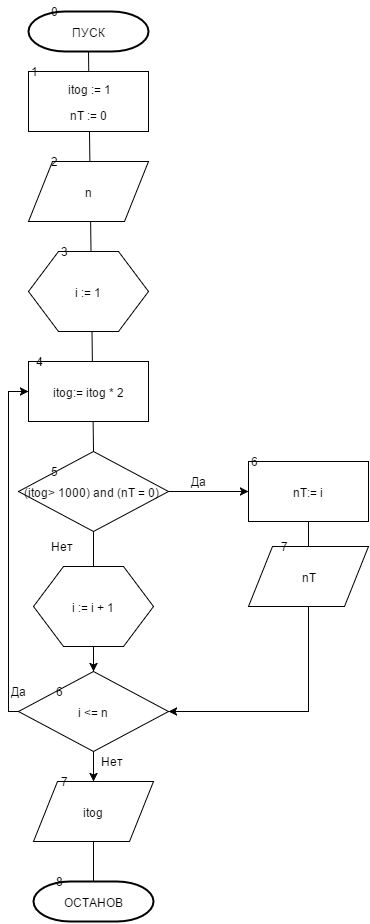
Лабораторная работа №7.

Задание 1.

1. Тема: итерационные циклические вычислительные процессы с управлением по функции.
2. Цель: научиться реализовывать итерационные циклические вычислительные процессы с управлением по функции средствами FreePascal.
3. Используемое оборудование: ПК, среда программирования Lazarus.
4. Постановка задачи: вычислить 2 в степени n и при этом определить первое значение степени, при котором результат будет превышать значение 1000.
5. Математическая модель:

Просто умножаем двойку саму на себя такое кол-во раз, какое нам нужно.

1. Блок-схема:



1. Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| n | integer | степень |
| i | integer | счетчик |
| nT | integer | Логическая проверка и степень, в которую нужно возвести, чтобы получить больше 1000 |
| itog | real | 2 в степени n |
| k | integer | Проверка во второй программе |

1. Код программы:

**1-ый способ** (более адекватный):

program lr7\_task1;

var

n,i,nT : integer;

itog : real;

begin

itog := 1;

nT := 0;

writeln('Vvedite N: '); readln(n);

for i := 1 to n do begin

itog := itog \* 2;

if (itog > 1000) and (nT = 0) then begin

nT := i;

writeln('Pervoe Znachenie N, pri kotorom Itog > 1000 = ',nT);

end;

end;

writeln('Resultat Raven : ',itog:2:2);

readln();

end.

**2-ой способ:**

program lr7\_task1\_2;

var

n, k,i:integer;

itog:real;

begin

k:=1;

i:=0;

writeln('Vvedite stepen'); readln(n);

repeat

k:=2\*k;

i:=i+1;

if k>1000 then begin

writeln('Stepen pri kotoroy k>1000 = ', i); break;

end;

until i>n;

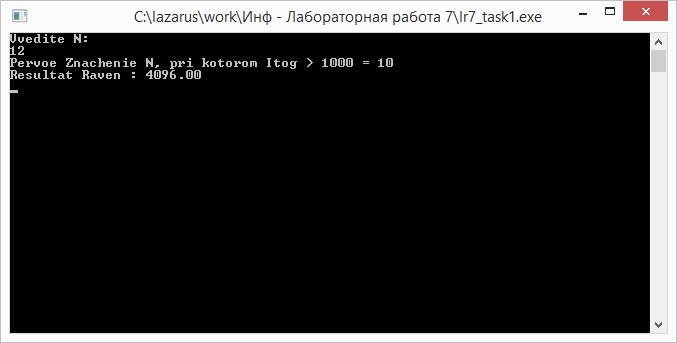
itog:=exp(n\*ln(2));

writeln(' itog = ',itog:2:0);

readln()

end.

1. Результат работы программы:



1. Анализ результатов вычисления:

В цикле с параметром, мы умножаем переменную, изначально равную 1, на 2. Потом при помощи доп. Переменной проверяем, какое первое значение n, чтобы 2 в степени n было больше 1000.

1. Вывод: программа работает корректно.

Задание 2.

1. Тема: итерационные циклические вычислительные процессы с управлением по функции.
2. Цель: научиться реализовывать итерационные циклические вычислительные процессы с управлением по функции средствами FreePascal.
3. Используемое оборудование: ПК, среда программирования Lazarus.
4. Постановка задачи: Решить нелинейное уравнение методом Ньютона на отрезке [-10;10] с точностью 10-6.
5. Математическая модель:

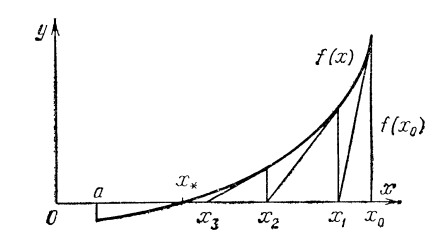
Рассмотрим в точке *x0* касательную к кривой *y=f(x)*, задаваемую уравнением:

решения нелинейных уравнений

Положим *y=0*, находим точку *x1* пересечения касательной с осью абсцисс:

решения нелинейных уравнений

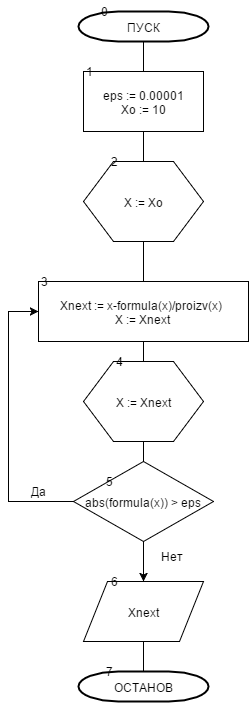
Построив касательную в точке *x1* получаем



по аналогичной формуле точку *x2* пересечения этой касательной с осью *x* и т.д.:

решения нелинейных уравнений

1. Блок-схема:



1. Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| eps | real | точность |
| X | real | Текущее x |
| Xo | real | Первое приближение |
| Xnext | real | Следующее x |
| formula | real | Формула |
| proizv | real | Производная |

1. Код программы:

program lr7\_task2;

var eps, X, Xo, Xnext : real;

function formula(per:real):real;

begin

formula := per\*per\*per-cos(per)+1;

end;

function proizv(per:real):real;

begin

proizv := 3\*per\*per+sin(per);

end;

begin

eps := 0.00001;

Xo := 10;

X := Xo;

repeat

Xnext := x-formula(x)/proizv(x);

X := Xnext;

writeln(Xnext:2:6);

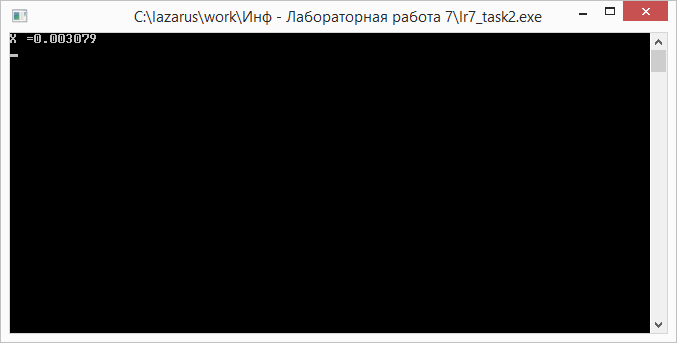
until abs(formula(x)) <= eps;

writeln(Xnext:2:6);

readln;

end.

1. Результат работы программы:



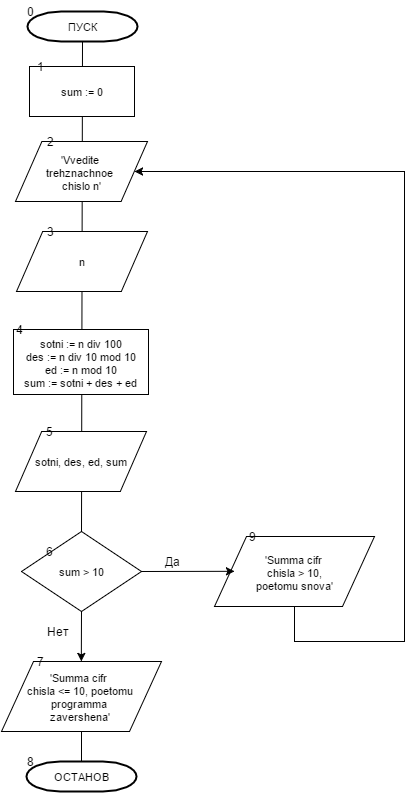
1. Анализ результатов вычисления: задаем первое приближение и точность вычислений. Начинаем цикл с постусловием, в котором проводим вычисления согласно мат.модели.
2. Вывод: программа работает корректно.

Задание 3.

1. Тема: итерационные циклические вычислительные процессы с управлением по функции.
2. Цель: научиться реализовывать итерационные циклические вычислительные процессы с управлением по функции средствами FreePascal.
3. Используемое оборудование: ПК, среда программирования Lazarus.
4. 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.
5. Математическая модель:

разобьём введённое число на разряды – сотни, десятки, единицы. Сложим значения разрядов. Проведём проверку на то, что сумма цифр меньше либо равна 10. Если нет, то необходимо организовать повторный ввод. Если да, то программа завершается.

1. Блок-схема:



1. Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| n | integer | Вводимое число |
| sum | integer | Сумма цифр числа |
| sotni | integer | Значение разряда сотен |
| des | integer | Значение разряда десятков |
| ed | integer | Значение разряда единиц |

1. Код программы:

program lr7\_task3;

var

n,sum, sotni, des, ed : integer;

begin

sum := 0;

repeat

writeln('Vvedite trehznachnoe chislo n: ');

readln(n);

sotni := n div 100;

des := n div 10 mod 10;

ed := n mod 10;

writeln('sotni: ',sotni,' desyatki: ',des,' edinici: ',ed);

sum := sotni + des + ed;

writeln('sum: ',sum);

writeln;

if sum > 10 then

writeln('Summa cifr chisla > 10, poetomu snova');

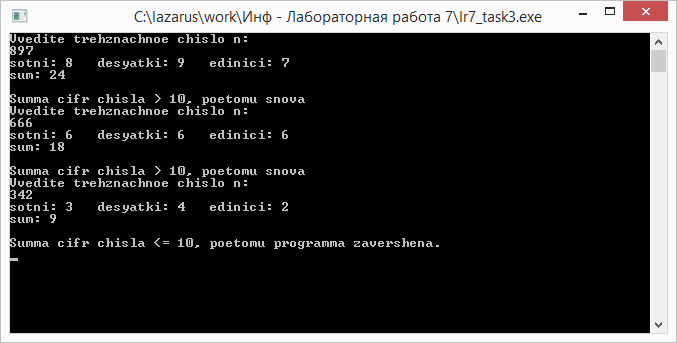
until sum <= 10;

writeln('Summa cifr chisla <= 10, poetomu programma zavershena.');

readln;

end.

1. Результат работы программы:



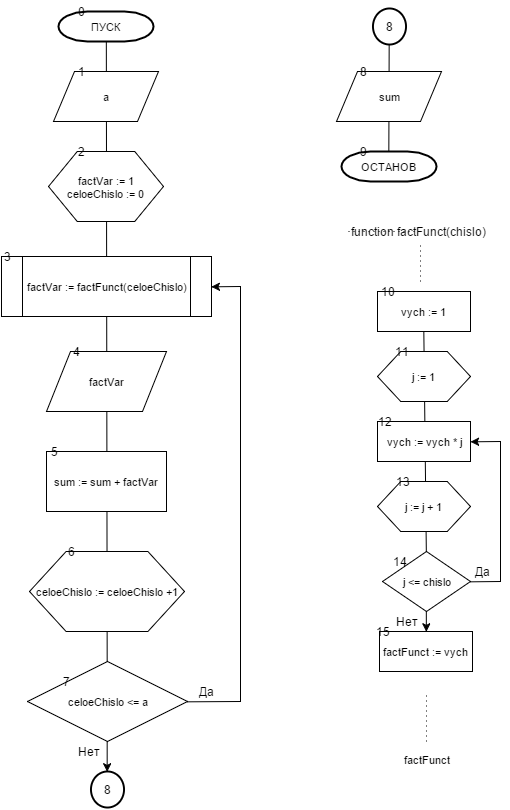
1. Анализ результатов вычисления:

В цикле с постусловием repeat … until, мы каждый раз вводим значение n и обрабатываем его в соответствии с поставленной задачей. Дополнительная проверка if введена, чтобы отобразить комментарий – почему же нам снова приходиться вводить число n.

1. Вывод: мы написали программу, которая получает на вход трехзначное число, считает сумму его цифр. Если сумма цифр числа больше 10, то вводится следующее трехзначное число, если сумма меньше либо равна 10 – программа завершается.

Задание 4.

1. Тема: итерационные циклические вычислительные процессы с управлением по функции.
2. Цель: научиться реализовывать итерационные циклические вычислительные процессы с управлением по функции средствами FreePascal.
3. Используемое оборудование: ПК, среда программирования Lazarus.
4. Постановка задачи: составить программу подсчета суммы факториалов целых чисел, где сумма не превышает число А, которое вводится с клавиатуры. На экран вывести сумму и все слагаемые.
5. Математическая модель:
6. Блок-схема:



1. Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| a | longint | Вводимое число. Его не должна превышать сумма факториалов. |
| celoeChislo | integer | Целые числа, факториалы которых мы вычисляем. |
| sum | real | Сумма факториалов |
| factVar | real | Для обращения к функции и вывода возвращенного значения на экран |

1. Код программы:

program lr7\_task4\_2;

var

a : longint;

celoeChislo : integer;

sum, factVar : real;

function factFunct(chislo:integer):real;

var

vych : real;

j : integer;

begin

vych := 1;

for j := 1 to chislo do

vych := vych \* j;

factFunct := vych;

end;

begin

writeln('Vvedite chislo A, kotorogo ne doljna prevychat summa factorialov: '); readln(a);

sum := 0;

factVar := 1;

celoeChislo := 0;

for celoeChislo := 0 to a do begin

factVar := factFunct(celoeChislo);

writeln ('Slagaemoe ',celoeChislo+1,' - Factorial ',celoeChislo,' raven ',factVar:2:1);

sum := sum + factVar;

if (sum+factFunct(celoeChislo+1)) > a then break;

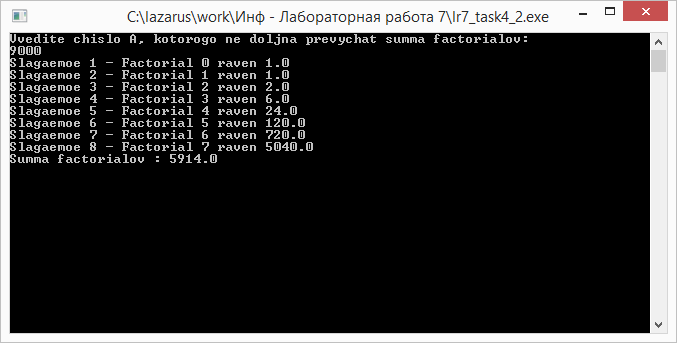
end;

writeln ('Summa factorialov : ',sum:2:1);

readln;

end.

1. Результат работы программы:



1. Анализ результатов вычисления:

В цикле с постусловием repeat … until, мы каждый раз вводим значение n и обрабатываем его в соответствии с поставленной задачей. Дополнительная проверка if введена, чтобы отобразить комментарий – почему же нам снова приходиться вводить число n.

1. Вывод: мы написали программу, которая получает на вход трехзначное число, считает сумму его цифр. Если сумма цифр числа больше 10, то вводится следующее трехзначное число, если сумма меньше либо равна 10 – программа завершается.