Балтийский государственный технический университет
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра О7 «Информационные системы и программная инженерия»

Практическая работа №2
по дисциплине «Информатика: Основы программирования»
на тему «Ветвления и циклы»

Вариант 14

Выполнил:
Студент Смирнов И.Д.
Группа О719Б

Преподаватель:
Иоффе В.Г.

Санкт-Петербург
2021 г.

Задача 1. Вычислить значение функции $f(m,n)=\begin{cases}\frac{5}{m}-\frac{n}{5}, \text{ если } n>-5, m\neq 0;\\ 3m+n^2, \text{ если } n\leq -5;\\ 2mn \text{ во всех остальных случаях}\end{cases}$,

используя условную операцию ? :

Исходные данные:

Аргументы функции m и n. Так как значения m и n могут быть любыми, объявим соответствующие переменные типа double.

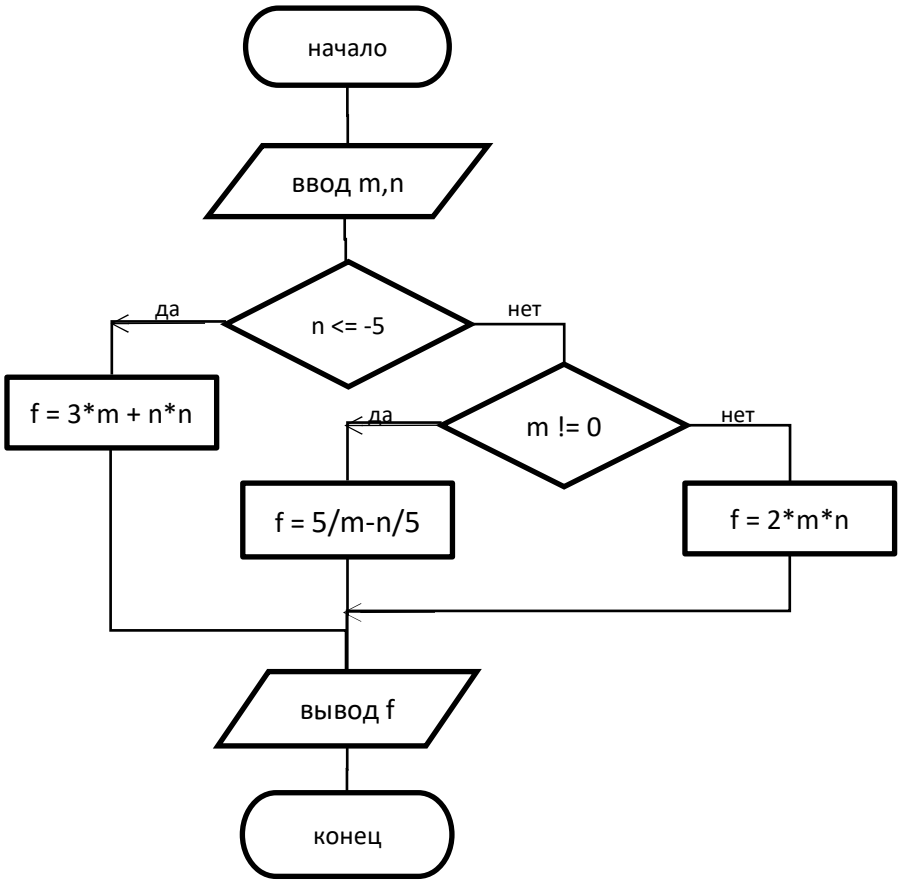
Результирующие данные:

Значение функции f , соответствующая переменная тоже будет типа double.

Таблица тестирования:

Входные данные	**Ожидаемый результат**	**Результат работы программы**
m=2,n=3.	1.9	1.900000
m=1,n=-6	39.0	39.000000
m=0,n=5	0.0	0.000000
m=0,n=-2	0.0	0.000000

Схема программы



Текст программы

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

```
int main(){
    double m,n,f; /* объявление переменных */
    printf("m = ");
    scanf("%lf",&m); /* ввод аргументов функции m,n */
    printf("n = ");
    scanf("%lf",&n);
    f = n<=-5 ? 3*m+n*n : m ? 5/m-n/5 : 2*m*n;
    /* вложенное условие: при ложности условия первого усл. оператора,
программа переходит во второй усл. оператор*/
    printf("f = %lf",f); /* вывод результата */
    return 0;
}
```

Задача 2. Вычислить значение функции $S=\frac{\sin\alpha^3+2\cos^2\beta}{\sqrt{2{,}5\,\alpha+3\,\beta+\sqrt{2}}*\ln\beta}$

Исходные данные:

аргументы a и b — действительные числа, тип double

Результирующие данные:

S — результат функции, тип double

Предварительные вычисления:

Чтобы можно было вычислить значение функции, должны быть выполнены следующие условия: b > 0 (т.к. находится под знаком натурального логарифма), b!=1 (логарифм как множитель в знаменателе не может быть равен 0), подкоренное выражение должно быть больше 0;

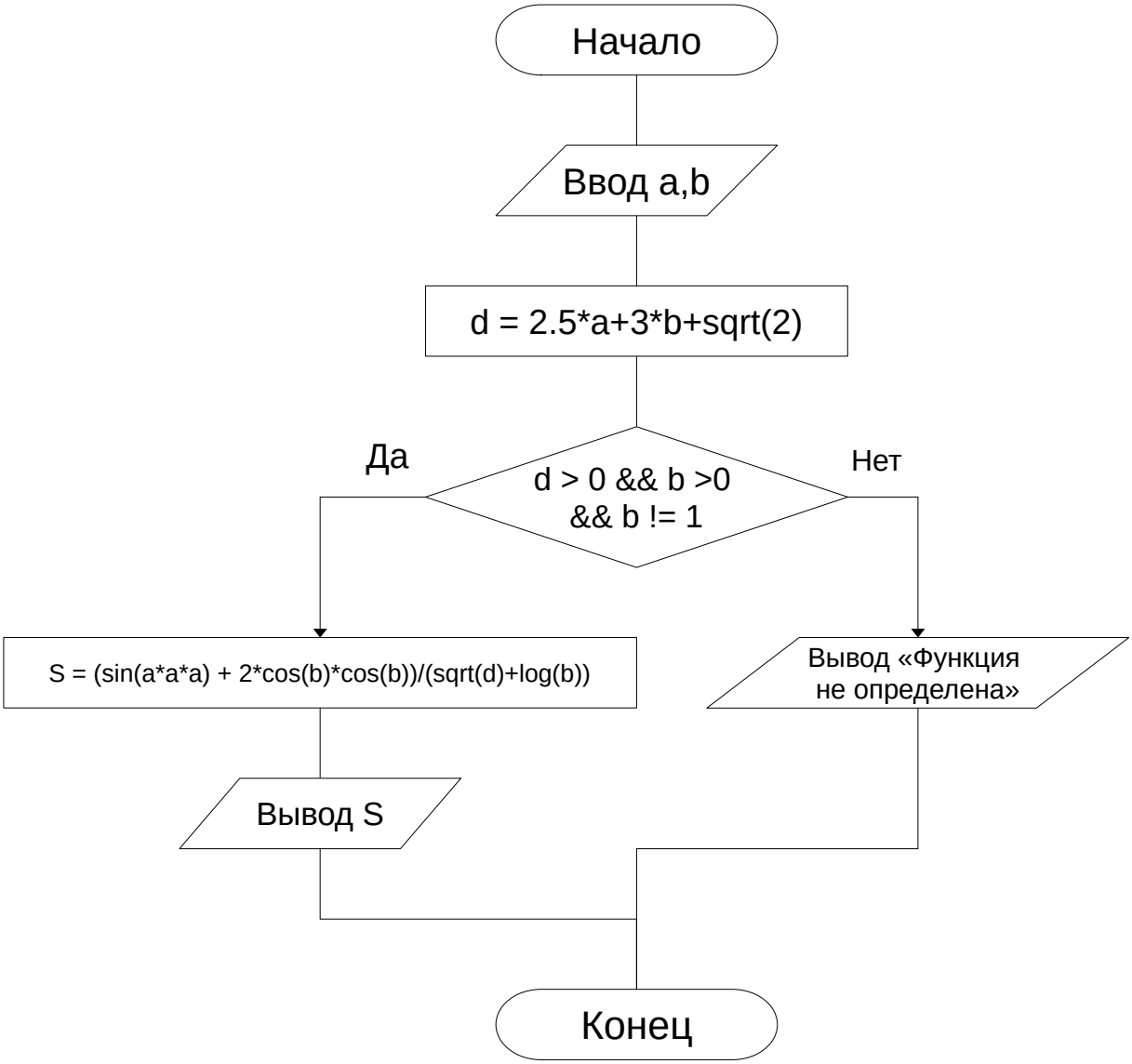
Вспомогательные переменные:

d – подкоренное выражение, тип double.

Таблица тестирования:

Входные данные	**Ожидаемый результат**	**Результат работы программы**
a=1,b=2	0.54425	0.544250
a=1,b=0	Функция не определена	Функция не определена
a=-22,b=5	Функция не определена	Функция не определена
a=5,b=1	Функция не определена	Функция не определена
a=0,5,b=0,5	-1.177108	-1.177108

Схема программы:



Текст программы

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

int main(){
    double a,b,s; /* объявление переменных */
    printf("a = ");
    scanf("%lf",&a);
    printf("b = ");
    scanf("%lf",&b); /* ввод аргументов функции */
    double d = 2.5*a+3*b+sqrt(2); /* подкоренное выражение */
    if (d > 0 && b > 0 && b != 1){ /* проверка на соответствие ООФ */
        s = (sin(a*a*a) + 2*cos(b)*cos(b))/(sqrt(d)*log(b));
        printf("%lf\n",s); /* вычисление и вывод результата */
    }
    else{
```

```
        printf("Функция не определена\n"); /* в случае несоответствия - вывод
сообщения об ошибке */
    }
    return 0;}
```

Задача 3. В финале шашечной партии остались белая дамка и две черные пешки, позиции которых известны. Ход белых. Сможет ли дамка «срубить» хотя бы одну пешку?

Исходные данные:

queen_x, queen_y — координаты дамки,

x1, y1 — координаты первой пешки,

x2, x2 — координаты второй пешки,

все переменные типа int

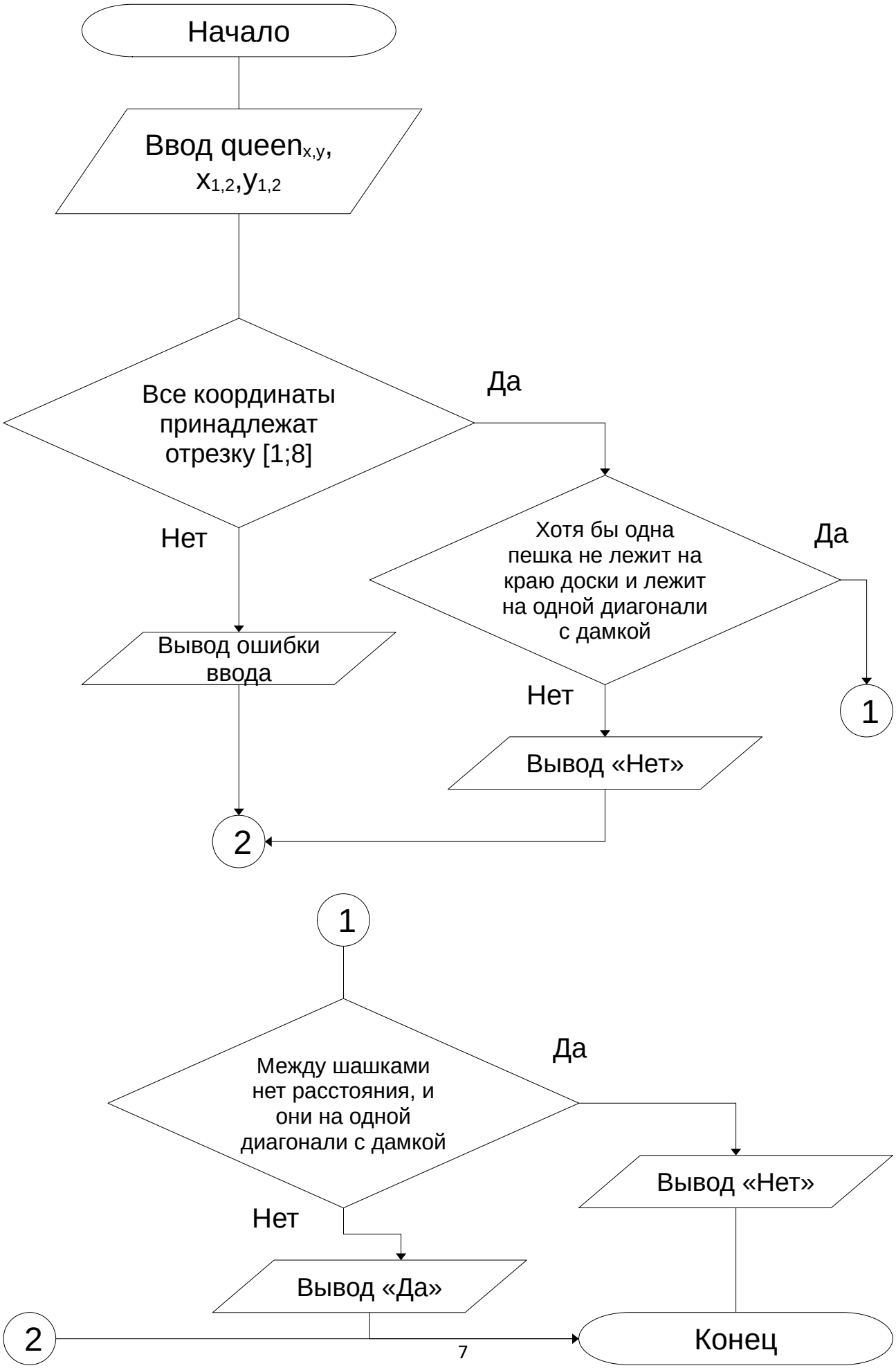
Результирующие данные:

Ответ на вопрос («Да» / «Нет») или сообщение об ошибке

Таблица тестирования:

Входные данные	**Ожидаемый результат**	**Результат работы программы**
Queen x,y — 2 7 Pawn 1 x,y — 5 4 Pawn 2 x,y — 6 5	Да	Да
Queen x,y — 6 7 Pawn 1 x,y — 4 5 Pawn 2 x,y — 3 4	Нет	Нет
Queen x,y — 3 1 Pawn 1 x,y — 6 4 Pawn 2 x,y — 1 8	Да	Да
Queen x,y — 2 9 Pawn 1 x,y — 5 1 Pawn 2 x,y — 7 4	Ошибка	Ошибка
Queen x,y — 1 1 Pawn 1 x,y — 9 9 Pawn 2 x,y — 4 4	Ошибка	Ошибка
Queen x,y — 2 1 Pawn 1 x,y — 8 7 Pawn 2 x,y — 1 2	Нет	Нет

Схема программы:


Начало
Ввод queen$_{x,y}$, $x_{1,2}$,$y_{1,2}$
Все координаты принадлежат отрезку [1;8]
Да
Нет
Вывод ошибки ввода
Хотя бы одна пешка не лежит на краю доски и лежит на одной диагонали с дамкой
Да
Нет
Вывод «Нет»
1
2
1
Между шашками нет расстояния, и они на одной диагонали с дамкой
Да
Вывод «Нет»
Нет
Вывод «Да»
2
Конец

Текст программы

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

int main(){
    int queen_x, queen_y, x1,y1,x2,y2; /* объявление переменных */
    printf("Queen x,y - ");
    scanf("%d%d",&queen_x,&queen_y);
    printf("Pawn 1 x,y - ");
    scanf("%d%d",&x1,&y1);     /* ввод координат дамки и шашек */
    printf("Pawn 2 x,y - ");
    scanf("%d%d",&x2,&y2);
    if(queen_x >= 1 && queen_x <= 8 && queen_y >= 1 && queen_y <= 8 && x1 >=
1 && x1 <= 8 && y1 >= 1 && y1 <= 8 && x2 >= 1 && x2 <= 8 && y2 >= 1 && y2 <=
8){
        /* если координаты не выходят за пределы доски */
        if(((abs(queen_x - x1) == abs(queen_y - y1)) && x1 > 1 && x1 < 8 &&
y1 > 1 && y1 < 8) || ((abs(queen_x - x2) == abs(queen_y - y2)) && (x2 > 1) &&
(x2 < 8) && (y2 > 1) && (y2 < 8))){
        /* если модуль разности координат дамки и хотя бы одной шашки равны,
а хотя бы одна из шашек не лежит на краю */
            if(abs(x1-x2) == 1 && abs(y1-y2) == 1 && abs(queen_x - x1) ==
abs(queen_y - y1) && abs(queen_x - x2) == abs(queen_y - y2)){
                printf("Нет"); /* если между шашками нет расстояния, и они на
одной диагонали с дамкой */
            }
            else{
                printf("Да");
            }
        }
        else{
            printf("Нет");
        }
    }
    else{
    /* если координаты не на шахматной доске*/
    printf("Ошибка: координаты выходят за пределы шахматной доски");
    }
    return 0;
}
```

Задача 4. Составить программу для определения, в каких двузначных числах удвоенная сумма цифр равна их произведению. Использовать управляющую инструкцию *for*.

Исходные данные:

Все двузначные числа, обозначим i, типа char будет достаточно

Результирующие данные:

Вывод сообщения. Отдельной переменной не требуется.

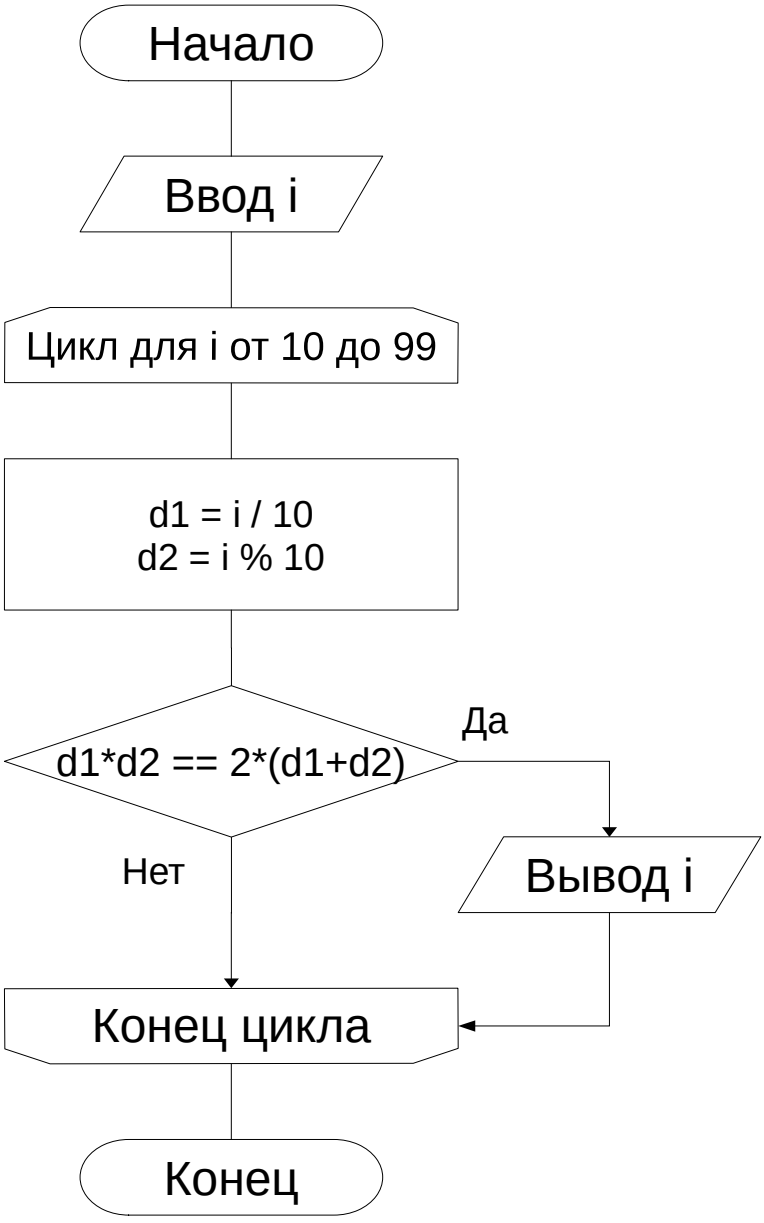
Вспомогательные переменные:

d1, d2 — первая и вторая цифра числа соответственно, используются для проверки условия

Таблица тестирования:

Входные данные	Ожидаемый результат	Результат работы программы
	36 44 63	36 44 63

Схема программы:



Текст программы

```
#include <stdio.h>

int main(){
    char d1, d2,i; /* объявление вспомогательных переменных и переменной-
счётчика i */
    for(i = 10; i < 100; i++){
        d1 = i / 10; /* первая цифра - целая часть от деления на 10 */
        d2 = i % 10; /* вторая цифра - остаток от деления на 10 */
        if(2*(d1 + d2) == d1 * d2){ /* проверка условия задачи */
            printf("%d\n",i); /* вывод на экран */
```

```
        }
    }
    return 0;
}
```

Задача 5. Несколько деталей должны последовательно пройти обработку на каждом из трех станков. Продолжительности обработки каждой детали на каждом станке вводятся группами по три числа до исчерпания ввода (признак окончания ввода – задание некорректной тройки чисел). Сколько времени займет обработка всех деталей, если на каждом станке они могут обрабатываться только поштучно?

Исходные данные:

part1, part2, part3 — время обработки деталей на первом, втором и третьем станке, тип int

Вспомогательные переменные:

machine1_time, machine2_time, machine3_time — координаты времени первого, второго и третьего станка соответственно, тип int

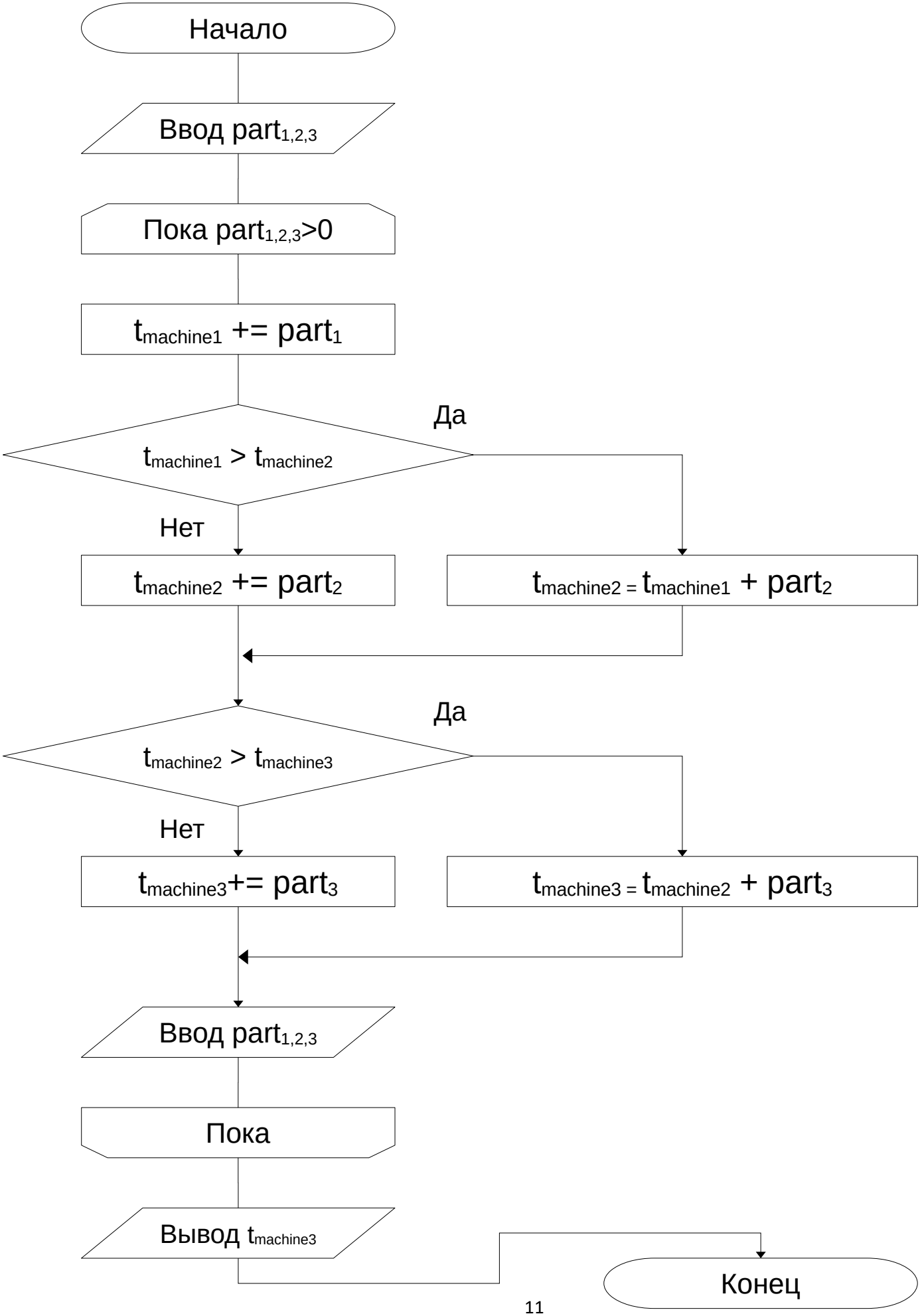
Результирующие данные:

machine3_time — координаты времени третьего станка

Таблица тестирования:

Входные данные	**Ожидаемый результат**	**Результат работы программы**
1 2 1 1 2 2 2 3 1 3 1 1 2 1 1 0 -1 2	11	11
4 3 2 -1 0 0	9	9
34 -5 2	0	0
1 1 2 2 1 1 0 0 0	5	5
45 93 23 89 34 11 45 78 2 92 0 4	259	259

Схема программы:


Начало
Ввод $part_{1,2,3}$
Пока $part_{1,2,3}>0$
$t_{machine1}$ += $part_1$
Да
$t_{machine1} > t_{machine2}$
Нет
$t_{machine2}$ += $part_2$
$t_{machine2} = t_{machine1} + part_2$
Да
$t_{machine2} > t_{machine3}$
Нет
$t_{machine3}$+= $part_3$
$t_{machine3} = t_{machine2} + part_3$
Ввод $part_{1,2,3}$
Пока
Вывод $t_{machine3}$
Конец

Текст программы

```
#include <stdio.h>

int main(){
    int part1, part2, part3, machine1_time = 0, machine2_time = 0,
machine3_time = 0; /* объявление переменных */
    scanf("%d%d%d",&part1,&part2,&part3); /* ввод первой тройки */
    while(part1 > 0 && part2 > 0 && part3 > 0){ /* считываем данные до первой
некорректной тройки чисел */
        machine1_time += part1; /* добавляем время обработки первой детали */
        machine2_time = (machine1_time > machine2_time) ? machine1_time+part2
: machine2_time+part2; /* проверяем, занят ли 2 станок (нужно ли сдвигать
обработку детали)

и присваиваем получившуюся координтау времени*/
        machine3_time = (machine2_time > machine3_time) ? machine2_time+part3
: machine3_time+part3; /* аналогично проверяем третий станок */
        scanf("%d%d%d",&part1,&part2,&part3); /* ввод новой тройки */
    }
    printf("Общее время на обработку деталей - %d", machine3_time); /* вывод
результата - координата времени окончания работы третьего станка */
    return 0;
}
```

Задача 6. Получить число, образованное записью цифр исходного числа

N в обратном порядке.

Исходные данные:

n — изначальное число, тип int.

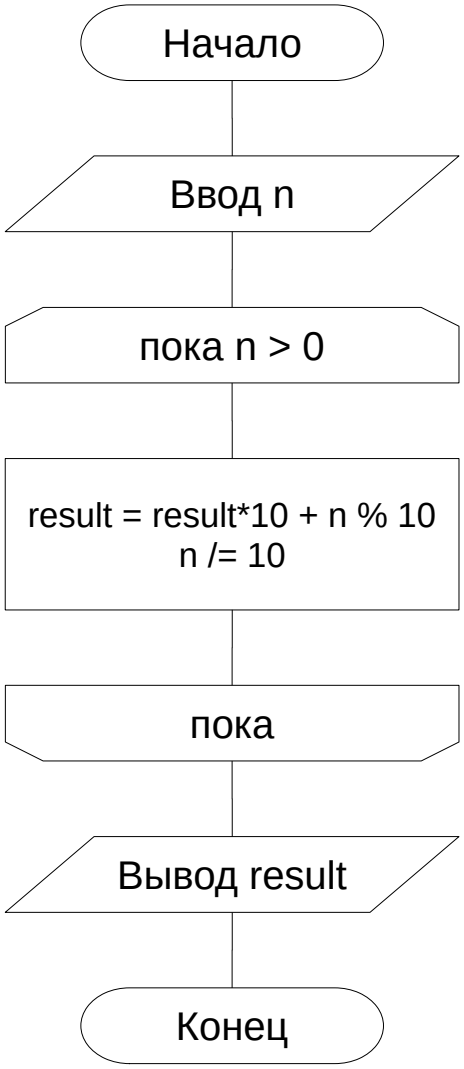
Результирующие данные:

result — число, образованное записью цифр в обратном порядке, тип int.

Таблица тестирования:

Входные данные	**Ожидаемый результат**	**Результат работы программы**
874563	365478	365478
-4525345	0	0
41753465843756	65734856435714	65734856435714
2	2	2

Схема программы:



Текст программы

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

int main(){
    long int n,result = 0; /* объявление переменных */
    printf("Изначальное число - ");
    scanf("%li",&n); /* ввод изначального числа */
    while(n > 0){
        result = result*10 + n % 10; /* разворачиваем его, прибавляяя остаток
от деления к разряду результата */
        n /= 10; /* делим само число на 10 */
    }
    printf("Результат - %li",result); /* вывод на экран */
    return 0;
}
```

Задача 7. Вычислить значение суммы бесконечного ряда с заданной точностью $\varepsilon=10^{-5}$

$f(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-...+\frac{(-1)^n x^{2n}}{(2n)!}$ и значение функции (для проверки) $y=\cos x$.

Использовать рекуррентные зависимости для вычисления значений слагаемых.

Исходные данные:

Аргумент функции x. Он может быть любой, поэтому тип переменной - double

Результирующие данные:

Значение суммы s тоже будет типа double.

Вспомогательные переменные:

n – индекс слагаемого – целое число типа int, a – значение текущего слагаемого – вещественное число типа double.

Предварительные вычисления:

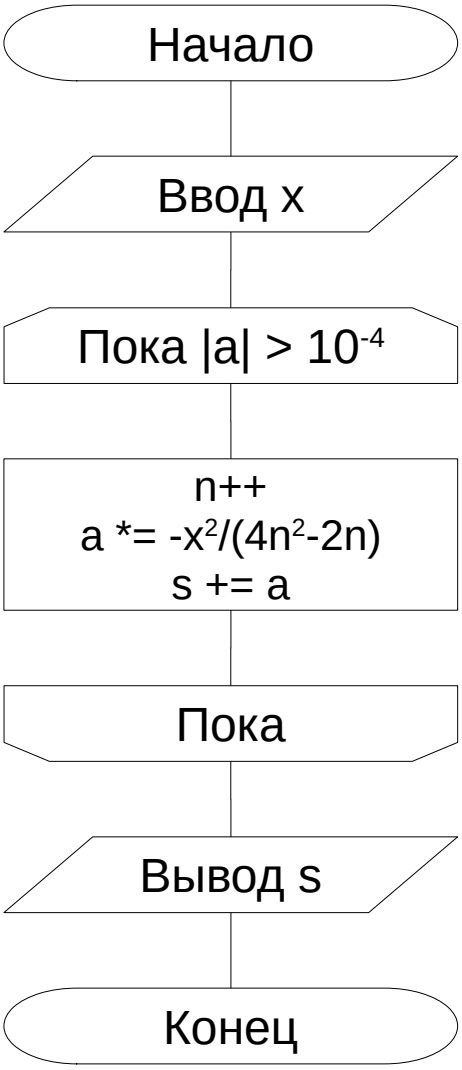
n-ное слагаемое: $a_n = \frac{(-1)^n x^{2n}}{(2n)!}$,

предшествующее слагаемое: $a_{n-1} = \frac{(-1)^{n-1} x^{2(n-1)}}{(2(n-1))!} = \frac{(-1)^{n-1} x^{2n-2}}{(2n-2)!}$

коэффициент пропорциональности: $\frac{a_n}{a_{n-1}} = \frac{\frac{(-1)^n x^{2n}}{(2n)!}}{\frac{(-1)^{n-1} x^{2n-2}}{(2n-2)!}} = \frac{-x^2}{(2n)(2n-1)}$

индекс первого слагаемого в рекуррентной последовательности n=0, первое слагаемое: $a_0 = \frac{(-1)^0 x^0}{0!} = 1$

Схема программы:



Текст программы

```
#include <stdio.h>
#include <math.h>
#define EPS 1e-5

int main(){
    double x,a=1,s=0;           /* объявление переменных x (аргумент), a
(слагаемое), s (результат) */
    int n = 0;                  /* объявление индекса слагаемого n */
    s += a;                     /* прибавляем значение первого слагаемого к
результату */
    printf("Значение аргумента - ");
    scanf("%lf",&x);            /* ввод значения аргумента*/
    printf("Ожидаемое значение: %.5f\n",cos(x));
    while(fabs(a) > EPS){   /* пока слагаемое не достигнет заданной точности
*/
        n++;                    /* прибавляем индекс */
        a *= -x*x/(4*n*n-2*n);  /* домножаем слагаемое на коэффициент
пропорциональности */
        s += a;                 /* прибавляем слагаемое к результату */
    }
    printf("Результат: %.5f",s);  /* вывод результата */
```

```
    return 0;
}
```

Результаты тестирования

Исходные данные	Ожидаемый результат	Результат работы программы
0	1	1.00000
1	0.54030	0.54030
3,1415926	-1	-1.00000
0.5	0.87758	0.87758
6,283	1	1.00000

Задача 8. Вычислить значение суммы членов бесконечного ряда:

$$S = \ln x + 2\left[\frac{a}{2x+a} + \frac{a^3}{(2x+a)^3} + \frac{a^5}{(2x+a)^5} + \ldots + \frac{a^{2n-1}}{(2n-1)(2x+a)^{2n-1}} + \ldots\right]$$

Использовать рекуррентные зависимости для вычисления значений слагаемых по частям.

Исходные данные:

x,a — аргументы, тип double.

Результирующие данные:

result — результат значения, тип double.

Вспомогательные переменные:

n — индекс слагаемого, тип int, member — член последовательности, тип double

Предварительные вычисления:

$$n-ое\,слагаемое: \frac{a^{2n-1}}{(2n-1)(2x+a)^{2n-1}}$$

$$(n-1)-ое\,слагаемое: \frac{a^{2n-3}}{(2n-3)(2x+a)^{2n-3}}$$

$$коэф.пропорциональности: \frac{\frac{a^{2n-1}}{(2n-1)(2x+a)^{2n-1}}}{\frac{a^{2n-3}}{(2n-3)(2x+a)^{2n-3}}} = \frac{a^{2n-1}(2n-3)(2x+a)^{2n-3}}{a^{2n-3}(2n-1)(2x+a)^{2n-1}} = \frac{a^2(2n-3)}{(2n-1)(2x+a)^2}$$

$$Первое\,слагаемое\,с\,индексом\ n=1: \frac{a}{2x+a}$$

Условие также можно упростить:

$a^2 < (2x+a)^2 \Rightarrow a^2 < 4x^2 + 4ax + a^2 \Rightarrow 4x^2 + 4ax > 0 \Rightarrow x(x+a) > 0$

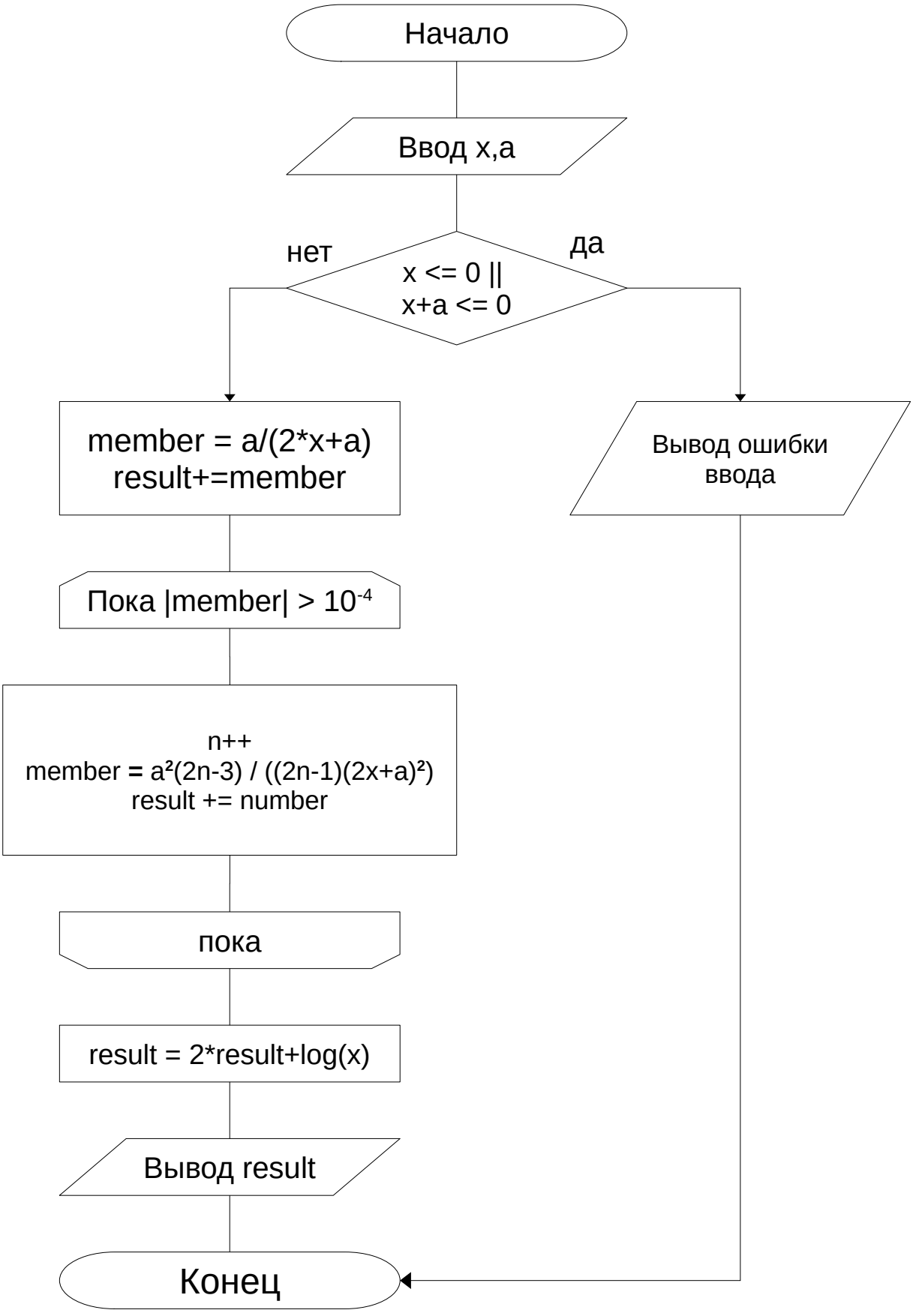
По условию:

$\left.\begin{matrix} x>0 \\ x+a>0 \end{matrix}\right\}$, *значит* $x(x+a)>0$, *тогда*

в программе можно проверить на ложность:

$\left.\begin{matrix} x \le 0 \\ x+a \le 0 \end{matrix}\right\}$

Схема программы:



Текст программы

```
#include <stdio.h>
#include <math.h>
#define EPS 1e-4
 int main(){
    double x,a,member,result=0; /* объявление переменных */
    int n = 1;
    printf("x = ");
    scanf("%lf",&x);
    printf("a = ");
    scanf("%lf",&a); /* ввод аргументов x и a*/
    if(x <= 0 || x+a<=0){
        printf("Ошибка ввода - неверные значения аргументов"); /* если
аргументы не удовлетворяют условию задачи - программа завершается с
сообщением об ошибке */
        return 0;
    }
    member = a/(2*x+a); /* первый член последовательности */
    result+=member; /* прибавляем его к результату */
    printf("Ожидаемое значение: %.4f\n",log(x+a)); /* вывод на экран
ожидаемого значения */
    while(fabs(member) > EPS){ /* пока член последовательности не достигнет
заданной точности */
        n++;
        member *= a*a*(2*n-3) / ((2*n-1)*(2*x+a)*(2*x+a)); /* домножаем член
последовательности на коэффициент пропорциональности */
        result += member; /* добавляем его к результату */
    }
    result = 2*result+log(x); /* домножаем результат на 2 и прибавляем нат.
логарифм аргумента по условию задачи */
    printf("Полученное значение: %.4f",result); /* вывод результата на экран
*/
    return 0;
}
```

Результаты тестирования

Исходные данные	**Ожидаемый результат**	**Результат работы программы**
x=0.5, a=0.5	0	-0.0000
x=1, a=1	0.6931	0.6931
x=0.5, a=3	1.2528	1.2526
x=4, a=-3.5	-0.6931	-0.6930
x=10, a=10	2.9957	2.9957

Задача 9. Вычислить $P = \prod_{i=1}^{10} \sum_{j=1}^{20} \frac{1}{i+j^2}$

(произведение 10 сомножителей, каждое из которых является суммой 20 дробей).

Исходные данные:

i, j — переменные-счётчики, тип int.

Результирующие данные:

P — результат выражения, тип double.

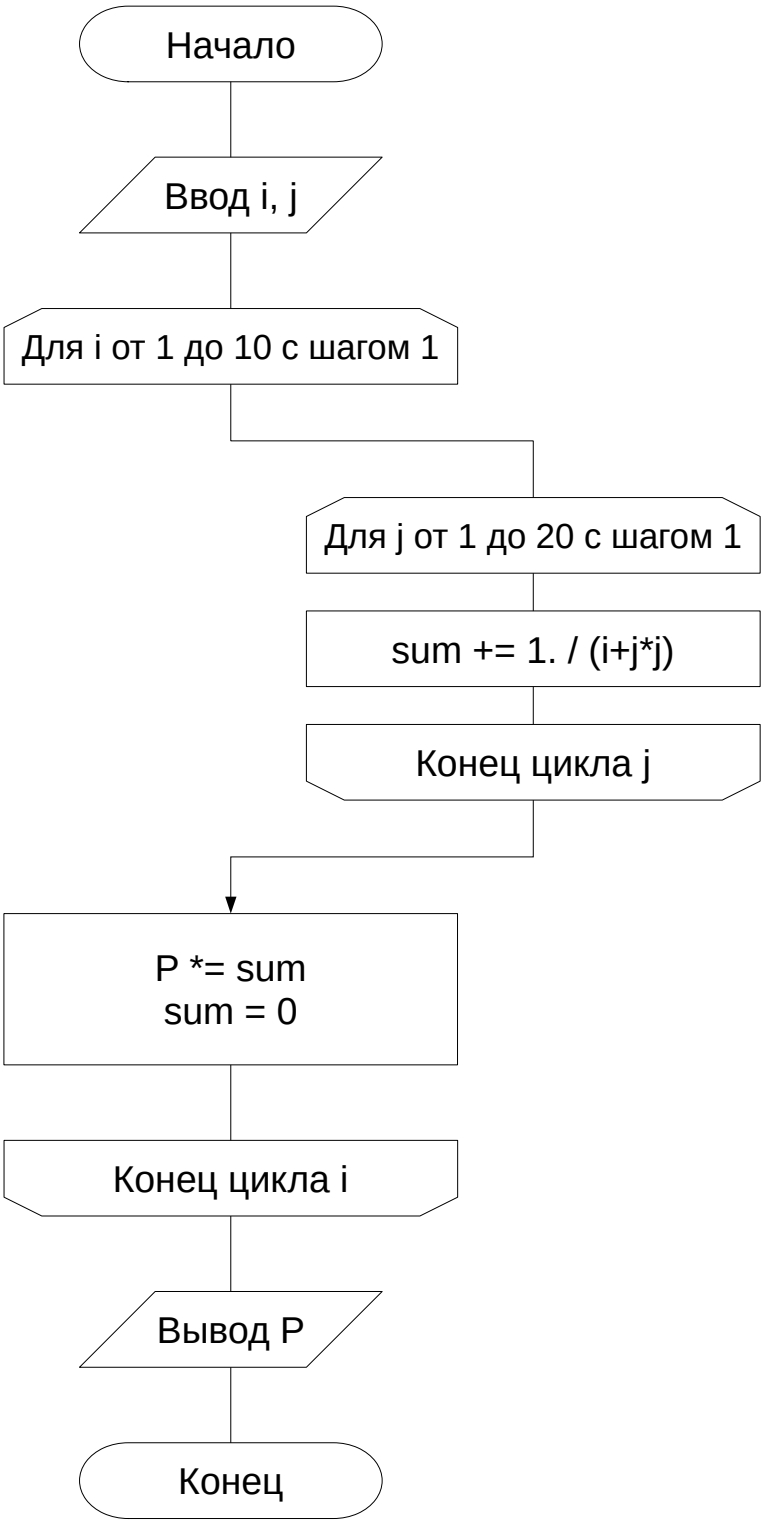
Вспомогательные переменные:

sum — переменная, хранящая сумму до домножения, тип double.

Таблица тестирования:

Ожидаемый результат	Результат работы программы
0.003508369293155	0.003508369293155

Схема программы:



Текст программы

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

int main(){
    int i,j; /* объявление переменных счётчиков */
    double P = 1, sum = 0; /* объявление переменных произведения и суммы */
    for(i = 1; i <= 10; i++){ /* пока i не достигнет 10 */
        for(j = 1; j <= 20;j++){ /* пока j не достигнет 20 */
            sum += 1. / (i+j*j); /* вычисление значения дроби */
        }
        P *= sum; /* умножение полученной суммы на общее произведение */
        sum = 0; /* обнуление суммы для вычисления новой */
    }
    printf("%.15lf",P); /* вывод результата с точностью 15 знаков после
запятой */
    return 0;
}
```