Министерство науки и высшего образования Российской Федерации

Федеральное бюджетное образовательное учреждение

высшего образования

**«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ В.Ф. УТКИНА»**

Направление 02.03.01 «Математика и компьютерные науки»

Кафедра «Космические технологии»

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Программирование и алгоритмические языки»

Тема курсового проекта: «Интерполяция математических функций интерполяционной формулой Ньютона»

Вариант 3

Научный руководитель:

Доцент кафедры КТ

Наумов Д.А.

Выполнила:

Студентка 2 курса группы №848

Елисеева Е.В.

*Рязань, 2019*

СОДЕРЖАНИЕ

Оглавление

[**ВВЕДЕНИЕ 3**](#_Toc28022520)

[**1. ОПИСАНИЕ МЕТОДА РЕШЕНИЯ ЗАДАЧИ 4**](#_Toc28022521)

[**1.1. Интерполяция математических формул 4**](#_Toc28022522)

[**1.2. Интерполяционный многочлен в форме Ньютона 4**](#_Toc28022523)

[**1. 2. 1. Интерполяционная формула Ньютона для не равноотстоящих значений аргумента. 5**](#_Toc28022524)

[**1.2.2. Интерполяционная формула Ньютона для равноотстоящих значений аргумента. 5**](#_Toc28022525)

[**3. РАЗРАБОТКА АЛГОРИТМОВ 9**](#_Toc28022526)

[**4. ТЕСТРИОВАНИЕ ПРОГРАММЫ 14**](#_Toc28022527)

[**5. РАЗРАБОТКА ДОКУМЕНТАЦИИ 21**](#_Toc28022528)

[**5.1. Назначение программы 21**](#_Toc28022529)

[**5.2. Работа программы 21**](#_Toc28022530)

[**5.3. Пример некорректного ввода данных 23**](#_Toc28022531)

[**ЗАКЛЮЧЕНИЕ 24**](#_Toc28022532)

[**БИБЛИОГРАФИЧЕСКИЙ СПИСОК 25**](#_Toc28022533)

[**ПРИЛОЖЕНИЕ 1. ЛИСТИНГ ПРОГРАММЫ 26**](#_Toc28022534)

# ВВЕДЕНИЕ

Целью курсового проекта является разработке программной системы для интерполяции математической функции интерполяционным полиномом Ньютона.

Задачами курсового проектирования является:

* изучение метода решения задачи;
* разработка алгоритма и структур данных;
* разработка программы, позволяющей решить задачу интерполирования функции;
* выполнить контрольный расчет в ручном режиме;
* провести тестирование работы программы на контрольном примере;
* выполнить тестирование работы программы на нескольких тестовых примерах;
* разработать документацию к программе.

Пояснительная записка к курсовому проекту состоит из следующих разделов:

* введение;
* описание метода решения задачи;
* разработка структур данных;
* разработка алгоритмов;
* тестирование программы;
* разработка документации;
* заключение;
* библиографический список;
* приложение (листинг программных модулей).

# 1. ОПИСАНИЕ МЕТОДА РЕШЕНИЯ ЗАДАЧИ

## 1.1. Интерполяция математических формул

Задача интерполирования — это задача восстановления функции, которая задана на дискретном множестве точек , i = 0,1,..., n. Для вещественных функций одного аргумента её постановка такова. Задан интервал [a, b] ⊂ R и конечное множество несовпадающих точек ∈ [a, b], i = 0,1,..., n , называемых узлами интерполяции. Совокупность всех узлов — множество {,,...,} — будем называть сеткой. Даны также вещественные числа , i = 0,1,..., n. Требуется построить функцию g(x) от непрерывного аргумента x ∈ [a, b], которая принадлежит заданному классу функций G и в узлах принимает значения , i = 0,1,..., n.

Искомую функцию g(x) называют при этом интерполирующей функцией или интерполянтом. Часто значения , , ..., принимаются в заданных узлах ,,..., некоторой реальной функцией непрерывного аргумента f(x). Как и ранее, требуется построить функцию g(x) от аргумента x ∈ [a, b], которая принадлежит заданному классу функций G и в узлах принимает значения y = f(), i = 0,1,..., n. В этом случае будем говорить, что рассматривается задача интерполяции функции f(x) по узлам ,,...,.  
Практическая значимость задачи интерполяции чрезвычайно велика. Она встречается всюду, где у функции непрерывного аргумента (который может быть временем, пространственной координатой и т.п.) мы имеем возможность наблюдать лишь значения в дискретном множестве точек, но хотим восстановить по ним ход функции на всём множестве значений аргумента.

## 1.2. ****Интерполяционный многочлен в форме Ньютона****

Выборка экспериментальных данных представляет собой массив данных, который характеризует процесс изменения измеряемого сигнала в течение заданного времени (либо относительно другой переменной). Для  выполнения теоретического анализа измеряемого сигнала необходимо найти аппроксимирующую функцию, которая свяжет дискретный набор экспериментальных данных с непрерывной функцией - интерполяционным полиномом n-степени. Данный интерполяционный полином n-степени может быть записан, например, в форме Ньютона (один из способов представления).

**Интерполяционный многочлен в форме Ньютона**– это математическая функция позволяющая записать полином n-степени, который будет соединять все заданные точки из набора значений, полученных опытным путём или методом случайной выборки с постоянным/переменным временным шагом измерений.

### 1. 2. 1. Интерполяционная формула Ньютона для не равноотстоящих значений аргумента.

Обозначим через (x) интерполяционный полином степени k, построенный по узлам , , ..., . В частности, (x) = = f() — интерполяционный полином нулевой степени, построенный по одному узлу . Тогда очевидно следующее тождество

(x) = (x) +

Замечательность этого представления состоит в том, что при добавлении или удалении последних по номеру узлов интерполяции перестройке должны подвергнуться лишь те последние слагаемые суммы из правой части уравнения, которые вовлекает эти изменяемые узлы. Первые слагаемые зависят только от первых узлов интерполяции и останутся неизменными. Таким образом, стоящая перед нами задача окажется решённой, если будут найдены удобные и просто выписываемые выражения для разностей (x) − −1(x). Заметим, что разность (x) − − 1(x) есть полином степени k, который обращается в нуль в узлах , , ..., , общих для (x) и (x), где эти полиномы должны принимать одинаковые значения, , ...,. Поэтому должно быть

(x) − − 1(x) = (x - )(x - )…(x − )

с некоторой константой .

Окончательное представление интерполяционного полинома Ньютона:

(x) = + (, )^ (x − ,) + (, , )^ (x − ,)(x − ,) + ... + (, , ... , )^ (x − )(x − )…(x − ).

Для задачи интерполирования заданной функции f аналогичное выражение для интерполяционного полинома имеет вид

(x)= f() + f^(, )(x − ) + f^(, )(x − )(x − ) + ... + f^(, , ..., )(x − )(x − )...(x − ).

В правой части неравенства стоит интерполяционный полином Ньютона.

**1.2.2. Интерполяционная формула Ньютона для равноотстоящих значений аргумента.**

Важнейший частный случай интерполирования относится к равномерному расположению узлов, когда величина = − , называемая шагом сетки {,,..., }, постоянна и не зависит от i, т.е. = h = const. Тогда вычисление разделённых разностей решительно упрощается, сводясь к оперированию с так называемыми конечными разностями. По определению конечной разностью (иногда добавляют — первого порядка) от функции f в точке x называется величина ∆y = ∆f(x) = f(x + h) − f(x).

Индукцией по порядку разделённых и конечных разностей не трудно показать, что они связаны друг с другом соотношением

f^(, , ... , ) = , k = 1, 2, ...

Как следствие, интерполяционный полином Ньютона для равномерно расположенных узлов принимает вид:

(x) = f() + (x − ) + (x − )(x − ) + (x − )(x − )...(x − ).

**2. РАЗРАБОТКА СТРУКТУР ДАННЫХ**

Типы данных в Паскале определяют возможные значения переменных, констант, выражений и функций. Они бывают встроенными и пользовательскими. Встроенные типы изначально присутствуют в языке программирования, а пользовательские создаются программистом.

Рассмотрим наиболее простые типы данных:

* Целочисленный тип

Сюда входят несколько целочисленных типов, которые различаются диапазоном значений, количеством байт отведённых для их хранения и словом, с помощью которого объявляется тип. Над переменными этой категории можно выполнять все арифметические и логические операции за исключением деления, для него нужен вещественный тип.

|  |  |  |
| --- | --- | --- |
| Тип | Диапазон значений | Размер в байтах |
| Shortint | -128…127 | 1 |
| Integer | -32 768…32 767 | 2 |
| Longint | -2 147 483 648…2 147 483 647 | 4 |
| Byte | 0…255 | 1 |
| Word | 0…65 535 | 2 |

*Таблица 1 – Целочисленный тип данных*

* Вещественный тип

Над ними может быть выполнено большее количество операций и функций, чем над целыми.

|  |  |  |
| --- | --- | --- |
| Тип | Диапазон | Память, байт |
| Real | 2.9e-39 … 1.7e38 | 6 |
| Single | 1.5e-45 … 3.4e38 | 4 |
| Double | 5.0e-324 …1.7e308 | 8 |
| Extended | 3.4e-4932 … 1.1e493 | 10 |
| Comp | -9.2e63 … (9.2e63)-1 | 8 |

*Таблица 2 – Вещественный тип данных*

* Логический тип

Переменная, имеющая логический тип данных может принимать всего два значения: true (истина) и false (ложь). Здесь истине соответствует значение 1, а ложь тождественная нулю. Над данными этого типа могут выполняться операции сравнения и логические операции: not , and, or, xor.

Символьный тип данных – это совокупность символов, используемых в том или ином компьютере. Переменная данного типа принимает значение одного из этих символов, занимает в памяти компьютера 1 байт. Слово **Char** определяет величину данного типа.

Для решения задачи используем целочисленный и вещественный типы данных. Тип данных *Real* выберем потому, что с ним возможны все виды арифметических операций, нужных для данной задачи. Тип данных *Integer* подходит по диапазону значений для того, чтобы обозначить константы.

# 3. РАЗРАБОТКА АЛГОРИТМОВ

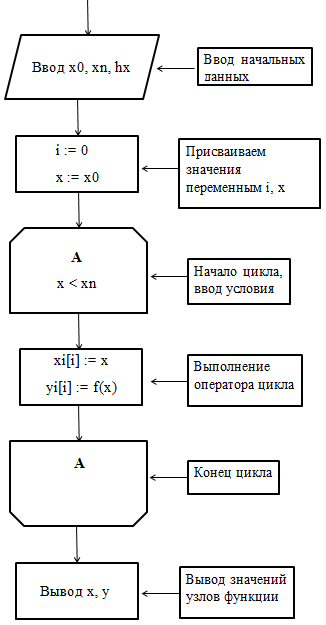
Вычисление полинома Ньютона в данной программе происходит по нескольким возможным вариантам, но все они находятся в теле цикла с постусловием, поэтому за основу алгоритма возьмем схему «Цикл с постусловием», изображенной на Рисунке 1.



*Рисунок 1 – Общая схема цикла с постусловием*

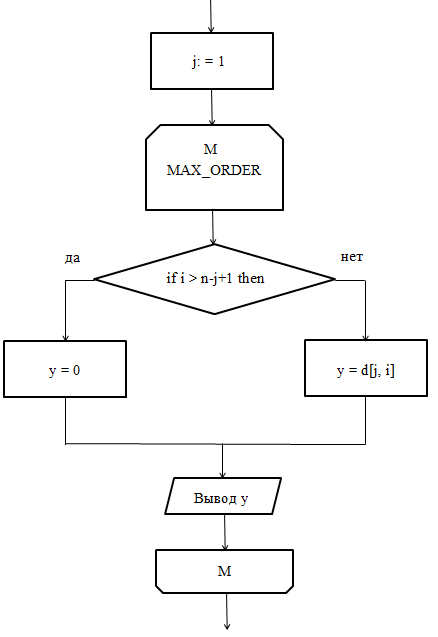
Для вычисления полинома Ньютона нужно ввести значение , , , где

До вычисления непосредственно значения полинома Ньютона, требуется вычислить значения узлов, затем разности первого и j – того порядков. Для этого потребуется схема «Цикл с предусловием», изображенная на Рисунке 2, и схема алгоритма «Условный оператор», которая показана на Рисунке 3.

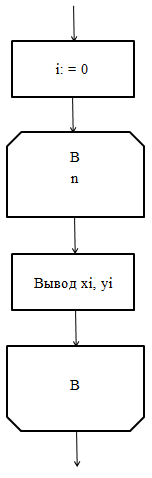


*Рисунок 2 – Схема цикла с предусловием*

После этого, с помощью «Цикла *for»* выводятся заданные значения узлов, теперь можем приступить к нахождению разностей.



*Рисунок 3 – Вычисление j – той разностей*

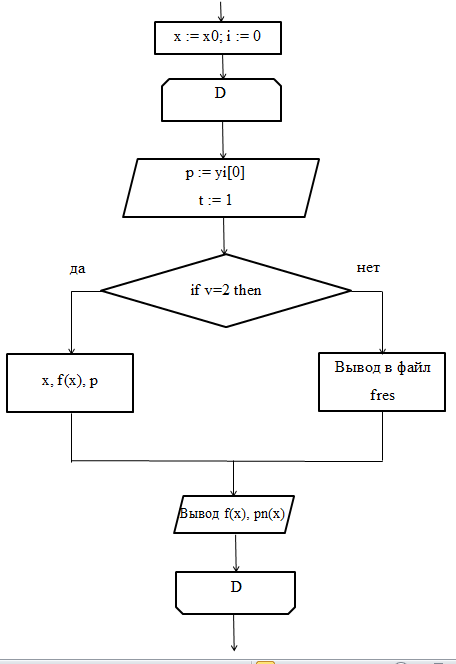


*Рисунок 4 – Схема цикла for*

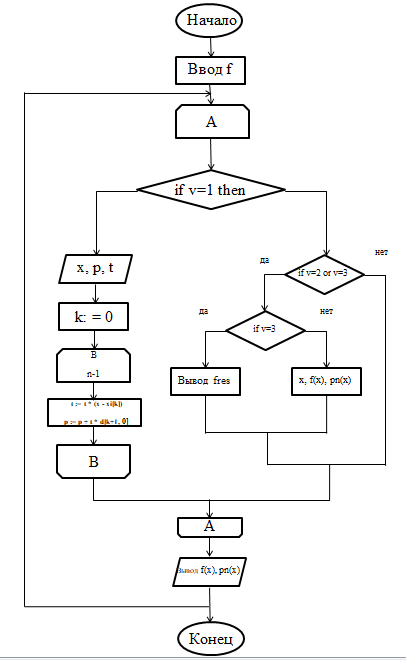
Алгоритм интерполяции функции можно произвести тремя способами:

1. Одиночный расчет
2. Табуирование функции
3. Табуирование функции с выводом результата в файл result.csv

Последовательная схема расчета представлена на Рисунке 6.



*Рисунок 5 – Подробный алгоритм схемы с предусловием для нахождения f(x), pn(x)*



*Рисунок 6 –Последовательная схема интерполяции*

# ****4. ТЕСТРИОВАНИЕ ПРОГРАММЫ****

**Тестирование программного обеспечения** — процесс исследования, испытания программного продукта, имеющий две различные цели:

1. Продемонстрировать разработчикам и заказчикам, что программа соответствует требованиям;
2. Выявить ситуации, в которых поведение программы является неправильным, нежелательным или не соответствующим спецификации.

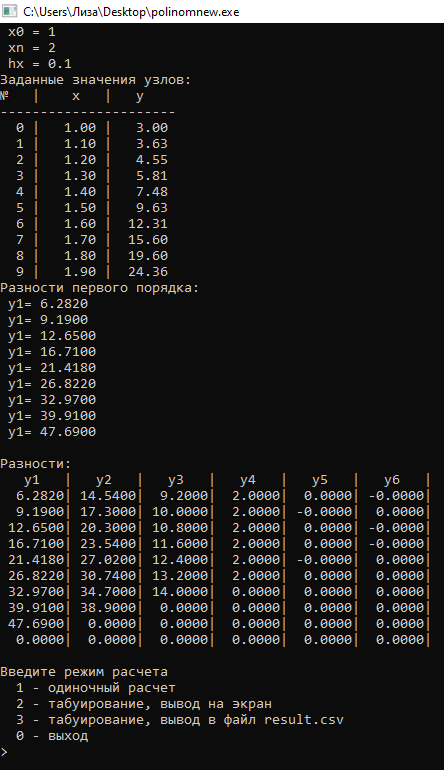
Существующие на сегодня методы тестирования программного обеспечения не позволяют однозначно и полностью выявить все дефекты и установить корректность функционирования анализируемой программы, поэтому все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого программного обеспечения. Такой процесс формальной проверки, или верификации, может доказать, что дефекты отсутствуют с точки зрения используемого метода.

Для тестирования программы «Интерполяция функции с помощью интерполяционной формулы Ньютона», нужно:

1. Выбрать интерполируемую функцию;
2. Провести расчет значения функции, задав ее в программе;
3. Сверить полученное значение полинома с также подсчитанным значением функции;
4. Построить график этой функции при различных x вручную и с помощью программы WPS Office.

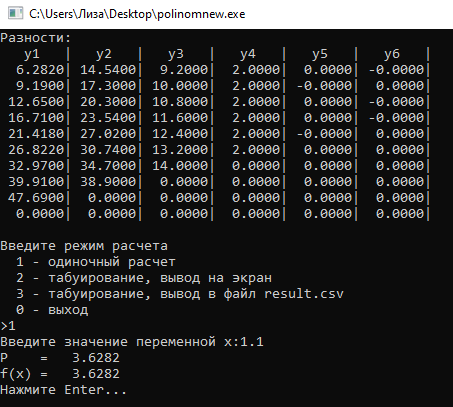
Проведем тестирование программы на примере функции **f(x) = 2\*x\*x\*x\*x - 3\*x + 4**.

Начальными данными будут



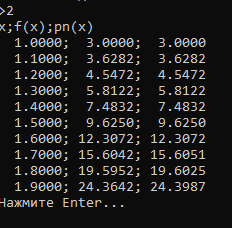
*Рисунок 7 –Расчет узлов функции и разностей первого и j – того порядков*

Теперь программа предлагает пользователю выбрать, каким образом будут получены значения функции и полинома. Вводим «1 – одиночный расчет»:



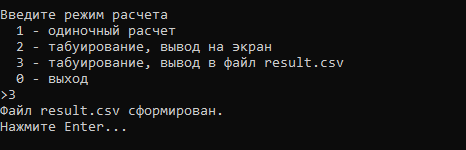
*Рисунок 8 –Одиночный расчет*

Далее, нажимаем «Enter» и программа возвращает нас в начало цикла. Выбираем «2 – табуирование функции»:



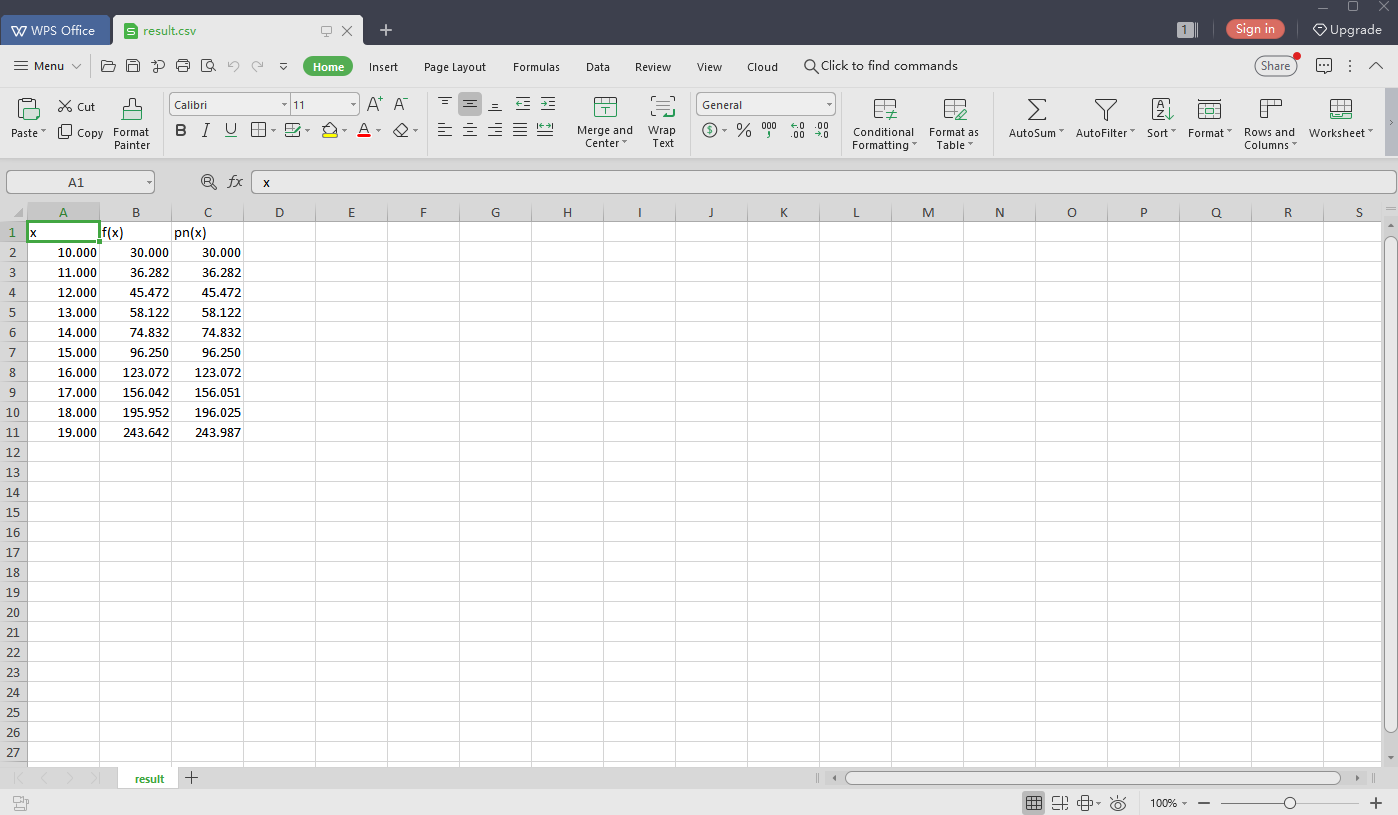
*Рисунок 9 –Табуирование функции с выводом значений на экран*

Если же выбираем «3 – табуирование с выводом в файл result.csv»:



*Рисунок 10 –Табуирование функции с выводом значений в файл result.csv*

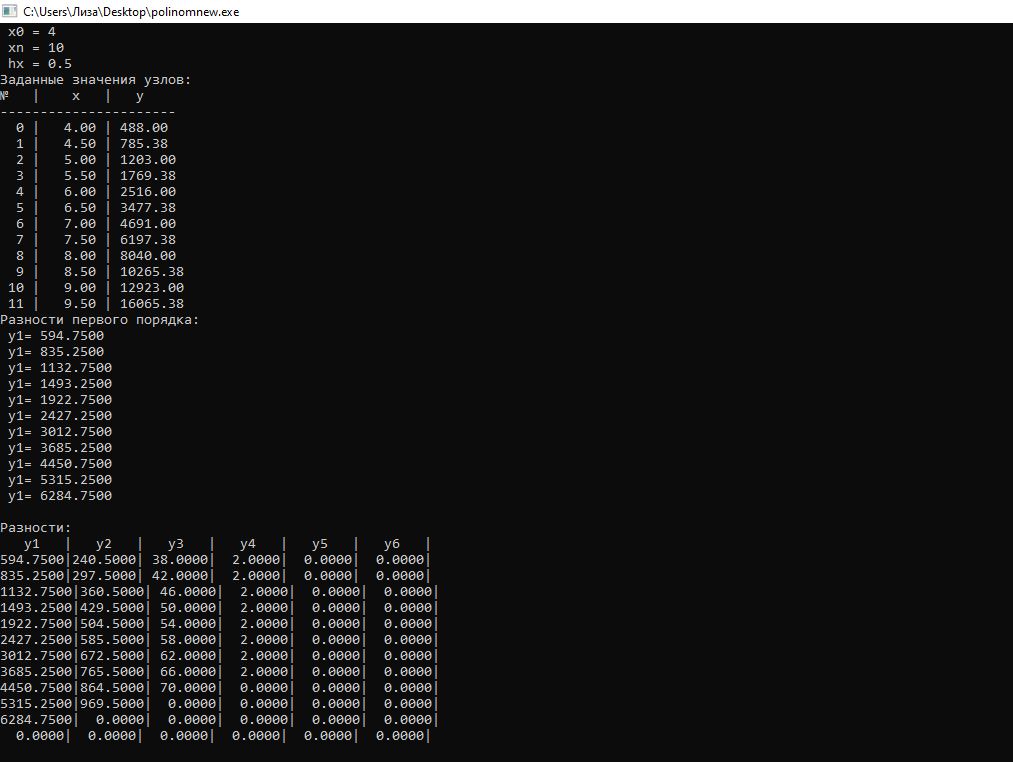
В последнем случае все полученные значения x, f(x), p(x) будут сформированы в файл, который представлен на Рисунке 11.



*Рисунок 11 –Файл result.csv*

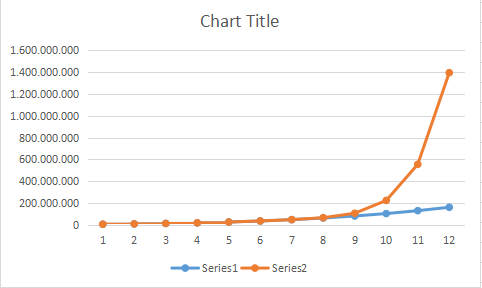
Как можно видеть, среди полученных значений f(x) и p(x) последние три различаются между собой. Построим график в программе WPS Office, чтобы увидеть, почему возникает такое различие. Возьмем другую функцию, более сложную, чтобы расстояние между кривыми было значимым.

**f(x) = 2\*x\*x\*x\*x - 3\*x\*x + 4\*x + 8**

****

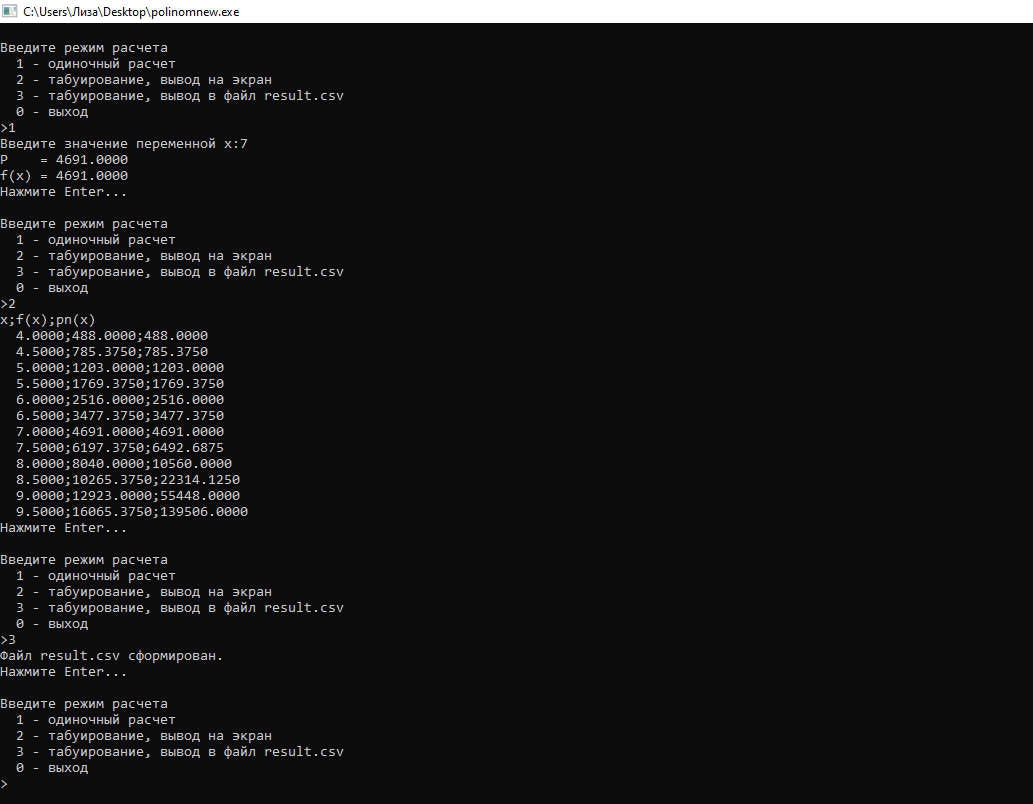
*Рисунок 12 –Вывод разностей для функции f(x)*

Так как разность порядка четыре остается константой, далее пятая и шестая равны нулю. Заметим, что при подсчете с помощью «Одиночного расчета» значения функции и полинома совпали: это значит, что программа верна. Выводим полученные данные в файл, открываем с помощью WPS Office:

****

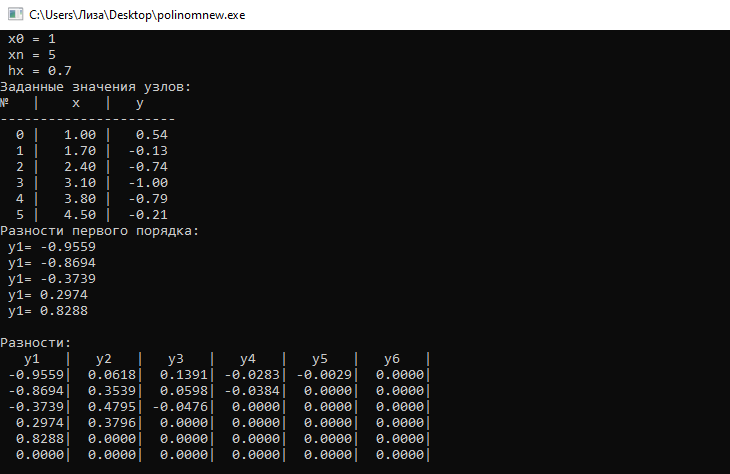
*График 1 – Графики функции и полинома*

На графике видно, что расхождение кривых функции f(x) и p(x) возникает при x = 7. При расчете значений табулированием функции, расхождение также начинается при том же x.

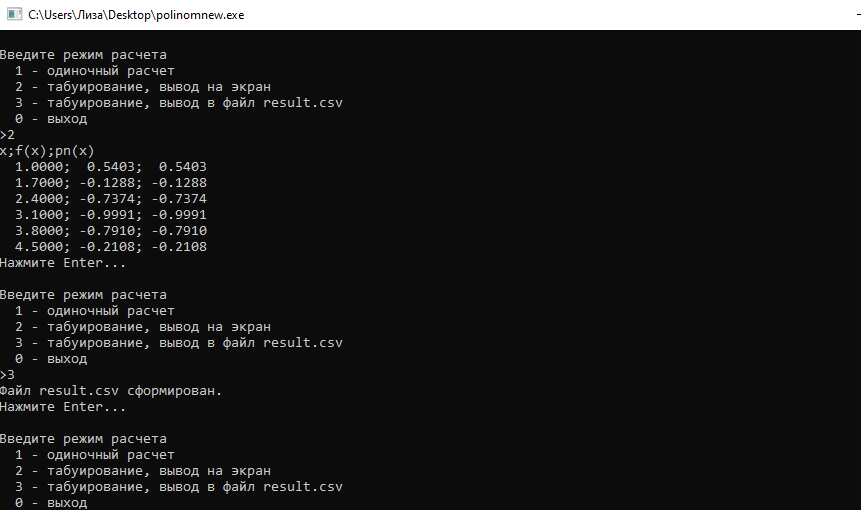
****

*Рисунок 13 –Расчет функции и полинома*

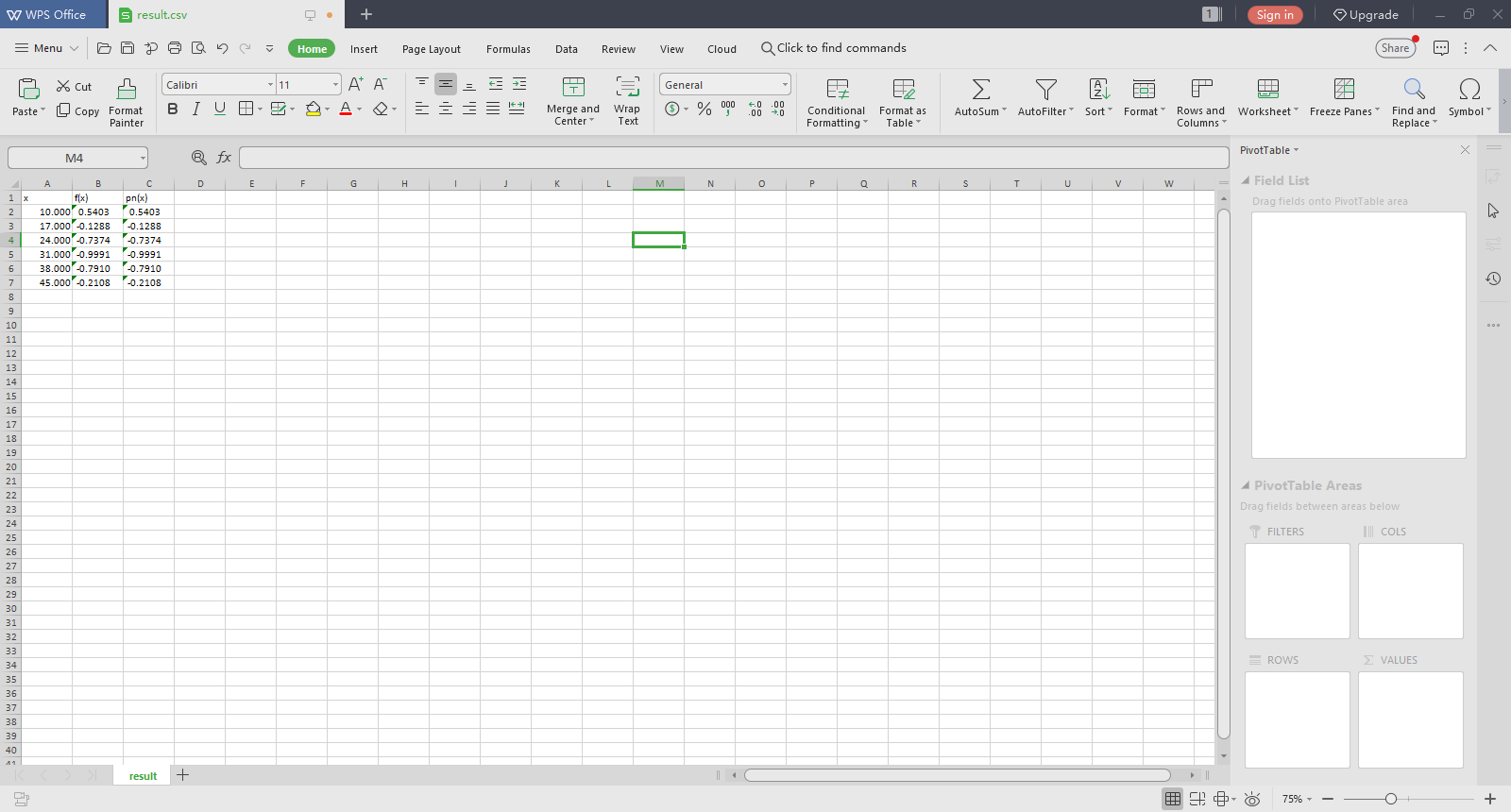
Построив вручную график данной функции в заданных узлах, можно сделать вывод, что программа успешно вычисляет в каждом режиме нужные нам функции, их узлы и разности разных порядков.

**

*Рисунок 14 –Тестирование программы на тригонометрической функции cos(x)*



*Рисунок 15 –Тестирование программы на тригонометрической функции cos(x)*



*Рисунок 16 –Файл с результатами вычислений тригонометрической функции cos(x)*

# 5. РАЗРАБОТКА ДОКУМЕНТАЦИИ

## 5.1. Назначение программы

В данном разделе приведено руководство программиста по использованию программы «Интерполяция математических формул интерполяционной формулой Ньютона» *Polinomnew.exe*.

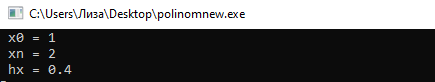
Программа предназначена для интерполирования заданной пользователем функции. Программа позволяет вводить нужные для исследования функции Она имеет несколько режимов работы:

* Одиночный расчет
* Табуирование функции и вывод на экран
* Табуирование функции и вывод в файл

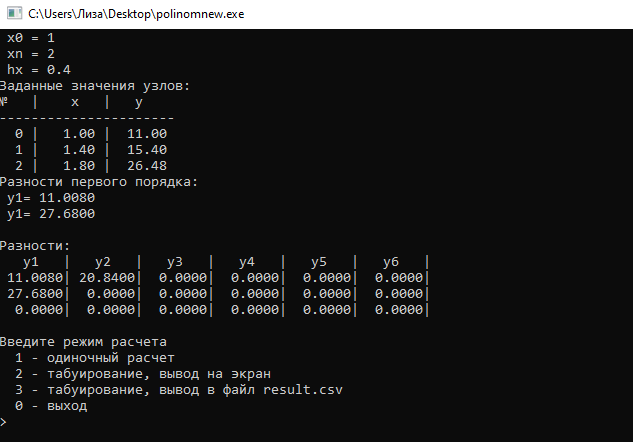
Работа программы осуществляется в консольном режиме. Размер исполняемого файла *Polinomnew.exe* равняется 96 981 байт.

## 5.2. Работа программы

Для запуска программы требуется двойной щелчок левой кнопкой мыши.

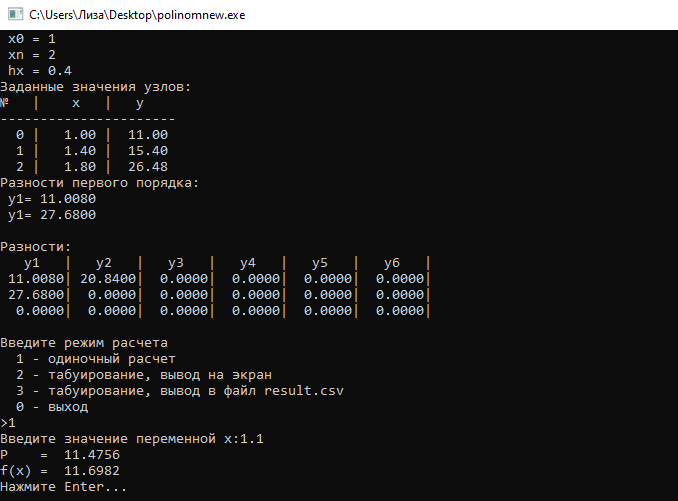
После запуска программа предоставляет возможность ввода начальных данных:

*Рисунок 17 –Ввод начальных данных*

Для ввода данных необходимо нажать клавишу «Enter». Программа производит расчет узлов, разностей первого и j – того порядков:

*Рисунок 18 –Расчет узлов функции, разностей*

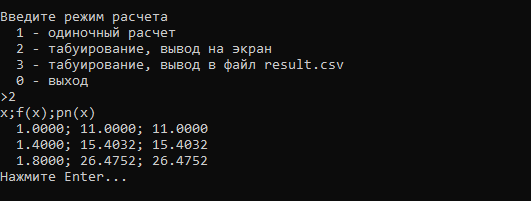
Программа предлагает пользователю выбрать режим работы. В первом случае необходимо ввести значение x, чтобы продолжить дальнейший расчет:



*Рисунок 19 –Одиночный расчет*

Здесь программа рассчитает функцию в произвольной точке x, определенной самим пользователем.

Выбирая второй режим, пользователь увидит значения f(x) и p(x) в каждой точке сегмента [x0, xn]:



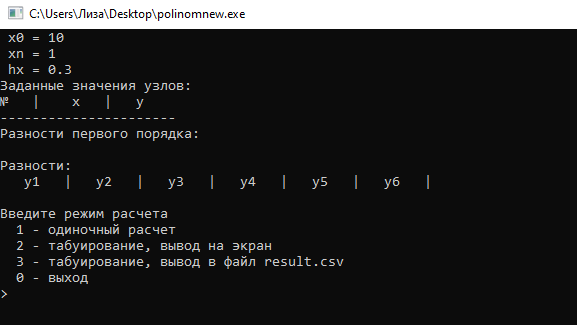
*Рисунок 20 –Табуирование с выводом на экран*

Последний режим работы выводит данные, полученные с помощью расчета во втором режиме, в файл *result.csv:*

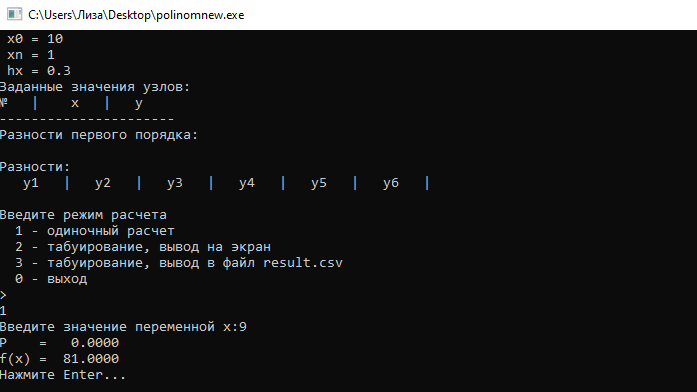


*Рисунок 21–Табуирование с выводом в файл result.csv*

## 5.3. Пример некорректного ввода данных



*Рисунок 22 –Пример некорректного ввода данных*



*Рисунок 23 –Пример расчета некорректно введенных данных*

# ЗАКЛЮЧЕНИЕ

В результате работы над программой «Интерполяция математических формул интерполяционной формулой Ньютона» *Polinomnew.exe*, был разработан алгоритм для написания данной программы. Она была протестирована на нескольких функциях, в том числе на тригонометрической, представлена документация и руководство по использованию программы *Polinomnew.exe.* Также были изучены «Массивы», в дальнейшем использованы в этой работе. Успешный тест этой программы показал, что все темы, усвоенные за курс «Программирование и алгоритмические языки», хорошо освоены.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. С.П. Шарый – «Курс вычислительных методов», Новосибирск - 2019
2. <https://github.com/naumovda/pascal-base/blob/master/slides/p-slides-05.pdf>
3. <https://github.com/naumovda/pascal-base/blob/master/slides/p-slides-03.pdf>
4. <https://github.com/naumovda/pascal-base/blob/master/slides/p-slides-04.pdf>
5. unetway.com/tutorial/testirovanie-programmnogo-obespecenia/
6. [www.pascal.helpov.net/index/one-dimensional\_arrays\_pascal\_programming](http://www.pascal.helpov.net/index/one-dimensional_arrays_pascal_programming)

**ПРИЛОЖЕНИЕ 1. ЛИСТИНГ ПРОГРАММЫ**

{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}

{ }

{ Проект PolinomNuytona }

{ Copyright (c) 2019 ФГБОУ РГРТУ им. В.Ф. Уткина }

{ ФВТ/кафедр КТ }

{ }

{ Разработчик: ст. гр. 848 Елисеева Е.В. }

{ Модифицирован: 19 декабря 2019 }

{ }

{\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}

**program** *polinomnew*;

//Установка кодовой страницы для отображения русских символов

**{$codepage utf8}**

**const**

N\_MAX = 100; //Максимальное значение i – ой переменной

MAX\_ORDER = 5; //Максимальный порядок разностей

**type**

//Тип данных для вычислений

TValue = real;

function f(x: TValue) : TValue;

**begin**

f : = x\*x\*x;

//Ввод произвольной функции

//exp(x) \* cos(x);

//2\*x\*x\*x\*x - 3\*x + 4;

//f := 2\*x\*x\*x\*x - 3\*x + 4;

//f := x\*x;

end;

**var**

xi, yi: array [0..N\_MAX] of TValue;

d: array[1..MAX\_ORDER, 0..N\_MAX] of TValue;

x, x0, xn, hx : TValue;

i, j, n, k, v : integer;

p, t : TValue;

fres : text;

begin

write (' x0 = '); readln(x0);

write (' xn = '); readln(xn);

write (' hx = '); readln(hx);

//Присваивание значений переменным xi и yi

i : = 0;

x : = x0;

**while** x < xn **do**

begin

xi[i] : = x;

yi[i] : = f(x);

i : = i + 1;

x : = x + hx;

end;

n : = i - 1;

writeln ('Заданные значения узлов:');

writeln ('№ | x | y ');

writeln ('----------------------');

**for** i : = 0 to n **do**

writeln (i: 3,' | ', xi[i]: 6:2,' | ', yi[i]: 6:2); //Выводим значения переменных xi, yi

**for** i := 0 to n - 1 **do**

d[1, i] := (yi [i+1] – yi [i])/hx;

//Формируем разность первого порядка

writeln ('Разности первого порядка:');

**for** i := 0 to n-1 **do**

writeln(' y1 = ', d[1, i]: 0: 4, ' ');

writeln;

**for** j : = 2 to MAX\_ORDER **do**

for i : = 0 to n do

d[j, i] : = 0;

//Формируем разность порядка j

**for** j : = 2 to MAX\_ORDER **do**

**for** i : = 0 to n - j **do**

d[j, i] : = (d[j - 1, i + 1] - d[j - 1, i])/(j\*hx);

writeln ('Разности:');

**for** j : = 1 to MAX\_ORDER **do**

write(' y', j,' |');

writeln;

**for** i : = 0 to n **do**

begin

**for** j : = 1 to MAX\_ORDER **do**

**if** i > n – j + 1 **then**

write(0.0:8:4, '|')

**else**

write(d[j, i]:8:4, '|');

writeln;

end;

writeln;

//Предлагаем пользователю выбрать режим дальнейшей работы

**repeat**

writeln('Введите режим расчета');

writeln(' 1 - одиночный расчет');

writeln(' 2 - табуирование, вывод на экран');

writeln(' 3 - табуирование, вывод в файл result.csv');

writeln(' 0 - выход');

write('>');

readln(v);

**if** v = 1 **then**

begin

write('Введите значение переменной х:');

readln(x);

p : = yi[0];

t : = 1;

**for** k : = 0 to n-1 **do**

begin

t : = t \* (x - xi[k]);

p : = p + t \* d[k + 1, 0];

end;

writeln ('P = ', P:8:4);

writeln ('f(x) = ', f(x): 8:4);

writeln('Нажмите Enter...');

//Возвращение к началу цикла

readln;

end

**else** **if** (v = 2) or (v = 3) **then**

begin

**if** v = 3 **then**

begin

assign(fres, 'result.csv');

//Связывает файловую переменную fres с физическим файлом result.csv

rewrite(fres);

//Открытие файла для записи в него информации

writeln (fres,'x; f(x); pn(x)');

end

**else**

writeln ('x; f(x); pn(x)');

x : = x0; i : = 0;

**while** x < xn **do**

begin

p := yi[0];

t := 1;

**for** k := 0 to n-1 **do**

begin

t := t \* (x - xi[k]);

p := p + t \* d[k+1, 0];

end;

**if** v = 2 **then**

begin

write(x: 8:4, ';');

write(f(x): 8:4, ';');

writeln(p: 8:4);

end

**else**

begin

write(fres, x: 8:4, ';');

write(fres, f(x): 8:4, ';');

writeln(fres, p: 8:4);

end;

i : = i + 1;

x : = x + hx;

end;

**if** v = 3 **then**

begin

close(fres);

//Конец работы с файлом, закрытие файла

writeln('Файл result.csv сформирован.');

end;

writeln('Нажмите Enter...');

//Возвращение к началу цикла

readln;

end

**until** v = 0; //Цикл выполняется до тех пор, пока пользователь не введет 0

**end**.