Лабораорная работа №7. Итерационные ​​циклические​​ вычислительные ​​процессы​​ с управлением​​ по​​ функции.

Цель л/р: рассмотрение методов реализации ИЦВП с управлением по функции, решение задач при помощи данного вида ИЦВП средствами Free Pascal и Lazarus.

Оборудование: ПК, Word, Lazarus, Draw.io.

Задание 1.1

1. 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. Например:  
   1. цикл с постусловием  
   2. цикл с предусловием
2. Математическая модель:
3. Блок-схема:

1. Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| n | Вводимое пользователем значение, показатель степени двойки | byte |
| n2 | Степень двойки | longword |
| par | Условие выхода из цикла | real |

1. Код программы:

program project1;

var n:byte;

n2:longword;

par:real;

begin

writeln('Enter n');

readln(n);

par:=exp(n\*ln(2));

n2:=2;

while n2<=1000 do

n2:=n2\*2;

writeln('First n value for 2^n > 1000 is ',ln(n2)/ln(2):2:0);

n2:=1;

while n2<par do

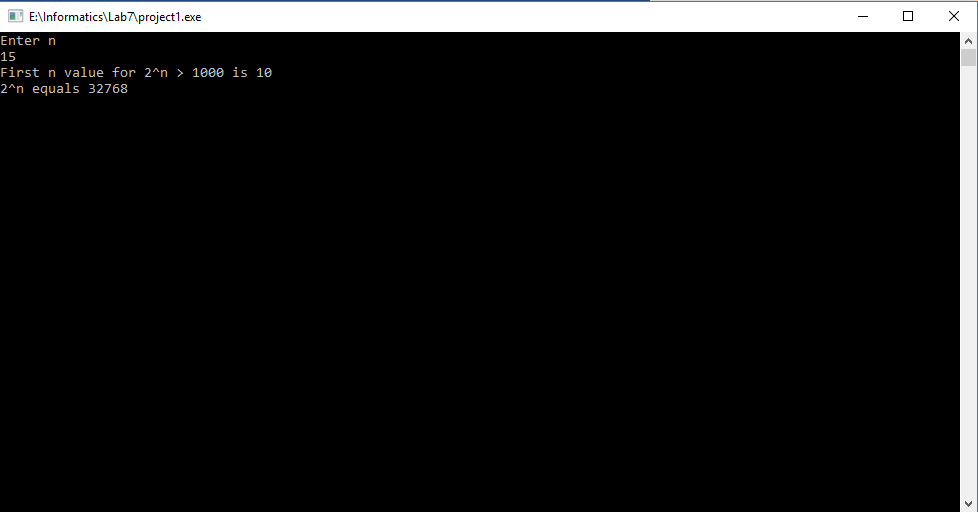
n2:=n2\*2;

writeln('2^n equals ',n2);

readln();

end.

1. Результат выполненной работы:



1. Анализ результатов вычисления:

В первом цикле программы вычисляется первый показатель степени n, при котором 2^n превышает 1000. Второй цикл расчитывает значение 2^n, где n – пользовательское значение. Программа выдаёт корректный результат при n ∈ [0;30], в противном случае происходит переполнение переменной n2.

Задание 1.2

3. Блок-схема:

5. Код программы:

program project1a;

var n:byte;

n2:longword;

par:real;

begin

writeln('Enter n');

readln(n);

par:=exp(n\*ln(2));

n2:=2;

repeat

n2:=n2\*2;

until n2>=1000;

writeln('First n value for n^2 > 1000 is ', ln(n2)/ln(2):0:0);

n2:=1;

repeat

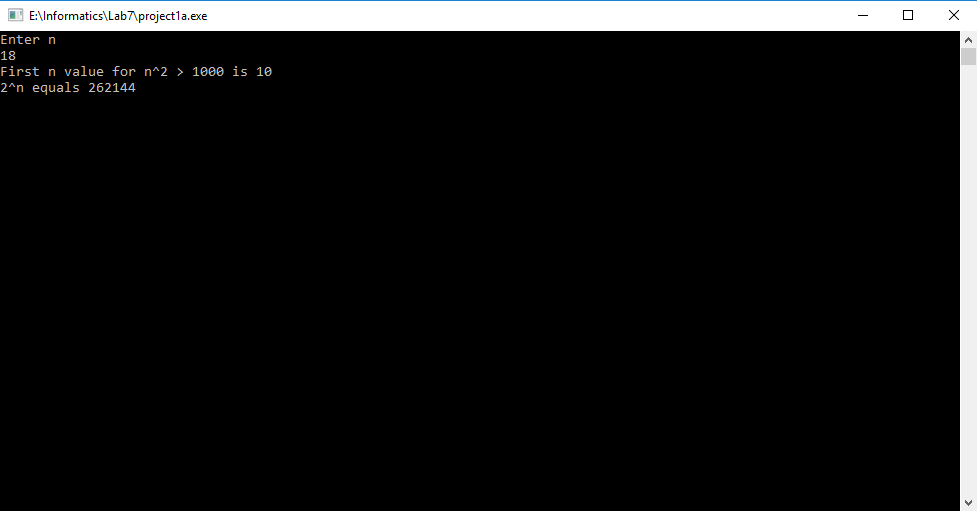
n2:=n2\*2;

until n2 >= par;

writeln('2^n equals ',n2);

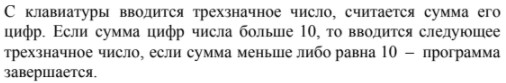
readln;

end.

6. Результаты выполненной работы:

7. Анализ результатов вычисления:

Данный вариант программы практически индентичен предыдущему, отличается лишь использованием циклов с постусловием.

Задание 2.

2. Математическая модель:
3. Блок-схема:



1. Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| x | Вводимое пользователем трёхзначное число | integer |
| digsum | Сумма цифр, стоящих в разрядах числа | shortint |

1. Код программы:

program project2;

var x:integer;

digsum:shortint;

begin

repeat

writeln('Enter the number');

readln(x);

digsum:=0;

while x<>0 do begin

digsum:=digsum+x mod 10;

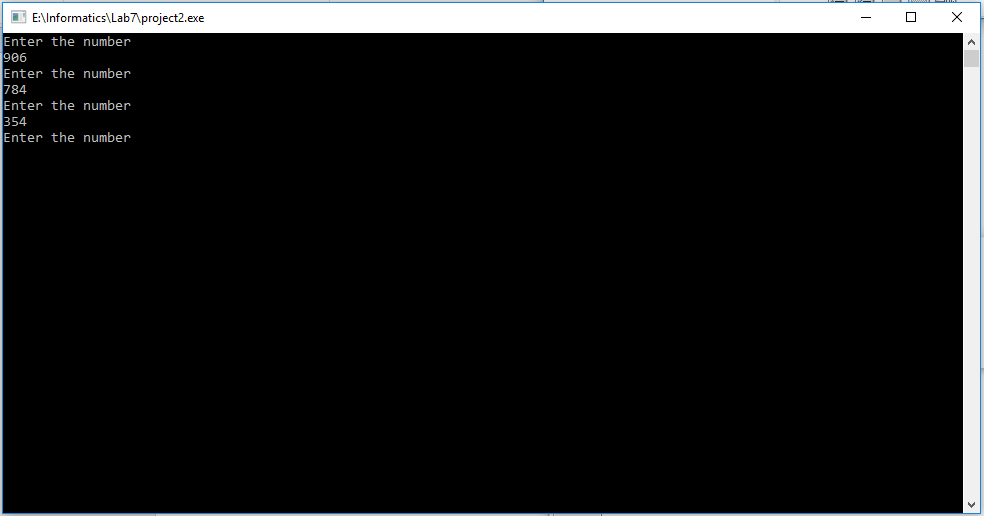
x:=x div 10;

end;

until abs(digsum)<=10;

end.

1. Результаты выполненной работы:



1. Анализ результатов вычисления:

Тело программы охватывает цикл repeat…until, внутри которого происходит ввод пользователем трёхзначного числа, последовательное его разбиение на разряды и подсчёт суммы их значений. Допустим ввод отрицательных значений, т.к. в операторе проверки условия until используется модуль суммы цифр числа. При вводе числа, сумма цифр которого меньше либо равна 10 программа закрывается.

Задание 3.

1. 



1. Математическая модель:
2. Блок-схема:



1. Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| a | Левая граница интервала изоляции | real |
| b | Правая граница интервала изоляции | real |
| eps | Точность вычисления, abs(x\_next-x\_cur) | real |
| x\_cur | Текущее значение x | real |
| x\_next | Следующее значение x | real |

1. Код программы:

program Project3;

var eps, a, b, x\_cur, x\_next:real;

begin

eps:=1e-6;

a:=-10;

b:=10;

x\_next:=a;

repeat

x\_cur:=x\_next;

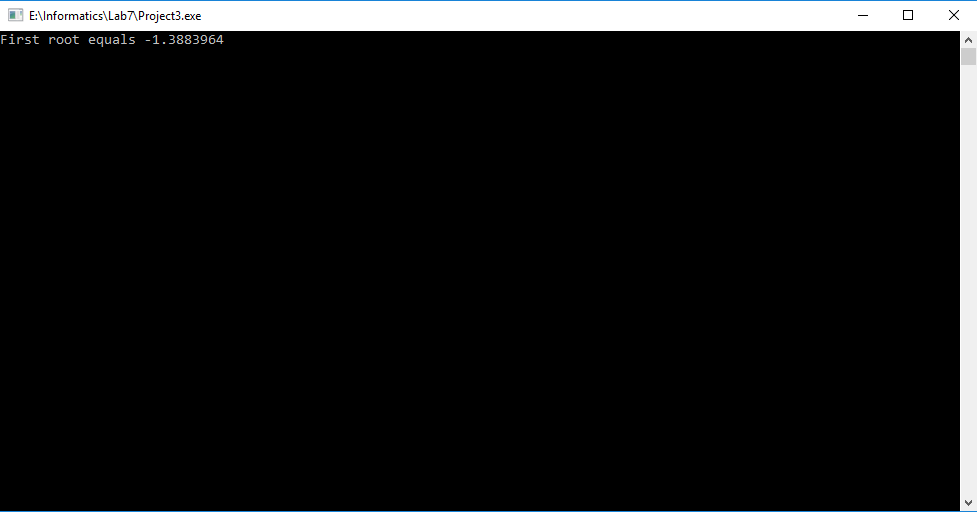
x\_next:=x\_cur-(((((x\_cur-4)\*x\_cur)-8)\*x\_cur\*x\_cur+1))/((((x\_cur-3)\*x\_cur)-4)\*x\_cur);

until abs(x\_next-x\_cur)<eps;

writeln('First root equals ',x\_next:0:7);

readln();

end.

1. Результаты выполненной работы:
2. Анализ результатов вычисления:

Тело программы состоит из ввода заданных параметров(границы интервала изоляции, точность вычисления) и цикла repeat…until, в котором вычисляется каждое последующее приближенное значение корня, до тех пор, пока результат не будет соответствовать установленному требованию . После выполнения условия на экран выводится полученное значение с точностью в 7 десятичных знаков.

Вывод:

Итерационные циклические вычислительные процессы просты в организации, не требуют выделения дополнительной переменной, используются главным образом для нахождения значения функции методом последовательных приближений. В языке Free Pascal для реализации данного вида ИЦВП используются конструкции while...do

и repeat…until.