

Итерационные циклические вычислительные процессы с управлением по функции.

Цель работы: разработать и научиться использовать алгоритмы, основанные на циклических вычислительных процессах, управление которыми осуществляется по функции.

Используемое оборудование: ПК.

Задание №1

Постановка задачи: Вычислить 2 в степени n и при этом определить первое значение степени, при котором результат будет превышать значение 1000. Для данной задачи написать максимально возможное количество вариантов программ, используя разные виды циклов.

Математическая модель:

$$2^n = 2_1 * 2_2 * ... * 2_n$$

Блок схема:

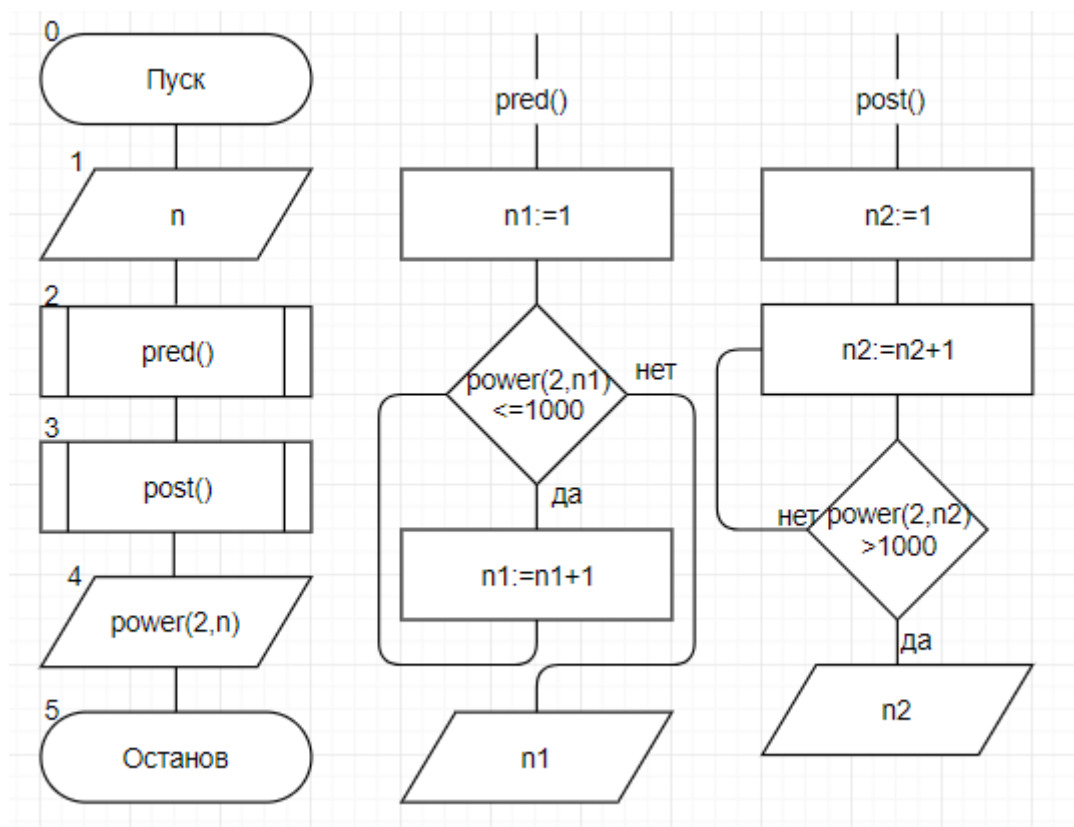


Рис 1 – блок схема к 1 задаче

Список идентификаторов:

Таблица 1 – список идентификаторов для 1 задачи

Название	Смысл	Тип
N1	Показатель степени для цикла с предусловием	Integer
N2	Показатель степени для цикла с постусловием	Integer
N	Вводимая с клавиатуры степень для возведение 2 в нее	Integer

Код программы:

```
program mda;
var n: integer;

procedure pred();
var n1: integer;
begin
    n1:=1;
    while power(2,n1)<=1000 do
        n1:=n1+1;
    writeln('В цикле с предусловием получилось: ',n1);
end;

procedure post();
var n2: integer;
begin
    n2:=1;
    repeat
        n2:=n2+1;
    until power(2,n2)>1000;
    writeln('В цикле с постусловием получилось: ',n2);
end;

begin
    readln(n);
    pred();
    post();
    writeln(power(2,n));
end.
```

Результаты выполненной работы:

```
4
В цикле с предусловием получилось: 10
В цикле с постусловием получилось: 10
16
```

Рис 2 – результат выполнения 1 программы

Анализ результатов вычисления: Для использования циклов с пред и постусловиями были созданы 2 процедуры в программе. Существует вариант с использованием функций и выводом результатов в теле программы, но вариант с процедурами выглядит предпочтительней благодаря независимому выводу. Также может появиться мысль о нерациональности использования

функции power из-за вызова дополнительной подпрограммы из дополнительной библиотеки, нами же было принято решение написать программу именно таким способом потому, что введение дополнительных переменных повлечет увеличение использования памяти, а также такое написание кода облегчает его чтение и понимание.

Задание №2

Постановка задачи: С клавиатуры вводится трехзначное число, считается сумма его цифр. Если сумма цифр числа больше 10, то вводится следующее трехзначное число, если сумма меньше либо равна 10 – программа завершается.

Математическая модель:

Сумма цифр числа = 1 цифра + 2 цифра + 3 цифра

Блок схема:

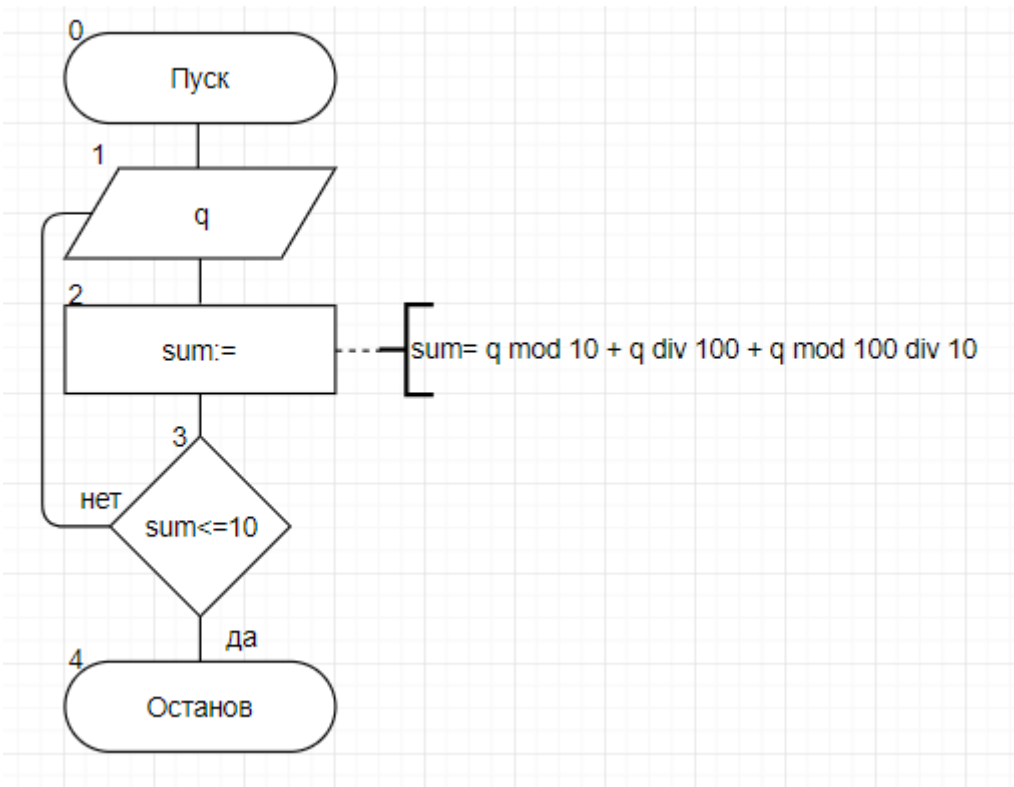


Рис 3 – блок схема к 2 задаче

Список идентификаторов:

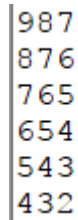
Таблица 2 – список идентификаторов для 2 задачи

Название	Смысл	Тип
Q	Вводимое число, аргумент вызываемой функции	Integer
Sum	Сумма цифр числа для проверки условием	Integer

Код программы:

```
program mda;  
var q, sum: integer;  
  
begin  
  repeat  
    readln(q);  
    sum:=q mod 10+q div 100+q mod 100 div 10;  
  until sum<=10;  
end.
```

Результаты выполненной работы:



```
987  
876  
765  
654  
543  
432
```

Рис 4 – результат работы 2 программы

Анализ результатов вычисления: В теле программы был организован ввод числа, для которого будет рассчитана сумма цифр, в отдельной функции организован расчет суммы цифр. Это позволяет быстро и легко рассчитать сумму цифр для каждого введенного числа и условием определить, нужно ли оставаться в цикле, либо нужно выйти из него.

Задание №3

Постановка задачи: Решить нелинейное уравнение методом Ньютона.

Математическая модель:

$\ln(x) = \sin(x)$ на отрезке от 1 до 3 с
точностью 10^{-6}

Блок схема:

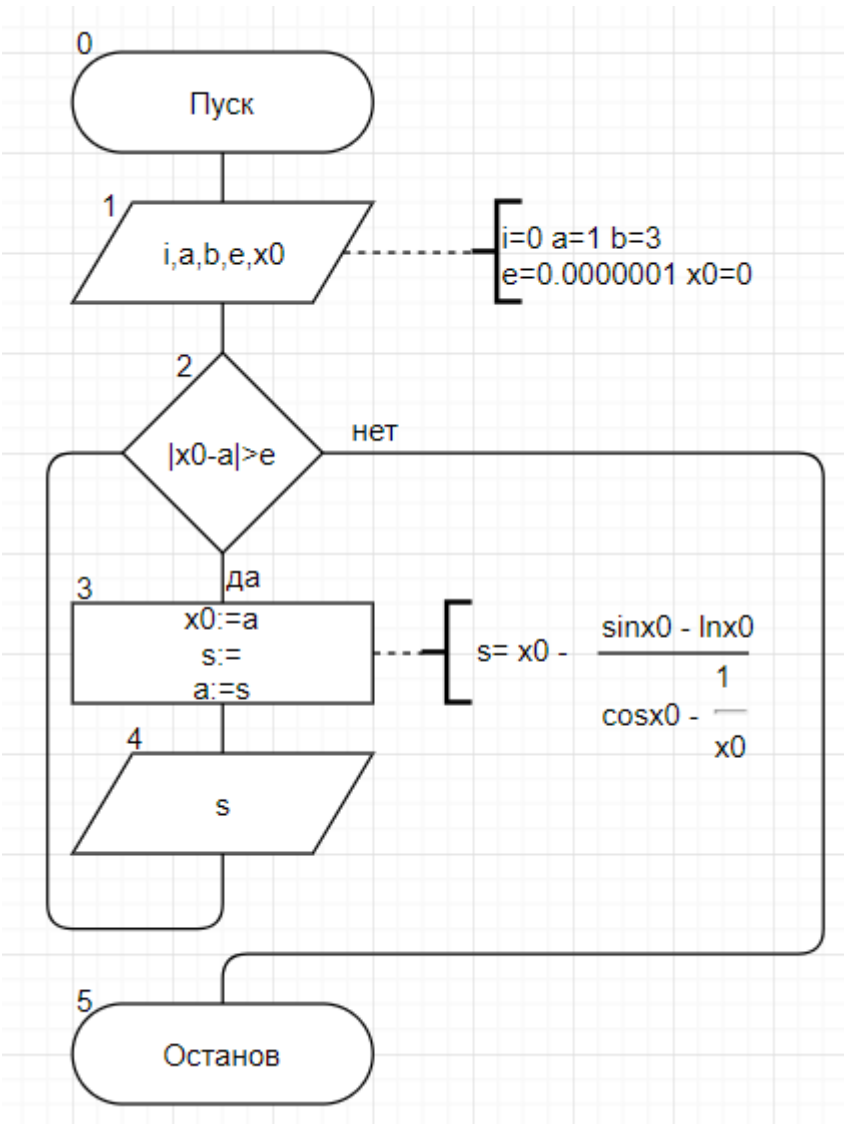


Рис 5 – блок схема к 3 задаче

Список идентификаторов:

Таблица 3 – список идентификаторов к 3 задаче

Наименование	Смысл	Тип
A	Левая граница для расчета	Real
B	Правая граница для расчета	Real
E	Точность рассчитываемой точки	Real
X0	Рассчитываемая точка	Real
S	Результат рассчитываемой точки	Real
I	Индекс рассчитываемой точки	Integer

Код программы:

```
program mda;
```

```

var a,b,e,x0,s:real;
i:integer;

begin
  i:=0;
  a:=1;
  b:=3;
  e:=0.0000001;
  x0:=0;
  while abs(x0-a)>e do
  begin
    x0:=a;
    s:=x0-((sin(x0)-ln(x0))/(cos(x0)-1/x0));
    a:=s;
    writeln('x',i,' = ',s);
    i:=i+1;
  end;
end.

```

Результаты выполненной работы:

```

x0 = 2.83048772171245
x1 = 2.26790221121112
x2 = 2.21974445251704
x3 = 2.21910726324201
x4 = 2.21910714891375
x5 = 2.21910714891375

```

Рис 6 – результат работы 3 программы

Анализ результатов вычисления: В программе был реализован метод Ньютона для решения нелинейного уравнения. Цикл продолжается пока разница между последним рассчитываемым значением и предпоследним больше заданной точности.

Вывод: Все поставленные задачи были выполнены. Все алгоритмы соответствуют необходимым требованиям. Были использованы циклы с предусловием и с постусловием. Также были использованы умения по разработке процедур и функций, развитые в прошлых лабораторных работах.