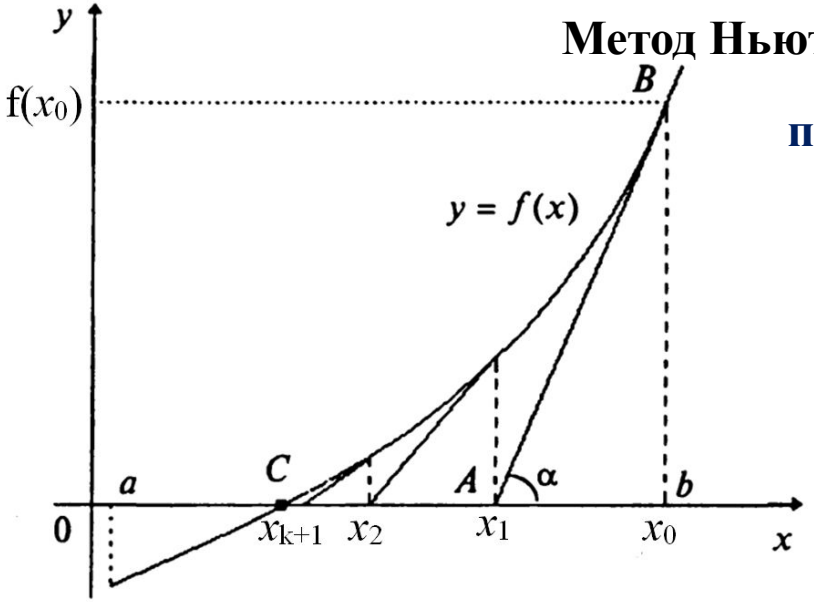
В итоге получаем корень уравнения на заданном отрезке

	A	B
1	x	f
2	-1,324717957245	0,000000000000

При этом решение задачи в MS Excel так же получено с погрешностью, но мы можем настроить «Параметры» таким образом чтобы она была минимальна.

Количество итераций метода половинного деления	Корень метод половинного деления	Корень, полученный поиском решений	Погрешность %
8 (ε<0,01)	**-1,324218750000**	**-1,324717957245**	**0,037684039989**
15 (ε<0,0001)	**-1,324737548828**	**-1,324717957245**	**0,001478924874**
21 (ε<0,0000001)	**-1,324717998505**	**-1,324717957245**	**0,000003114617**

Метод Ньютона (метод касательной)



Необходимо чтобы производная функции не равнялась 0 на отрезке

$$f'(x) \neq 0$$

Нулевое приближение корня, чаще всего берут равным началу отрезка

$$x_0 = a$$

Первое приближение корня находят по формуле

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Остановка итерационной процедуры

$$\left|x_{k+1} - x_k\right| < \varepsilon$$

$$\frac{f(x_0) - 0}{x_0 - x_1} = f'(x_0) \equiv \operatorname{tg}(\alpha)$$

Метод Ньютона (метод касательных) является одним из наиболее популярных численных методов. Он быстро сходится (имеет квадратичную сходимость). Однако этот метод эффективен при весьма жестких ограничениях на характер функции f(x).

Геометрическая интерпретация метода Ньютона состоит в следующем. Задается начальное приближение x_0. Далее проводится касательная к кривой y = f(x) в точке x_0, т.е. кривая заменяется прямой линией. В качестве следующего приближения выбирается точка пересечения этой касательной с осью абсцисс. Процесс построения касательных и нахождения точек пересечения с осью абсцисс повторяется до тех пор, пока приращение не станет меньше заданной величины ε.

В общем виде k приближение корня находят по формуле

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

$$k = 0,1,2,...$$

Решение задачи методом Ньютона

$$f(x) = x^3 - x + 1$$

Найти корень уравнения $x^3 - x + 1 = 0$ методом Ньютона.

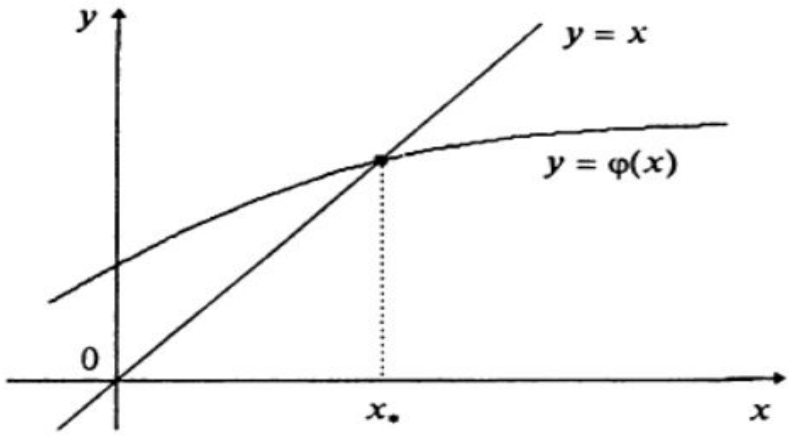
$$f'(x) = 3x^2 - 1$$

№ итер	xk	f(xk)	dif(f(xk)	xk+1	abs((xk+1)-(xk))
0	-2	-5	11	-1,545454545	0,454545455
1	-1,5454545454545500	-1,1457550713749100	6,1652892561983500	-1,3596149159151800	0,1858396295393610
2	-1,3596149159151800	-0,1537049344800750	4,5456581587371600	-1,3258013450058500	0,0338135709093390
3	-1,3258013450058500	-0,0046249170478299	4,2732476192579200	-1,3247190494171300	0,0010822955887198
4	-1,3247190494171300	-0,0000046577191093	4,2646416796658400	-1,3247179572458600	0,0000010921712676
5	-1,3247179572458600	-0,0000000000047404	4,2646329987489100	-1,3247179572447500	0,0000000000011116

Количество итераций метода Ньютона	Корень метод Ньютона	Корень, полученный поиском решений	Погрешность %
3 (ε<0,01)	**-1,3247190494171300**	**-1,3247179572447500**	**0,0000824456533064**
4 (ε<0,0001)	**-1,3247179572458600**	**-1,3247179572447500**	**0,0000000000835401**
5 (ε<0,0000001)	**-1,3247179572447500**	**-1,3247179572447500**	**0,0000000000003688**

! Можно отметить, что решения методом Ньютона требует намного меньше итераций для достижения требуемой погрешности между двумя соседними решениями.

Метод простой итерации



Необходимо уравнение $f(x) = 0$ равносильным преобразованием привести к виду $x = \phi(x)$.

При этом задача сводится к нахождению абсциссы точки пересечения прямой $y=x$ и кривой $y = \phi(x)$.

Нулевое приближение корня, чаще всего берут равным началу отрезка

$$x_0 = \frac{a+b}{2}$$

Первое приближение корня находят по формуле

$$x_1 = \varphi(x_0)$$

В общем виде k приближение корня находят по формуле

$$x_{k+1} = \varphi(x_k)$$

$$k = 0,1,2,...$$

Остановка итерационной процедуры

$$|x_{k+1} - x_k| < \varepsilon$$

Способ определения $\phi(x)$

1. Выразить из $f(x) = 0$ x, таким образом преобразовать уравнение к виду $x = \phi(x)$. При этом $|\phi'(x)| < 1$.

2. Записать $\phi(x)$ через формулу $x=g(x)f(x)+x \cong \phi(x)$. $g(x)$ часто берут константой, которую можно определить как $g(x)=\pm 1/f'(x_0)$. При этом $|\phi'(x)| < 1$.

Решение задачи методом простых итераций

Найти корень уравнения $x^3 - x + 1 = 0$ методом простых итераций.

1. ϕ 1 – это **ϕ(x) = x³ + 1** при этом ϕ'(x) = 3x² тогда |ϕ'(a)| = |ϕ'(-2)|=3*4=12>1.
   // Такая функция нам не подходит

Тогда делаем преобразование по другому $x = \sqrt[3]{x-1}$ тогда $\varphi(x) = \sqrt[3]{x-1}$,

при этом $\varphi'(x) = \frac{1}{3}(x-1)^{\frac{1}{3}-1} = \frac{1}{3\sqrt[3]{(x-1)^2}}$

тогда $\varphi'(-2) = \frac{1}{3\sqrt[3]{(-2-1)^2}} = \frac{1}{3\sqrt[3]{9}} \approx 0{,}16 < 1$. ***// Такая функция нам подходит***

2. ϕ 2 – это **ϕ(x) = 0,09*(x³ – x + 1) + x** при этом ϕ'(x) = 0,09(3x² – 1) + 1 тогда
   |ϕ'(a)| = |ϕ'(-2)|=|0,09*(3*4-1)+1|=|0,09(12-1)+1|=1,99 > 1. ***// Такая функция нам не подходит***

ϕ 2 – это **ϕ(x) = -0,09*(x³ – x + 1) + x** при этом ϕ'(x) = -0,09(3x² – 1) + 1 тогда
|ϕ'(a)| = |ϕ'(-2)|= |-0,09*(3*4-1)+1|=|-0,09(12-1)+1|=0,01 < 1. ***// Такая функция нам подходит***

!!! От выбора функции ϕ(x) зависит получите вы или нет сходящуюся итерационную процедуру

Решение задачи методом простых итераций

$$\varphi_1(x) = \sqrt[3]{x-1} \qquad \varphi_2(x) = -0,09\left(x^3 - x + 1\right) + x$$

iter	ϕ1			ϕ2		
	xk	xk+1 =ϕ1(xk)	abs((xk+1)-(xk))	xk	xk+1 = ϕ2(xk)	abs((xk+1)-(xk))
0	-1,5	-1,35720880829745	0,1427911917025470	-1,5	-1,42125	0,07875
1	-1,35720880829745	-1,33086095880143	0,0263478494960256	-1,42125000000000	-1,38078544576172	0,04046455423828
2	-1,33086095880143	-1,32588377423235	0,0049771845690798	-1,38078544576172	-1,35812555918506	0,02265988657665
3	-1,32588377423235	-1,32493936340188	0,0009444108304630	-1,35812555918506	-1,34490061068429	0,01322494850078
4	-1,32493936340188	-1,32476001129270	0,0001793521091822	-1,34490061068429	-1,33700773120303	0,00789287948126
5	-1,32476001129270	-1,32472594522689	0,0000340660658156	-1,33700773120303	-1,33223651780383	0,00477121339920
6	-1,32472594522689	-1,32471947453436	0,0000064706925231	-1,33223651780383	-1,32933050956421	0,00290600823963
7	-1,32471947453436	-1,32471824544894	0,0000012290854283	-1,32933050956421	-1,32755251513771	0,00177799442649
8	-1,32471824544894	-1,32471801198820	0,0000002334607387	-1,32755251513771	-1,32646168785786	0,00109082727985
9	-1,32471801198820	-1,32471796764309	0,0000000443451098	-1,32646168785786	-1,32579132643862	0,00067036141924
10	-1,32471796764309	-1,32471795921988	0,0000000084232099	-1,32579132643862	-1,32537893693206	0,00041238950657
11	-1,32471795921988	-1,32471795761992	0,0000000015999613	-1,32537893693206	-1,32512508541972	0,00025385151234
12	-1,32471795761992	-1,32471795731601	0,0000000003039076	-1,32512508541972	-1,32496876342553	0,00015632199418
13	-1,32471795731601	-1,32471795725828	0,0000000000577263	-1,32496876342553	-1,32487247725675	0,00009628616878
14	-1,32471795725828	-1,32471795724732	0,0000000000109650	-1,32487247725675	-1,32481316131365	0,00005931594310
15	-1,32471795724732	-1,32471795724524	0,0000000000020828	-1,32481316131365	-1,32477661713443	0,00003654417921

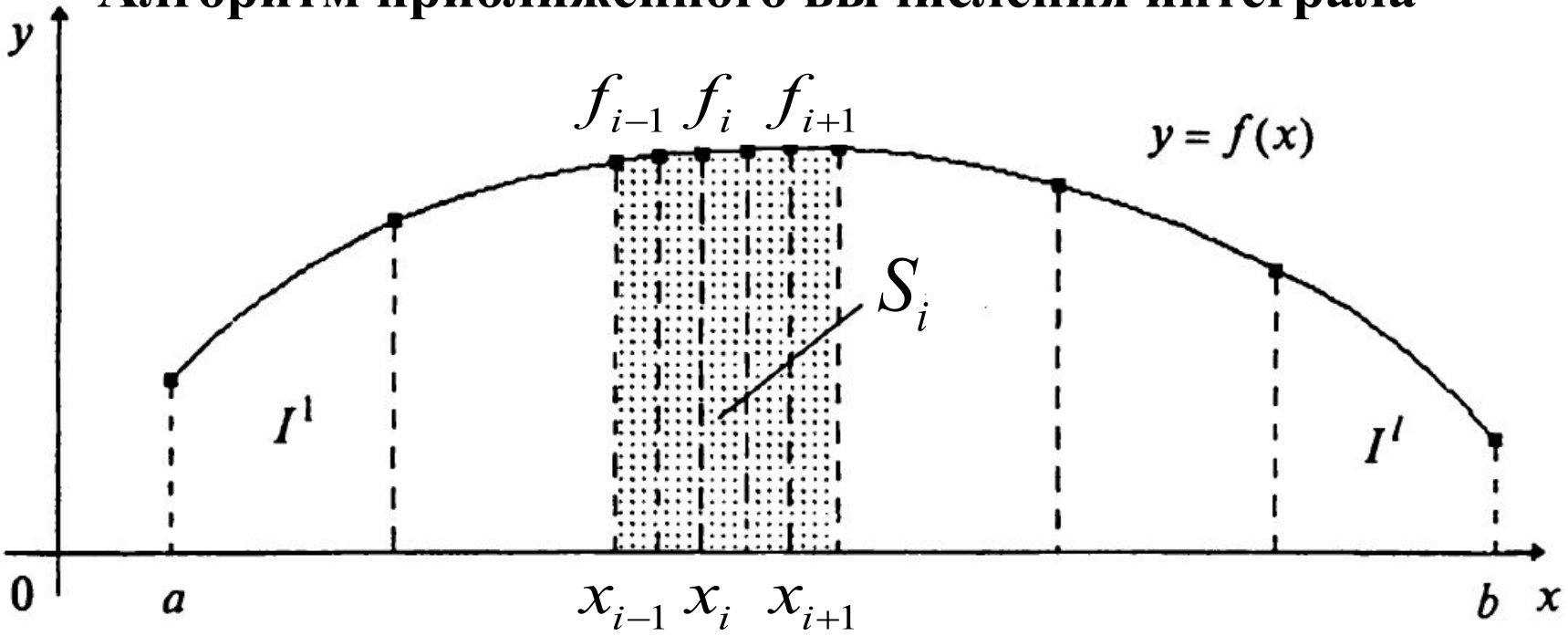
$$\varphi_1(x) = \sqrt[3]{x-1}$$

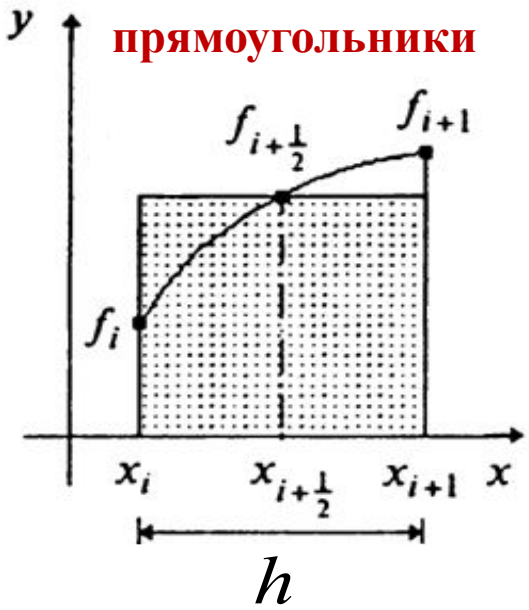
Количество итераций метода простых итераций	Корень метод простых итераций	Корень, полученный поиском решений	Погрешность %
2 (ε<0,01)	**-1,3258837742323500**	**-1,3247179572447500**	**0,0880049206867962**
5 (ε<0,0001)	**-1,3247259452268900**	**-1,3247179572447500**	**0,0006029949312989**
9 (ε<0,0000001)	**-1,3247179676430900**	**-1,3247179572447500**	**0,0000007849471758**

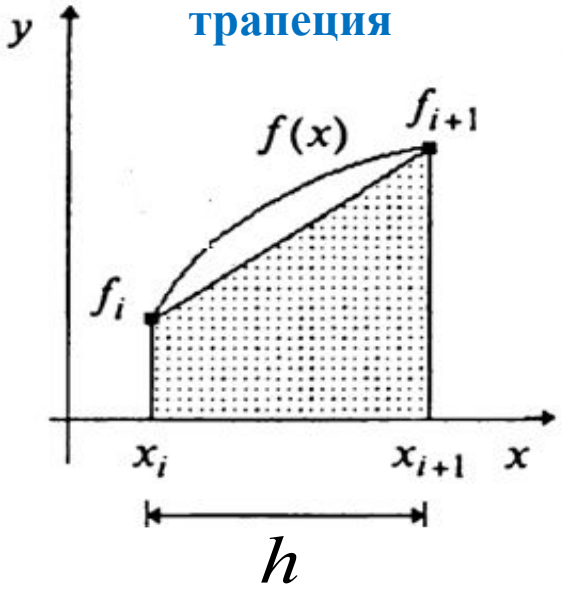
$$\varphi_2(x) = -0,09\left(x^3 - x + 1\right) + x$$

Количество итераций метода простых итераций	Корень метод простых итераций	Корень, полученный поиском решений	Погрешность %
4 (ε<0,01)	**-1,3370077312030300**	**-1,3247179572447500**	**0,9277275884326020**
13 (ε<0,0001)	**-1,3248724772567500**	**-1,3247179572447500**	**0,0116643706046554**
28 (ε<0,0000001)	**-1,3247180655200300**	**-1,3247179572447500**	**0,0000081734588053**

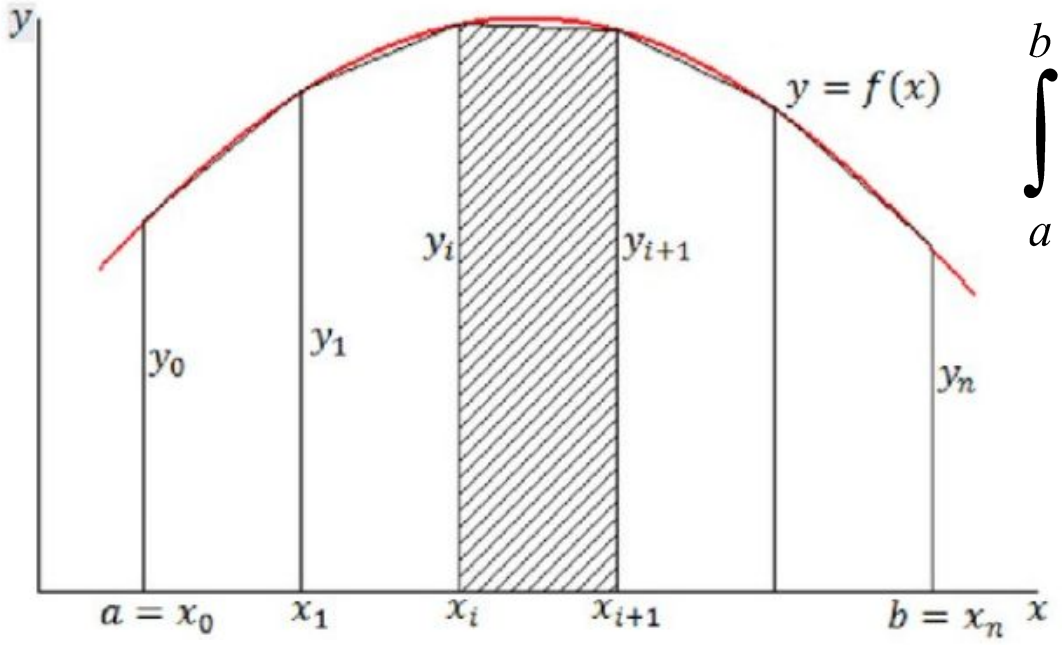
Алгоритм приближённого вычисления интеграла







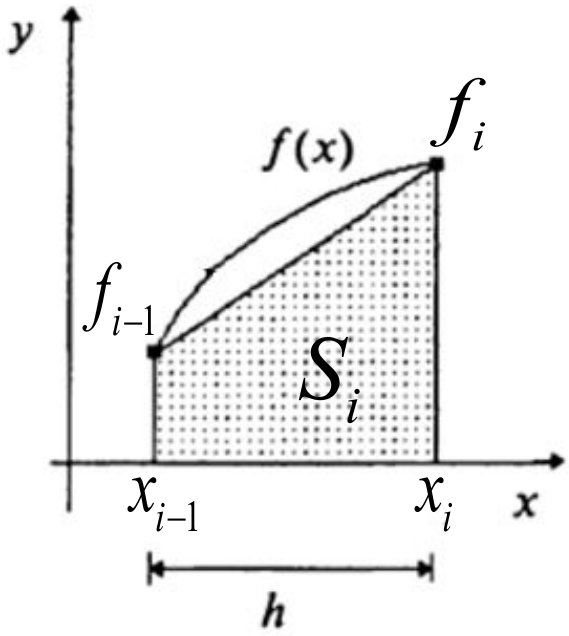
Алгоритм приближённого вычисления интеграла (трапеция)



$$\int_a^b f(x)\,dx \qquad h = \frac{b-a}{n}$$

Площадь (интеграл) вычисляется по формуле

$$S \approx \sum_{i=1}^{n} \frac{f(x_i) + f(x_{i-1})}{2} h$$



$$S_i \approx \frac{f(x_i) + f(x_{i-1})}{2} h$$

Площадь одного кусочка кривой

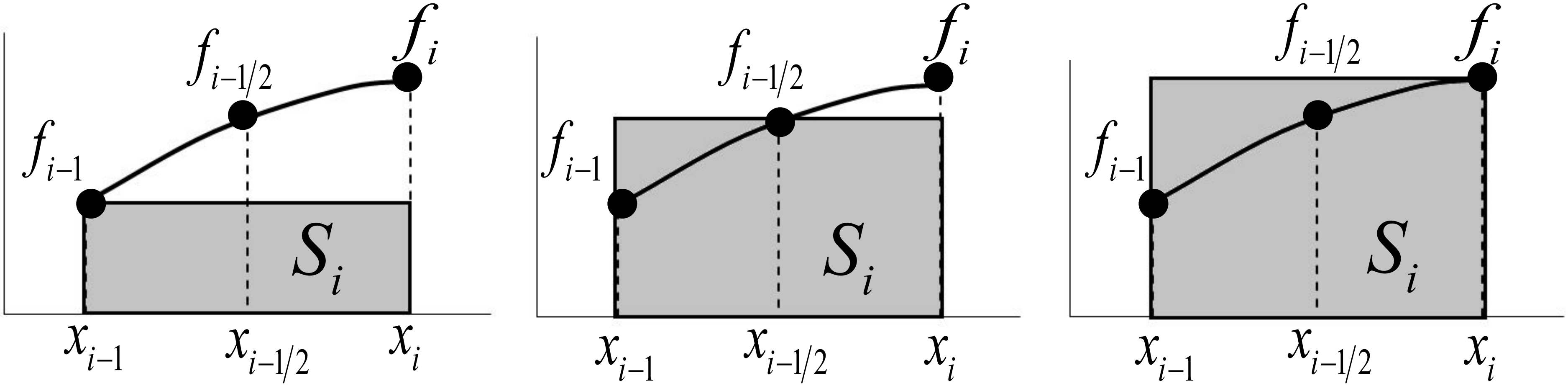
$$f_i = f(x_i)$$

$$f_{i-1} = f(x_{i-1})$$

Пример интеграла

$$\int_a^b \sin(x)\,dx$$

Алгоритм приближённого вычисления интеграла (прямоугольники)



$$h = \frac{b-a}{n} \qquad h = \frac{b-a}{n} \qquad h = \frac{b-a}{n}$$

Площадь одного кусочка кривой

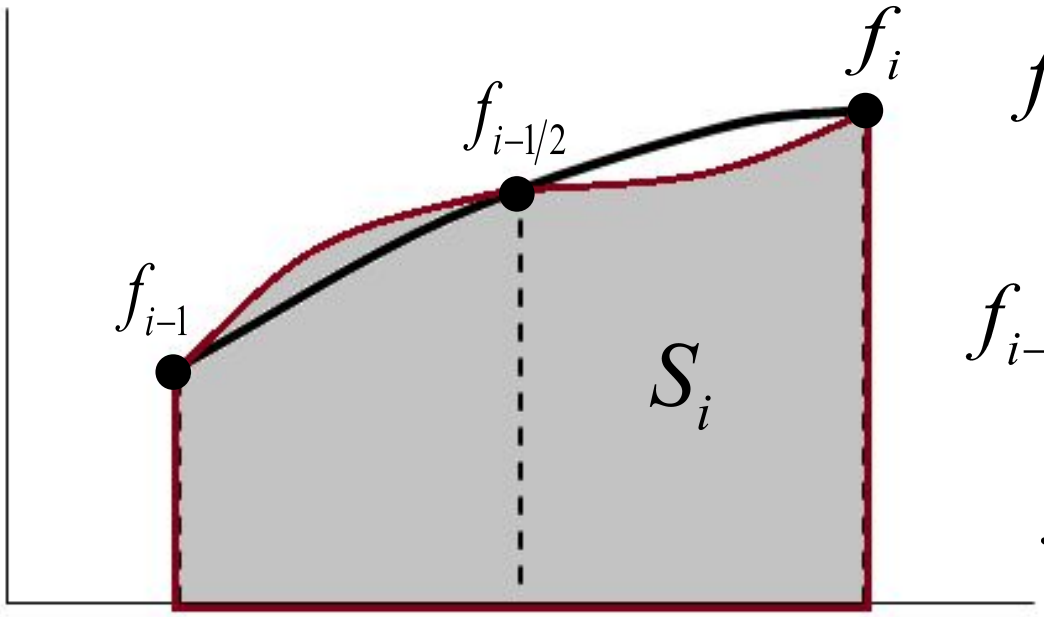
$$S_i \approx f\left(x_{i-1}\right)h \qquad S_i \approx f\left(x_{i-1/2}\right)h \qquad S_i \approx f\left(x_i\right)h$$

Площадь (интеграл) вычисляется по формуле

$$S \approx \sum_{i=1}^{n} f\left(x_{i-1}\right)h \qquad S \approx \sum_{i=1}^{n} f\left(x_{i-1/2}\right)h \qquad S \approx \sum_{i=1}^{n} f\left(x_{i-1}\right)h$$

$$f_{i-1} = f\left(x_{i-1}\right) \qquad f_{i-1} = f\left(x_{i-1/2}\right) \qquad f_i = f\left(x_i\right)$$

Алгоритм приближённого вычисления интеграла (парабола)



$$f_{i-1} = f\left(x_{i-1}\right) \qquad h = \frac{b-a}{n}$$

$$f_{i-1} = f\left(x_{i-1/2}\right)$$

$$f_i = f\left(x_i\right)$$

Площадь одного кусочка кривой

$$S_i \approx \frac{h}{6}\left(f\left(x_{i-1}\right) + 4f\left(x_{i-1/2}\right) + f\left(x_i\right)\right)$$

Площадь (интеграл) вычисляется по формуле

$$S \approx \sum_{i=1}^{n} \frac{h}{6}\left(f\left(x_{i-1}\right) + 4f\left(x_{i-1/2}\right) + f\left(x_i\right)\right)$$