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

Кафедра O7 «Информационные системы и программная инженерия»

**Практическая работа №2**по дисциплине «Информатика: Основы программирования» на тему

«Ветвления и циклы»

Вариант 5

Выполнил:  
Студент Вяткин Н.А.  
Группа О722Б  
  
Преподаватель: Назарова М.А.

Санкт-Петербург  
2022 г.

1. Вычислить значение функции, используя условную операцию ?:

*Исходные данные:*

Аргументы функции m и k. Так как значения m и k 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** |
| k=5, m=40 | 8 | 8.000000 |
| k=-1,5, m=-2,5 | 4 | 4.000000 |
| k=-13,m=1,2 | -15,6 | -15.600000 |
| k=0, m=5 | 0 | 0.000000 |

Схема программы

начало

конец

вывод *f*

Ввод k, m

m<0

k>0

f=|m+k|

f=m/k

f=mk

да

да

нет

нет

Текст программы

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int main()

{

double k, m, f;

printf("k=");

scanf("%lf", &k);

printf("m=");

scanf("%lf", &m);

f=k>0?m/k:m<0?fabs(m+k):k\*m;

printf("%lf", f);

return 0;

}

1. Вычислить значение функции

*Исходные данные:* Тип аргумента в задании не указан, поэтому переменные a, b будут типа double.

*Результирующие данные:* Значение функции d тоже будет типа double.

*Предварительные вычисления:*

Чтобы вычислить значение функции, должны быть выполнены следующие условия: cosa≠0, ненулевой знаменатель дроби, неотрицательное подкоренное выражение.

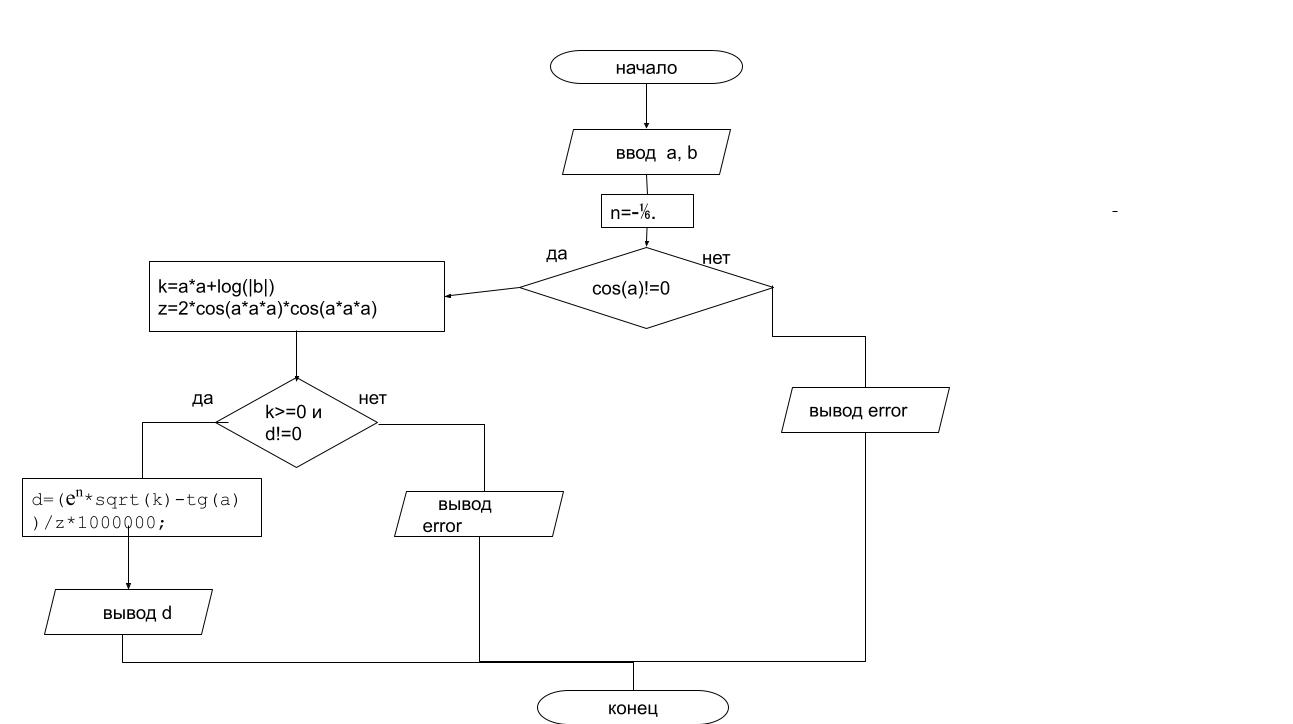
*Вспомогательные переменные:*

k – Подкоренное выражение, z – знаменатель второй дроби, n-переменная для запись вещественной степени e; все три переменные типа double.

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
| a=3 b=1 | 1.57126\*106 | 15712631.622159 |
| a=3 b=0,00001 | error | error |
| a=5 b=2 | 6.18152\*106 | 6181521.685239 |
| a=9 b=8 | 4,17783\*106 | 4177832.508264 |

Схема программы:



Текст программы

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int main()

{

double a, b, k, d, z, n=-1/6.;

printf ("a=");

scanf("%lf", &a);

printf("b=");

scanf("%lf", &b);

if (cos(a))

{

k=a\*a+log(fabs(b));

z=2\*cos(a\*a\*a)\*cos(a\*a\*a);

if (k>=0 && z)

{

d=(exp(n)\*sqrt(k)-tan(a))/z\*1000000;

printf("d=%lf\n", d);

}

else

printf("error");

return 0;

}

else

printf("error\n");

system("pause");

return 0;

}

1. Даны координаты двух точек на плоскости. Если хотя бы одна из них лежит на какой-нибудь оси, вывести сообщение об этом; если они обе находятся в одной четверти, найти расстояние между ними; иначе – найти точку, наиболее удаленную от центра координат.

*Исходные данные:* На вход мы получаем 4 числа- координаты двух точек, поэтому переменные x1, y1 x2, y2 тип int.

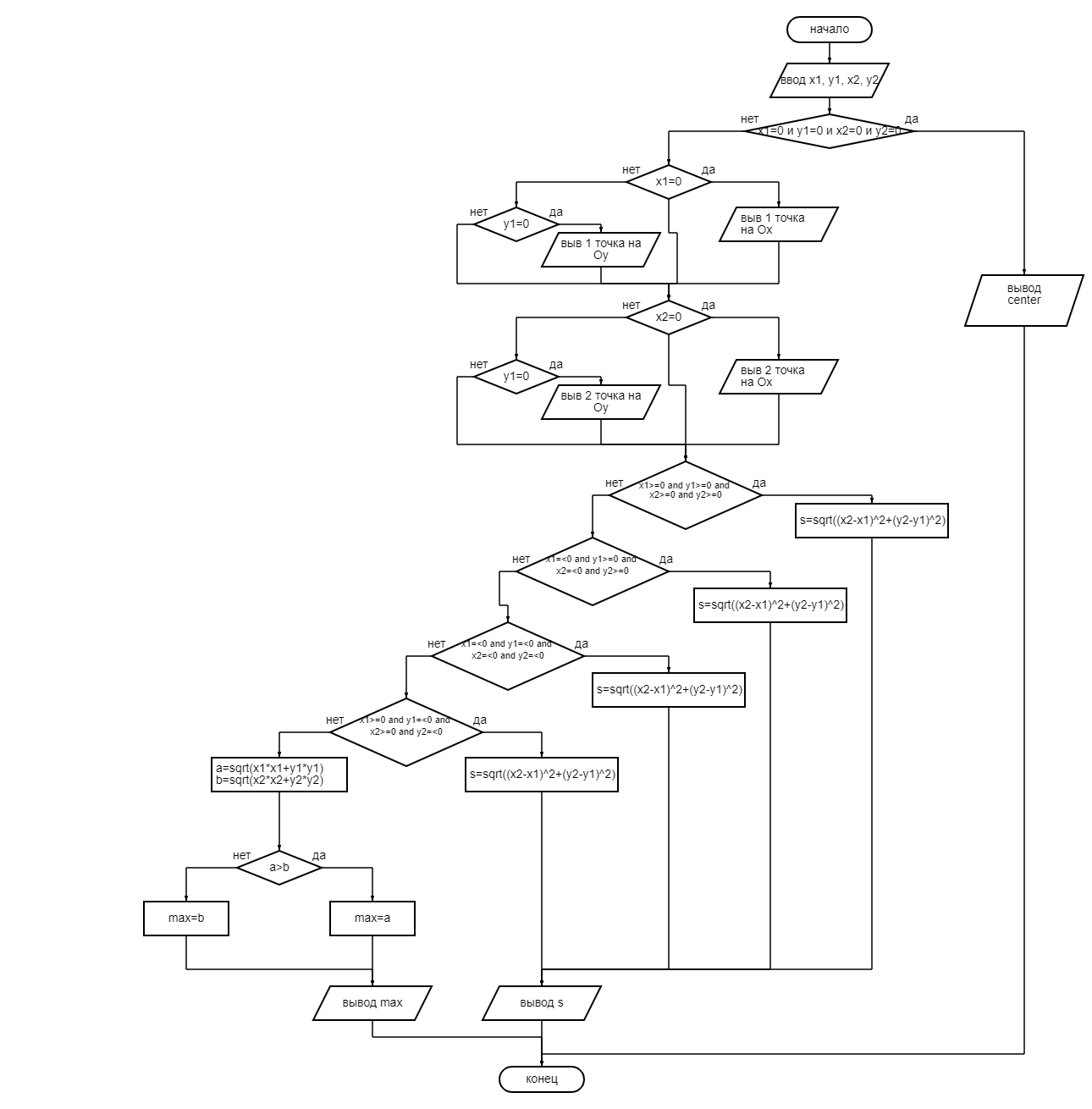
*Результирующие данные:* Переменная s- расстояние от точки до точки, тип double, max-максимальное расстояние от точки до центра координат, тип double

*Вспомогательные переменные:* Переменные a и b тип double для сохранения расстояния от точки до центра и сравнения для определения max

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
| x1=3, y1=4, x2=3, y2=6 | 2 | 2.000 |
| x1=-4, y1=4, x2=-5, y2=6 | 2.23606798 | 2.236 |
| x1=4, y1=4, x2=-5, y2=6 | 7.81024968 | 7.810 |
| x1=4, y1=-41, x2=-5, y2=36 | 41,1946598 | 41.195 |

Схема программы:



Текст программы

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int main()

{

int x1, y1, x2, y2;

double s, max, a, b;

printf("Enter the coordinates of the first point\n");

printf("input x1=");

scanf("%d", &x1);

printf("input y1=");

scanf("%d", &y1);

printf("Enter the coordinates of the seconds point\n");

printf("input x2=");

scanf("%d", &x2);

printf("input y2=");

scanf("%d", &y2);/\*Ввод координат\*/

if (x1==0 && x2==0 && y1==0 && y2==0)/\* проверяем не находятся ли точки в центре, иначе считаем, то что требуют в задании\*/

{

printf("center");/\* если обе точки находятся в центре заканчиваем программу\*/

return 0;

}

if (x1==0 && x2==0 && y1==0 && y2==0)

{

printf("center");

return 0;

}

else

{

if (x1==0)

printf("first point belongs to Ox\n");

else

if (y1==0)

printf("first point belongs to Oy\n");/\*проверка на принадлежность к Оу или Ох\*/

if (x2==0)

printf("second point belongs to Ox\n");

else

if (y2==0)

printf("second point belongs to Oy\n");/\*проверка на принадлежность к Оу или Ох\*/

if (x1>=0 && y1>=0 && x2>=0 && y2>=0)

{

printf("points belong 1 quater\n");

s=sqrt((x2-x1)\*(x2-x1)+(y2-y1)\*(y2-y1));

printf("s=%.3lf\n", s);/\*если обе точки лежат в первой четверти, находим расстояние между ними\*/

}

else

if (x1<=0 && y1>=0 && x2<=0 && y2>=0)

{

printf("points belong 2 quater\n");

s=sqrt((x2-x1)\*(x2-x1)+(y2-y1)\*(y2-y1));

printf("s=%.3lf\n", s);/\*если обе точки лежат в второй четверти, находим расстояние между ними\*/

}

else

if (x1<=0 && y1<=0 && x2<=0 && y2<=0)

{

printf("points belong 3 quater\n");

s=sqrt((x2-x1)\*(x2-x1)+(y2-y1)\*(y2-y1));

printf("s=%.3lf\n", s);/\*если обе точки лежат в третьей четверти, находим расстояние между ними\*/

}

else

if (x1>=0 && y1<=0 && x2>=0 && y2<=0)

{

printf("points belong 4 quater\n");

s=sqrt((x2-x1)\*(x2-x1)+(y2-y1)\*(y2-y1));

printf("s=%.3lf\n", s);/\*если обе точки лежат в четвертой четверти, находим расстояние между ними\*/

}/\*если точки лежат в разных четвертях, найдем расстояние от каждой точки до центра\*/

else

{

a=sqrt(x1\*x1+y1\*y1);

b=sqrt(x2\*x2+y2\*y2);

if (a>b)/\*сравним расстояния от точек до центра и найдем максимальное\*/

{

max=a;/\*если расстояние от 1 точки до центра больше растояния второй до центра, max=a(расстояние от 1 точки до центра)\*/

printf("max=%.3lf\n", max);

}

else

{

max=b;/\*если расстояние от первой точки до центра меньлше расстояния от 2 точки до центра, max=b(расстояние от 2 точки до центра) \*/

printf("max=%.3lf\n", max);

}

}

}

return 0;

}

1. Вычислить a(a-n)(a-2n)…(a-n^2). Вещественное, а и натуральное n вводятся с клавиатуры. Использовать управляющую инструкцию *for*.

*Исходные данные:* вещественное число переменная a тип double, увеличивающиеся переменная n тип char.

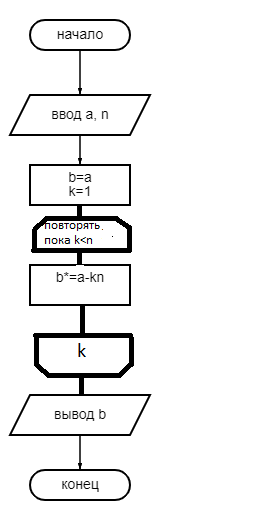
*Результирующие данные:* Переменная b, которая будет увеличиваться и в конченом итоге будет равна значению выражения тип double.

*Вспомогательные переменные:* Переменная k счётчик, чтобы увеличивать n пока n^2=n\*k

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
| a=3 n=2 | -3 | -3.000 |
| a=3 n=4 | 1755 | 1755.000 |
| a=15 n=6 | 3444525 | 3444525.000 |

Схема программы:



Текст программы

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int main()

{

double a, b;

unsigned n, k;

printf("Input a=");

scanf("%lf", &a);/\*вводим a \*/

printf("input n=");

scanf("%d",&n);/\*вводим n \*/

if (n>0)/\*проверяем натуральное ли число n\*/

{

b=a;/\*присваиваем b изначальное значение a,чтобы

записывать в нее результат выражения\*/

k=1;/\*присваиваем переменной k=1, для счетчика \*/

while (k<=n)/\*пока k не > n увеличиваем на каждом проходе k на 1, для подсчета выражения\*/

{

b\*=a-k\*n;/\*увеличиваем значение выражения пока k не стало больше n

например(n=3, на первом проходе цикла мы вычитаем kn=n на втором вычитаем kn=2n на третьем вычитаем kn=3n=n^2)\*/

k++;/\*увеличиваем количество вычитаемых n \*/

}

printf("b=%.3lf", b);/\*выводим резльтат выражения\*/

}

else

printf("error");

return 0;

}

1. Вычислить, за сколько лет в банке при начальном вкладе W и процентах годового прироста Pr будет накоплена сумма Sum (проценты капитализируются ежегодно).

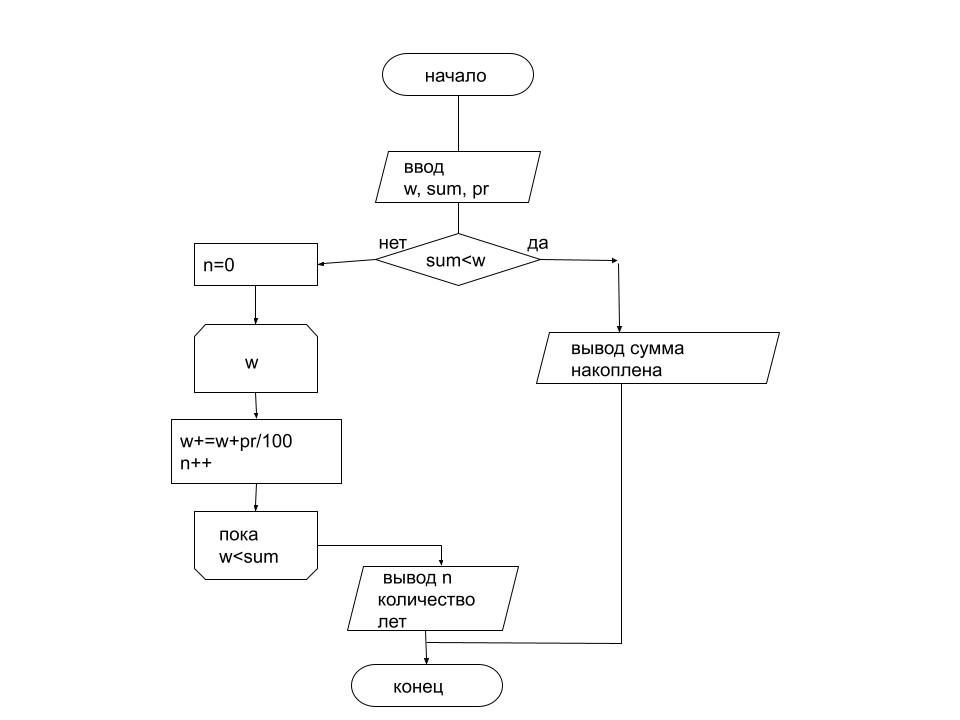
*Исходные данные:* w сумма которая планируется положить в банк для накопления тип double, pr процентная ставка в банке на которую будет увеличиваться изначальная сумма тип double, sum сумму которую необходимо накопить за n число лет тип double.

*Результирующие данные:* n количество лет за которое будет накоплена необходимая сумма тип unsigned int.

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
| w=1,2 pr=2,5 s=1,5 | 10 | 10 |
| w=5 pr=30 s=10 | 3 | 3 |
| w=101,1 pr=60 s=101 | Сумма накоплена | Сумма накоплена |

Схема программы:



Текст программы

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int main()

{

double w, pr, sum;

unsigned int n=0;

printf("initial deposit w=");

scanf("%lf", &w);

printf("percentage of annual growth pr=");

scanf("%lf", &pr);

printf("amount to be accumulated sum=");

scanf("%lf", &sum);/\*ввод исходных данных\*/

if (sum<w)/\* проверяем не накоплена ли уже сумма\*/

printf("accumulated amount");

else

{

do /\*увеличиваем исходную сумму, каждый раз увеличивая n(счетчик лет), до тех пор пока не преодалем необходимую для накопления сумму \*/

{

w+=w\*pr/100;/\*сумма исходного значения + увелечения по процентной ставке за год=получаем сумму в конце года\*/

n++;

}

while (w<sum);/\*увеличиваем исходное значение пока не преодолем sum \*/

printf("number of years to accumulate n=%d\n", n);/\*количество лет за которое будет накоплена нужная сумма\*/

}

system("pause");

return 0;

}

1. Определить, содержится ли в десятичной записи натурального числа N цифра 3.

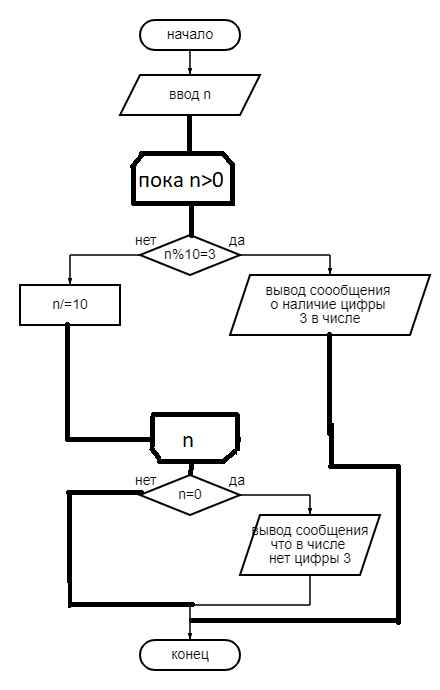
*Исходные данные:* вводим натуральное десятичное число n тип unsigned int

*Результирующие данные:* сообщение о наличие цифры 3 в числе n

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
| n=5127821 | Сообщение о том, что в числе нет цифры 3 | number does not contain 3 |
| n=31889290 | Сообщение о том, что в числе есть цифра 3 | number contains 3 |
| n=452364128364647 | Сообщение о том, что в числе есть цифра 3 | number contains 3 |

Схема программы:



Текст программы

#include <stdio.h>

#include <stdlib.h>

int main()

{

unsigned int n;

printf("input n=");

scanf("%d", &n);/\* ввод числа n\*/

while (n)/\*проверяем наличие в числе 3 путем сравнивания последней цифры с 3

если при деление числа на 10 с остатком, остаток равен 3, то останавливаем программу, иначе

уменьшаем число убирая последнюю цифру деля число на 10 целочисленно, повторяя даннный алгоритм пока не найдем цифру 3 в числе

либо число не станет равно 0\*/

{

if (n%10==3)/\*сравниваем последнюю цифру числа с 3\*/

{

printf("number contains 3");

break;/\*прекращаем работу программу, так как нашли 3 \*/

}

else

n/=10;/\*убираем последнюю цифру в числе, если она не 3, так как ее мы проверили \*/

}

if (n==0)/\*проверяем чему после цикла равно n, если цикл завершился и в числе не нашлась цифра 3 выводим

сообщение о том что в числе нет цифры 3 \*/

printf("number does not contain 3");

return 0;

}

1. Вычислить значение суммы бесконечного ряда с заданной точностью ε=10-5 , и значение функции (для проверки) . Использовать рекуррентные зависимости для вычисления значений слагаемых.

*Исходные данные:* Тип аргумента в задании не указан, поэтому переменная х будет типа 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:*

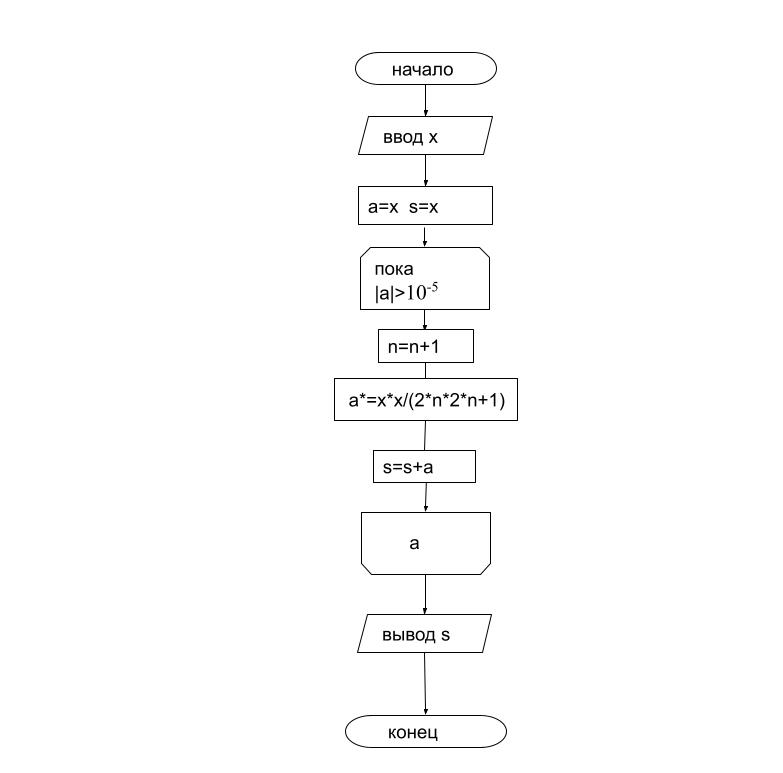
n-ное слагаемое  *,*

предшествующее слагаемое

коэффициент пропорциональности

индекс первого слагаемого в рекуррентной последовательности n=0, первое слагаемое*.*

Схема программы:



Текст программы:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define EPS 1e-5

int main()

{

double x, a, s, p;

int n=0;/\*индекс первого слагаемого\*/

printf("x=");

scanf("%lf",&x);

a=x;

s=x;

/\*пока слагаемое по модулю больше заданной точности \*/

while (fabs(a)>EPS)

{

n++;/\*увеличиваем индекс слагаемое \*/

a\*=x\*x/(2\*n\*(2\*n+1));/\*вычисляем очередное слагаемое домножением

предыдущего на коэффицент пропорциональности\*/

s+=a;/\*накапливаем суммму \*/

}

p=(exp(x)-exp(-x))/2;/\*проверяем результат работы с функцией для проверки\*/

printf("s=%lf\nsh(%lf)=%lf\n", s, x, p);

system("pause");

return 0;

}

*Результаты тестирования*

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Результат работы программы** |
| 0 | *0* | 0.000000 |
| 1 | 1.1752 | 1.175201 |

1. . Вычислить значение суммы бесконечного ряда с заданной точностью ε = , , и значение функции (для проверки), учесть, что 0,1 ≤ x ≤ 1. Использовать рекуррентные зависимости для вычисления значений слагаемых по частям.

*Исходные данные:* Тип аргумента в задании не указан, поэтому переменная х будет типа 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:*

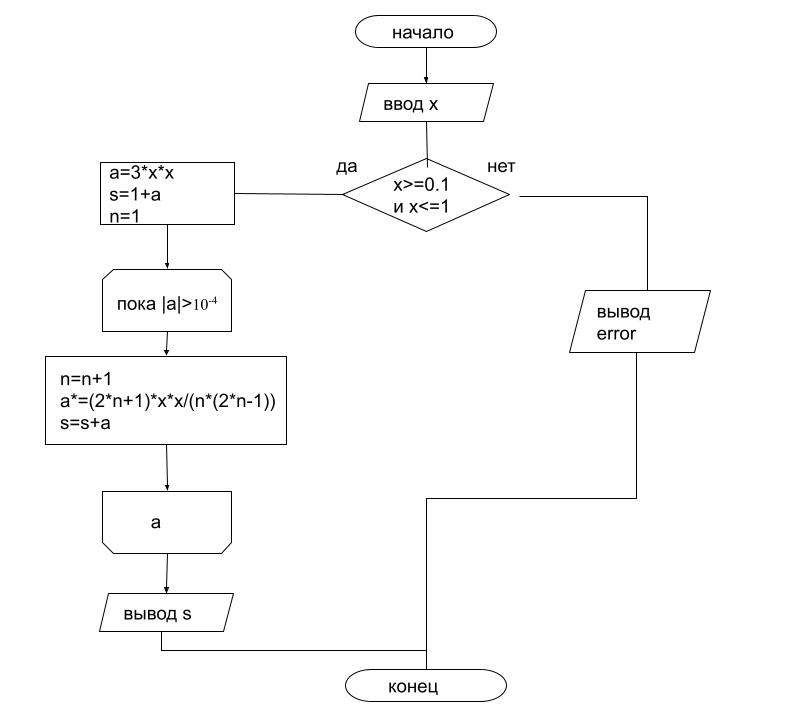
n-ное слагаемое ,

предшествующее слагаемое

коэффициент пропорциональности

индекс первого слагаемого в рекуррентной последовательности n=1, первое слагаемое*.*

Схема программы:



Текст программы

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#define EPS 1e-4

int main()

{

double a, s, x;

int n=1;

printf("x=");

scanf("%lf", &x);

if (x>=0.1 && x<=1)/\*проверяем удовлетворяет ли x поставленному условию

,если x принадлежит [0,1;1] то считаем сумму, иначе выводим ошибку \*/

{

a=3\*x\*x;/\*значение a при n=1 \*/

s=1+a;/\*сумма с а при n=1\*/

while (fabs(a)>EPS)/\*пока слагаемое по модулю больше заданной точности\*/

{

n++;/\*увеличиваем индекс слагаемого \*/

a\*=(2\*n+1)\*x\*x/(n\*(2\*n-1));/\*вычисляем очередное слагаемое домножением

предыдущегона коэфицент пропорциональности\*/

s+=a;/\*накапливаем сумму \*/

}

printf("s=%.4lf\n y(%lf)=%.4lf\n", s, x, (1+2\*x\*x)\*exp(x\*x));

system("pause");

}

else/\*если x не принадлежит нужному отрезку, выводим ошибку \*/

printf("error\n");

system("pause");

return 0;

}

*Результаты тестирования*

|  |  |  |
| --- | --- | --- |
| **Исходные данные** | **Ожидаемый результат** | **Результат работы программы** |
| x=0,1 | 1.03025 | 1.0303 |
| x=0 | error | error |
| x=0,6 | 2.46533 | 2.4653 |

1. Последовательно вводятся n натуральных чисел (n≤10). Вычислить сумму тех из них, у которых первая цифра равна последней.

*Исходные данные:* n-количество натуральных чисел, которое будет введено для подсчета суммы, тип unsigned char, a-число, которое будет вводиться с клавиатуры и проходить проверку для прибавления к сумме.

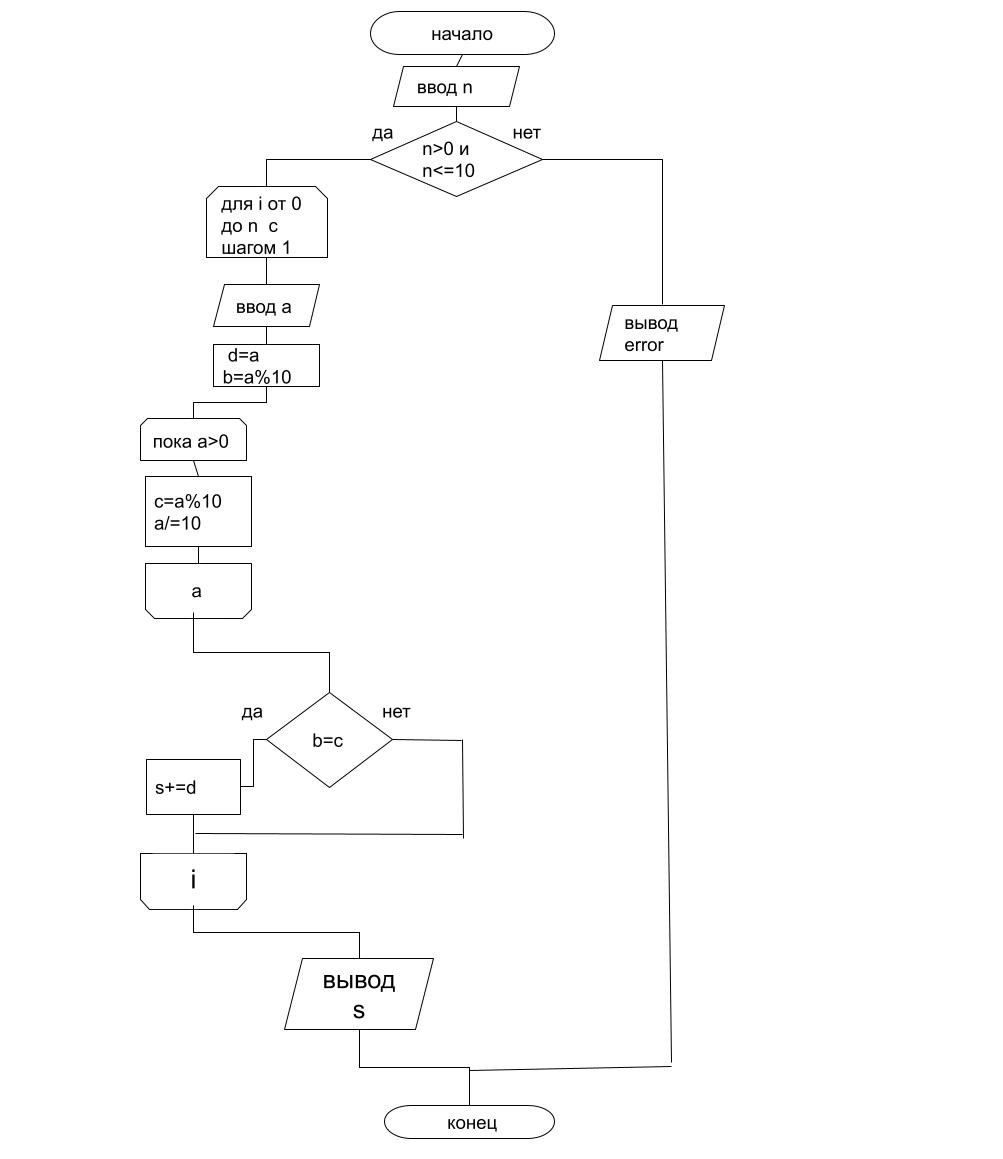
*Результирующие данные:* s-сумма чисел у которых первая и последняя цифра одинаковые.

*Вспомогательные переменные:* d- переменная, чтоб запоминать значение переменной **а** до прохождения проверки равенства первой и последней цифры, и для прибавления к сумме тип unsigned int, b-переменная, чтоб запоминать последнюю цифру числа a тип unsigned int, c- переменная, чтоб запоминать первую цифру числа a тип unsigned int, i-счетчик для записи n натуральных чисел тип unsigned char.

Таблица тестирования:

|  |  |  |
| --- | --- | --- |
| **Входные данные** | **Ожидаемый результат** | **Результат работы программы** |
| n=3 a1=0 a2=434 a3=1 | 435 | 435 |
| n=-3 | error | error |
| n=4 a1=121 a2=34 a3=51 a4=22 | 143 | 143 |

Схема программы:



Текст программы

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int main()

{

int a, b, c, s, d;

unsigned char n, i;

printf("input n<=10 n=");

scanf("%d", &n);/\*ввод количества натуральных чисел\*/

if (n>0 && n<=10)/\*проверяем натуральное ли число было введено и уовлетворяет ли оно условию задачи n<=10\*/

{

s=0;/\*обнуляем переменную для накопления суммы\*/

for(i==1;i<n;i++)/\*ввод чисел для вычисления суммы чисел, в которых первая и последняя цифра одинаковые\*/

{

printf("a=");

scanf("%d", &a);

d=a;/\*вводим число, в котором будем сравнивать первую и последнюю цифру\*/

b=a%10;/\*запоминаем в переменную b последнюю цифру\*/

while (a)/\*с помощью цикла запоминаем в переменную с первую цифру(запоминаем последнюю цифру и делим на 10 целочисленно

до тех пор пока не останется только первая цифра, которую мы запомним делением на 10 с остатком \*/

{

c=a%10;/\*запоминаем последнюю цифру \*/

a/=10;/\*убираем последнюю цифру \*/

}

if (b==c)/\*если первая и последняя цифра равны, то прибавляем к сумме это число\*/

s+=d;/\*накапливаем сумму \*/

}

printf("s=%d\n", s);

system("pause");

return 0;

}

else/\*если введенное n<=0 или n>10\*/

printf("error\n");

system("pause");

return 0;

}