# ВВЕДЕНИЕ

В данный момент существует много программ для решения уравнений, вычисления интегралов и дифференциалов: MathCAD, MATLAB, и т.д. Они имеют высокую точность вычисления, высокую функциональность, но имеют и свои недостатки. Главные из них – сложный непонятный интерфейс, высокая многофункциональность недоступна рядовому пользователю.

Рынок нуждается в более простых аналогах приведенных выше программ. Созданный программный продукт способен решать уравнения с одной переменной методом Ньютона (касательных). Он прост в эксплуатации, имеет интуитивно понятный интерфейс и способен выстраивать график уравнения, что является очень важным для пользователя.

Программа будет полезна всем, как студентам высших учебных заведений, так и школьникам.

## 1. ПОСТАНОВКА ЗАДАЧИ

Цель создания программного продукта

Главной целью работы является разработка программы способной решать уравнения с одной переменной методом Ньютона (касательных), что должно являться пособием для студентов высших учебных заведений и для учащихся математических классов среднеобразовательных школ в снижении ненужной нагрузки, связанной с многочисленными массивами вычислений.

### 1.2. Постановка задачи

В данном программном продукте необходимо реализовать решение двух видов уравнений: y(x) =a×ln(b×x), y(x) =ax2+bx+c. Вместо коэффициентов должны использоваться параметры a, b, c, которые принимают значения, вводимые пользователем. Для нахождения корней, обязательным является указание промежутков, на которых определена функция, поэтому пользователь обязательно вводит промежутки функции m, n. Метод Ньютона является итерационным методом, следовательно, должна указываться погрешность вычисления ε. Обязательным является построение графика выбранной функции на заданном промежутке.

## 2. МАТЕМЕТИЧЕСКАЯ МОДЕЛЬ

Дисциплина "Численные методы" содержит набор методов и алгоритмов приближенного (численного) решения разнообразных математических задач, для которых точное аналитическое решение либо не существует, либо слишком сложно для использования на практике. При численном решении задач всегда возникает погрешность.

Выделяют абсолютную и относительную погрешность. Пусть р – точное значение искомого ответа, а p – приближённое значение, полученное с помощью численного метода.

Тогда – абсолютная погрешность,

 – относительная погрешность.

На первом этапе необходимо найти отрезок [a,b], на котором функция имеет ровно один корень. На втором этапе происходит уточнение корня на отрезке с заданной точностью с помощью одного из численных методов.

Метод, реализуемый в РУОП, называется методом Ньютона. Другое название метода – метод касательных.

Начальное условие:

Дано:

уравнение f(x) =0,

где f(x) ∈C'' [m,n], f(m) ⋅f(n) <0,

f'(x) и f''(x) знакопостоянны на отрезке [m,n] ;

точность ε.

Найти: решение уравнения с заданной точностью.

Пусть корень где – некоторое приближение к корню, – необходимая поправка. Разложим f(x) линейно в ряд Тейлора в окрестности xn (что соответствует замене функции в точке на касательную):

f(ξ) =0=f(xi+hi) ≈f(xi) +f'(xi) ⋅hi.

Отсюда:

.

Закон получения приближений к корню:

(2.1)

Начальное приближение x0 выбирается из условия:

. (2.2)

Графическая иллюстрация метода приведена на рисунке 2.1. Начальная точка в этом случае совпадает с n.

*x*

*f(x)*

m

n

Рисунок 2.1. – Метод Ньютона

Идея метода заключается в том, что последовательность приближений к корню строится путем проведения касательных к графику функции и нахождения их точек пересечения с осью ОХ.

Алгоритм метода.

Шаг 1. Найти первое приближение к корню x0 по формуле (2.2).

Шаг 2. Находить следующие приближения к корню по формуле (2.1), пока не выполнится условия окончания:

|xi-xi+1|<ε.

Последнее найденное приближение и будет корнем.

## 3. ОПИСАНИЕ И ОБОСНОВАНИЕ ВЫБОРА МЕТОДА РЕШЕНИЯ

Для обоснования выбора метода Ньютона для нахождения корней уравнений с одной переменной рассмотрим два другие итерационные метода.

### 3.1. Метод половинного деления

Другое название метода – метод дихотомии.

Дано:

уравнение f(x) =0,

где f(x) ∈C [m,n], f(m) ⋅f(n) <0;

точность ε.

Найти: решение уравнения с заданной точностью.

Другими словами, необходимо найти нуль функции на отрезке с заданной точностью. При этом функция непрерывна и в концах отрезка принимает значения разных знаков.

Алгоритм метода:

Шаг 1. Отрезок делится пополам. Находится точка с: = (b+a) /2 (см. рисунок 3.1).

f(x)

f(n)

0mkn

x

f(m)

Рисунок 3.1. – Метод половинного деления

Шаг 2. Проверяются следующие условия.

1. Если f(c) =0 – корень найден.

2. Если f(a) ⋅f(c) <0 – корень на [a,c], поэтому b: =c.

3. Если f(c) ⋅f(b) <0 – корень на [c,b], поэтому a: =c.

Шаг 3. Проверяется условие |a-b|<ε. Если условие выполнено, то считается, что корень найден. В этом случае он принимается равным а (хотя можно принять его равным b или даже (a+b) /2). Иначе переход к шагу 1.

### 3.2. Метод итераций

Дано:

уравнение f(x) =0,

где f(x) ∈C' [m,n], f(m) ⋅f(n) <0,

f'(x) знакопостоянна на отрезке [a,b] ;

точность ε.

Найти: решение уравнения с заданной точностью.

Идея метода заключается в том, что от уравнения f(x) =0 переходим с помощью равносильных преобразований к уравнению вида x=φ(x).Т. е. задача сводится к нахождению абсциссы ξ точки пересечения двух графиков функции (см. рис.2). В общем случае φ(x) =x-f(x) \*C.

*х*

*у*

*у=х*

*у=ϕ(х)*

*ξ*

Рисунок 3.2. – Корень уравнения

Точка ξ, для которой выполняется ξ= φ(ξ), называется неподвижной точкой процесса итераций. Очевидно, что эта точка является корнем уравнения f(x) =0.

Константа С подбирается таким образом, чтобы функция φ(x) удовлетворяла условиям сходимости метода итераций:

1) – является непрерывной и дифференцируемой на [m,n] ;

2) значения ;

3) для .

Если , то С нужно выбирать так, чтобы и для .

Метод состоит в построении последовательности приближений к корню. В качестве начального приближения выбирается любая точка x0∈ [a,b]. Для определенности можно брать середину отрезка [a,b]. В качестве формулы получения последующих приближений выступает сама φ(x):

Алгоритм метода:

Шаг 1. Найти первое приближение к корню x0 как середину отрезка [m,n].

Шаг 2. Находить следующие приближения к корню по формуле, пока не выполнится условия окончания:

|xi - xi+1|<ε.

Последнее найденное приближение и будет корнем.

3.3. Обоснование выбора метода

При рассмотрении обоих методов видно, что скорость сходимости метода Ньютона (касательных) выше скорости сходимости метода секущих (хорд) и метода итераций, следовательно, оптимальным для реализации в программе является метод Ньютона.

## 4. ОБОСНОВАНИЕ ВЫБОРА ЯЗЫКА ПРОГРАММИРОВАНИЯ

Реализация поставленной задачи совершается на языке программирования Turbo Pascal 7.0.

Система программирования Turbo Pascal, разработанная американской корпорацией Borland, остаётся одной из самых популярных систем программирования в мире. Этому способствует простота лежащая в основе языка программирования Pascal, а также поддержка графического и текстового режимов, что делает Turbo Pascal мощной современной профессиональной системой программирования.

## 5. ОПИСАНИ ПРОГРАММНОЙ РЕАЛИЗАЦИИ

### 5.1 Информационные потоки

Для наглядности работы программы, движению информации и взаимодейстия програмной части с аппаратной, разработана схема информационных потоков (рисунок 5.1).

ПРОГРАММА

Подсказки при вводе данных

Сообщение об ошибке ввода

Входные данные

Ввод

промежутков

Ввод

параметров

Ввод

погрешности

Блок

вывода результатов на экран

Выбор уравнения

Блок вычислений

Варианты сохранения результатов

Значение корня уравнения

Варианты продолжения

Рисунок 5.1 – Схема информационных потоков

Рисунок 5.1 – Схема информационных потоков (продолжение)

Блок построения графика

Блок помощи и справочной информации

График

Помощь и справочная информация

Блок выхода из программы

Условные обозначения:

– Данные, ввидение которых возможно

как из файлов, расположенных на

жёстком диске, так и с клавиатуры;

–Данные, выводимые на экран;

Данные, вводимые из файла.

### 5.2. Описание функционирования программы

При запуске программы на экране появляется титульный лист, отображающий информацию о студенте; далее загружается меню программы, состоящее из пяти пунктов:

МЕНЮ

ЗАСТАВКА

Справка

y(x)=a\*ln(b\*x)

y(x)=a\*x^2+b\*x+c

Построение графика

ВХОД

МЕНЮ

Рисунок 5.2 – Схема функционирования программы

– Справка

– y(x) =a\*ln(b\*x)

– y(x) =a\*x^2+b\*x+c

– Построение графика

– Выход

Пункт "Справка" включает в себя информацию о методе Ньютона. Пункты "y(x) =a\*ln(b\*x)" и "y(x) =a\*x^2+b\*x+c" представляют собой решения уравнений, где задаются промежутки m и n, параметры a, b(, c), погрешность E и выполняется сохранение в файлы. Пункт "Построение графика" строит график выбранного уравнения в зависимости от введённых параметров и промежутков. Пункт "Выход" – выход из программы. Схема функционирования представлена на рисунке 5.2.

### 5.3. Описание процедур и функций программы

Procedure title () – выводит титульную страницу на экран монитора;

Procedure graphica () – инициализирует графику.

Procedure pro () – содержит в себе переменную р, которая отвечает за номер выделяемой кнопки, передаётся как параметр в procedure key (p) и в procedure eat (p, bool), а также содержит в себе переменную bool, отвечающую за цикл в рамках procedure pro, передаётся как параметр в procedure eat (p2, bool);

Procedure eat (p2: byte; var bool: boolean) – в зависимости от параметра p2 выполняет один из пяти вариантов дальнейших действий программы. Переменная bool передаётся как параметр обратно в procedure pro;

Procedure key (p1: byte) – выстраивает графическую картинку меню в зависимости от параметра р1;

Procedure equation\_1 () – решение уравнения вида y(x) =a×ln(b×x). Переменная Е (погрешность) принимается как параметр из procedure load\_file\_3 (E), передаёт переменную Е как параметр в procedure save\_file (E);

Procedure equation\_2 () – решение уравнения вида y(x) =a×x2+b×x+c. Переменная Е (погрешность) принимается как параметр из procedure load\_file\_3 (E), переменная Е передаётся как параметр в procedure save\_file (E);

Procedure load\_file\_1 () – загружает переменные m и n (промежутки функции) из файла, либо обеспечивает их ввод с клавиатуры, в зависимости от желания пользователя. m, n – глобальные переменные в рамках программы;

Procedure load\_file\_2 () – загружает переменные a и b либо a, b, c (в зависимости от вида функции) (коэффициенты уравнения) из файла, либо обеспечивает их ввод с клавиатуры, в зависимости от желания пользователя. a, b, c – глобальные переменные в рамках программы;

Procedure load\_file\_3 (var E: real) – загружает переменную Е (погрешность функции) из файла, либо обеспечивает их ввод с клавиатуры, в зависимости от желания пользователя. Е передаётся как параметр и принимается как переменная в procedure equation\_1 и equation\_2;

Procedure save\_file (E: real) – сохраняет переменные a, b, (c,) m, n – глобальные в рамках программы в файлы либо не сохраняет, сохраняет переменную Е в виде параметра в файл, либо не сохраняет;

Procedure groffunc () – выстраивает график по значениям глобальных в рамках программы переменных a, b, (c,) m, n, с отмеченными на оси х приближениями и корнем уравнения. Содержит в себе function f (x: real): real, высчитывающую значение одной из функций в зависимости от аргумента х. Переменные у0 (масштаб) и у2 (максимальное значение функции) передаются в виде параметров в procedure setka (y0, y2);

Procedure setka (yn: integer; y2: real) – выстраивает координатную сетку и оцифровку осей x и y в зависимости от глобальных в рамках программы переменных m, n и параметров yn и y2;

Procedure help () – предоставляет пользователю непосредственную методологическую помощь.

### 5.4. Схема взаимодействия процедур программы

Для наглядности работы подпрограмм программы необходимо изобразить в виде схемы их взаимодействие между собой. Взаимодействие подпрограмм изображено на рисунке 5.3.

procedure pro;

procedure eat (p2 : integer; var bool : boolean);

procedure help;

procedure equation\_1;

procedure equation\_2;

procedure groffunc;

procedure key;

Procedure save\_file (E : real);

load\_file\_1;

load\_file\_2;

load\_file\_3 (var E : real);

procedure setka (yn: integer; y2 : real);;

procedure title;

procedure graphica;

Рисунок 5.3 – Взаимодействие процедур программы

Условные обозначения:

– запуск процедуры на которую указывает стрелка, из процедуры из которой она исходит.

### 5.5. Перечень обозначений

#### 5.5.1 Обозначения вводимых данных

m, n – промежутки функции;

a, b, c – коэффициенты уравнения, представленные в виде параметров;

E – погрешность, аналог ε в разделе "Описании математической модели" и в разделе "Описание (и обоснование выбора) метода решения".

5.5.2 Обозначения выводимых данных

y(x) =a\*ln(b\*x), y(x) =a\*x^2+b\*x+c – уравнения используемые в программе;

x – неизвестная, корень уравнения;

ln – логарифм;

x^2 – неизвестная x в степени 2.

5.6 Входные и выходные данные

5.6.1 Входные данные

y(x) =a\*ln(b\*x), y(x) =a\*x^2+b\*x+c – функция;

m, n: real – левый и правый промежутки функции соответственно;

a, b, c: real – параметры, коэффициенты уравнения;

E: real – погрешность;

"Помощь и справочная информация".

5.6.2 Выходные данные

x1: real – значение корня уравнения;

st: string – текстовые сообщения, возникающие в процессе выполнения программы (ошибки и варианты дальнейшего продолжения).

5.6.3 Промежуточные данные

Bool\_of: Boolean – определяет цикл выполнения алгоритма решения;

mass: real – массив [1. . 20] ;

number: byte – глобальная переменная, номер функции;

code\_of: byte – переменная, отвечающая за необходимость поиска корня уравнения;

root: real – разность приближений.

### 5.7. Алгоритм решения задачи

#### 5.7.1. Алгоритм нахождения корня уравнения y(x) =a×ln(b×x)

Алгоритм решения уравнения вида y(x) =a×ln(b×x) приводится на рисунке 5.4.

выполнять

ввод

пока ((m <= n) и (((m < 0) и (n < 0)) или ((m > 0) и (n > 0))));

m, n;

выполнять

ввод

пока (b\*m > 0);

a, b;

если (a = 0) то

вывод

ввод

пока (E > 0);

Все х на промежутке – корни уравнения;

E;

number: =0;

иначе

выполнять

i: =1; если (a\*ln(b\*m) \*(-a/sqr(m))) > 0 то

mass [i]: =m;

code\_of: =1;

иначе

Рисунок 5.4 – Алгоритм решения уравнения вида y(x) =a×ln(b×x)

Уравнение не имеет корней;

Уравнение не имеет корней;

x1;

если (a\*ln(b\*n) \*(-a/sqr(n))) > 0 то

mass [i]: =n;

code\_of: =1;

иначе

вывод

number: =0; code\_of: =0;

если (code\_of = 1) то

выполнять

x1: =mass [i] -a\*ln(b\*mass [i]) /

(a/mass [i]);

root: =Abs (x1-mass [i]);

i: =i+1;

mass [i]: =x1;

пока (root < E);

если (x1 < m) или (x1 > n) то

вывод

number: =0; code\_of: =0;

вывод

Рисунок 5.4 – Алгоритм решения уравнения вида y(x) =a×ln(b×x) (продолжение)

5.7.2. Алгоритм нахождения корня уравнения y(x) =a×x2+b×x+c

Алгоритм решения уравнения вида y(x) =a×x2+b×x+c приводится на рисунке 5.5.

выполнять

ввод

пока (m <=n);

ввод

пока (E > 0);

Все х на промежутке – корни уравнения;

a, b, c;

m, n;

E;

ввод

если (a = 0) и (b = 0) и (c = 0) то

вывод

number: =0;

иначе

выполнять

i: =1;

если (a\*sqr(m) +b\*m+c) \*(2\*a) >= 0 то

mass [i]: =m;

code\_of: =1;

иначе

Рисунок 5.5 – Алгоритм решения уравнения вида y(x) =a×x2+b×x+c

Уравнение не имеет корней;

Уравнение не имеет корней;

x1;

если (a\*sqr(n) +b\*n+c) \*(2\*a) >= 0 то

mass [i]: =n;

code\_of: =1;

иначе

вывод

number: =0; code\_of: =0;

если (code\_of = 1) то

выполнять

x1: =mass [i] -((a\*sqr(mass [i]) +

b\*mass [i] +c) /(2\*a\*mass [i] +b));

root: =Abs (x1-mass [i]);

i: =i+1;

mass [i]: =x1;

пока (root < E);

если (x1 < m) или (x1 > n) то

вывод

number: =0; code\_of: =0;

вывод

Рисунок 5.5 – Алгоритм решения уравнения вида y(x) =a×x2+b×x+c (продолжение)

Алгоритмы решения уравнений рис.5.4 и рис.5.5 соответствуют procedure equation\_1 и procedure equation\_2 в программе соответственно.

## 6. КОМПЛЕКТАЦИЯ И ЗАГРУЗКА ПРОГРАММЫ

### 6.1. Комплектация

Папка my\_stuff, в которой содержится:

– RUOP. exe – основной файл программы;

– help. asc – файл с методологической информацией;

– m\_n. txt – файл, содержащий значения промежутков m и n;

– a\_b\_c. txt – файл, содержащий значения параметров a, b, c;

– E. txt – файл, содержащий значение погрешности E;

– egavga. bgi – файл для работы с графикой;

– keyrus. com – файл для работы с русским языком;

– trip. chr – файл, содержащий русский шрифт.

### 6.2. Порядок инсталляции и запуск программы

Требуется скопировать папку my\_stuff с содержащимися в ней файлами в папку “c: \temp\”. Для запуска программы необходимо запустить файл RUOP. exe, расположенный в папке my\_stuff.

При копировании программы в иную папку, невозможными становятся работа "Справки" загрузка и автоматическое сохранение информации в файлы.

## 7. ТЕСТОВЫЕ ПРИМЕРЫ

Тестовые примеры необходимы пользователю для того, чтобы узнать возможности, которые предоставляет данный программный продукт или протестировать его на правильность решения уравнений.

Тестовые примеры для решения уравнения вида y(x) =a\*ln(b\*x) приводятся в таблице 6.1.

Таблица 7.1. Тестовые примеры для уравнения вида y(x) =a\*ln(b\*x)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| m | n | a | b | E | Результат |
| 1 | 10 | 1 | 0.5 | 0.01 | 2 |
| -20 | -0.01 | 9 | -2 | 0.01 | -0.2 |
| 9 | 14 | 100 | 1 | 0.01 | Уравнение не имеет корней |

Тестовые примеры для решения уравнения вида y(x) =a\*x^2+b\*x+c приводятся в таблице 6.2.

Таблица 7.2. Тестовые примеры для уравнения вида y(x) =a\*x^2+b\*x+c

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| m | n | a | b | c | E | Результат |
| -10 | 10 | 5 | 29 | 3 | 0.01 | -0.1054 |
| -10 | 10 | 0 | 4 | 10 | 0.01 | -2.5 |
| 5 | 20 | 5 | 29 | 4 | 0.01 | Уравнение не имеет |

При введении в программу данных, не отвечающих требованиям типу, будет появляться сообщение "Ошибка ввода", пока не будут введены правильные данные, соответствующие требованиям программы.

Если уравнение не имеет корней, то построение графика и сохранение данных, результатов становиться невозможным.

При введении в программу данных, отвечающих требованиям, будут появляться сопроводительные сообщения (советы) по дальнейшим вариантам продолжения.

Если уравнение имеет корень, то построение графика и сохранение данных, результатов становится возможным.

# ВЫВОДЫ

В процессе создания была написана программа, осуществляющая решение уравнения с одной переменной методом Ньютона (касательных). Программа способна решать два вида уравнений, а также выстраивать график по вводимым данным.

В программе реализована работа с графикой и с файлами, имеет интуитивно понятный интерфейс, реализована возможность справки.

Корректная работа программы обеспечивается строгим следованием методическим указаниям, а также надёжной системой проверки промежуточных результатов в ходе выполнения самой программы.

Однако ощутимыми недостатками являются расчёт результатов всего для двух функций и отсутствие касательных к графику при построении графика функции, устранение которых планируется в ближайшее время.

В целом получившийся программный продукт является отличным пособием для студентов высших учебных заведений и для учащихся математических классов среднеобразовательных школ.

# ПЕРЕЧЕНЬ ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Фаронов В.В. "Turbo Pascal 7.0. Начальный курс": учебное пособие. – М.: Кнорус, 2006. – 576 с.

2. Сухарёв М. Turbo Pascal 7.0. Теория и практика программирования. – СПб: "Наука и техника", 2003. – 576 с.

3. Методические указания по оформлению студенческих работ для студентов специальностей 080403 "Программное обеспечение автоматизированных систем", 080404 "Интеллектуальные системы принятия решений", 050103 "Экономическая кибернетика"; Утверждено на заседании учёного совета ДонГИИИ протокол № 7 от 23.02. 2004 г. – Донецк: ДонГИИИ, 2004, 46 с.

# Приложение А

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

А.1 Общие сведения

Полное название программного продукта: "Численные методы. Решение уравнений с одной переменной методом Ньютона (касательных)". Её условное обозначение РУОП. Работа выполняется студентом 1-го курса Донецкого государственного института искусственного интеллекта (ДонГИИИ), факультета СКИТ, группы СУА-05, Николаевым Алексеем Сергеевичем.

Основанием для разработки РУОП является задание, выданное кафедрой Программного обеспечения интеллектуальных систем (ПОИС).

Плановый срок начала работы: 17 февраля 2006 года.

Дата защиты работы: 22 мая 2006 года.

А.2 Назначения и цели создания программы

Данная программа создана как учебное пособие для студентов высших учебных заведений и для учащихся математических классов среднеобразовательных школ. Позволяет решать уравнения вида y(x) =a×ln(b×x) и y(x) =ax2+bx+c методом Ньютона (касательных).

А.3 Требования к программному продукту

А.3.1. Общие требования

Программа должна выполнять следующие требования:

1) решать два вида уравнений: y(x) =a×ln(b×x) и y(x) =ax2+bx+c методом Ньютона (касательных);

2) поддержку графического меню, состоящего из пяти пунктов:

– помощь и справочная информация;

– y(x) =a×ln(b×x);

– y(x) =a×x^2+b×x+c;

– построение графика;

– выход;

3) по вводимым значениям промежутков уравнения и по вводимым значениям коэффициентов уравнения:

– вычислять корень уравнения в зависимости от вводимых данных;

– выстраивать график уравнения, отмечая, на оси абсцисс, промежуточные корни уравнения, выводить значение корня уравнения.

А.3.2. Функциональные требования

Для реализации программного продукта необходимо разработать:

1) поддержку файлов, предоставление возможности решать пользователю самому, вводить начальные данные из файла или с клавиатуры, необходимость сохранения данных и полученных результатов в файлы;

2) систему справочной информации по реализуемому в РУОП методу Ньютона.

А.3.2. Требования к техническому обеспечению

Рекомендуемые характеристики аппаратных средств:

– КПУ: i486;

– ОЗУ: 4 мб;

– видеоадаптер VGA, EGA;

– монитор: VGA, EGA;

– клавиатура;

– свободное дисковое пространство – около 100 килобайт.

А.3.3. Требования к программному обеспечению

Для успешной загрузки программы требуется наличие операционной системы MS DOS 6.0.

А.3.5. Требования к организационному обеспечению

Организационное обеспечение включает в себя пояснительную записку с приложениями: техническое задание, руководство пользователя, экранные формы, тексты программы.

# Приложение Б

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Главное меню появляется после титульного листа. Меню состоит из пяти пунктов. Скроллинг осуществляется клавишами "z" и "x". Вход в подменю осуществляется клавишей "Enter".

В пункте "Справка" содержится методологическая информация по методу Ньютона.

В пункте "y(x) =a\*ln(b\*x)" осуществляется решение уравнения y(x) =a\*ln(b\*x) по вводимым параметрам, промежуткам и погрешности. В пункте осуществляется загрузка данных из файлов и сохранение данных в файлы по желанию пользователя.

В пункте "y(x) =a\*x^2+b\*x+c" осуществляется решение уравнения y(x) =a\*x^2+b\*x+c по вводимым параметрам, промежуткам и погрешности. В пункте осуществляется загрузка данных из файлов и сохранение данных в файлы по желанию пользователя.

В пункте "Построение графика" осуществляется построение графика по вводимым в уравнение данным.

В пункте "Выход" осуществляет выход из программы.

# Приложение В

ЭКРАННЫЕ ФОРМЫ

Рисунок В.1 – Заставка, титульная страница

Рисунок В.2 – Меню

Рисунок В.3 – Общий вид окна "y(x) =a\*ln(b\*x)"

Рисунок В.4 – Общий вид окна "y(x) =a\*x^2+b\*x+c"

Рисунок В.5 – График функции y(x) =1\*ln(0.5\*x) на промежутке [1; 10]

Рисунок В.6 – График функции y(x) =5\*sqr(x) +29\*x+3 на промежутке [-10; 10]

# Приложение Г

ЛИСТИНГ ПРОГРАММЫ

program Restorant;

uses CRT, Graph;

var a, b, c, m, n: real;

number, i: byte;

mass: array [1. . 20] of real;

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

procedure title;

begin

textcolor(2);

writeln (' Министерство образования Украины');

writeln (' Донецкий государственный институт искусственного интеллекта');

writeln;

writeln (' Кафедра ПОИС');

writeln;

writeln;

writeln (' Курсовая работа');

writeln (' По курсу "АЯ и П"');

writeln (' На тему: "Решение нелинейных уравнений методом Ньютона');

writeln (' (методом секущих)" ');

writeln;

writeln;

writeln (' Выполнил: ');

writeln (' Студент группы СУА-05');

writeln (' Николаев А.С. ');

writeln (' Проверил: ');

writeln (' cт. преп. кафедры ПОИС');

writeln (' Бычкова Е.В. ');

writeln (' асс. кафедры ПОИС');

writeln (' Волченко E. B. ');

writeln;

writeln (' 2005');

writeln;

writeln;

textcolor (red);

writeln ('Нажмите "Ввод" для продолжения"');

textcolor (lightgray); Readln;

end;

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

procedure pro; FORWARD;

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

procedure graphica;

var d, r, e: integer;

begin

d: =detect;

InitGraph (d, r, '');

e: =GraphResult;

if e <> grOK then WriteLn (GraphErrorMsg (e)) else pro;

end;

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

procedure setka (yn: integer; y2: real);

var x, y, cross, dcross: integer;

lx, ly, dlx, dly: real;

st: string;

begin

If abs (m) < abs (n) then

dlx: =Abs (n/6.25) else dlx: =Abs (m/6.25);

dly: =y2/((yn-110) /40);

dcross: =0;

lx: =6\*dlx;

SetColor (LightGray);

For cross: = 1 to 7 do

begin

Str (lx: 0: 1, st);

If lx >=0 then

OutTextXY (535-dcross, yn+7, st) else

OutTextXY (525-dcross, yn+7, st);

lx: =lx-2\*dlx;

dcross: =dcross+80;

end;

x: =80;

Repeat

SetLineStyle (DottedLn, 0, NormWidth);

Line (x, yn-3, x, 110); Line (x, yn+3, x, 360);

SetLineStyle (SolidLn, 0, NormWidth);

Line (x, yn-3, x, yn+3);

x: =x+40;

Until x = 600;

ly: =0;

y: =yn;

Repeat

If ly > 0 then

begin

Line (317, y, 323, y);

Str (ly: 0: 1, st);

OutTextXY (295, y+7, st);

end;

ly: =ly+dly;

SetLineStyle (DottedLn, 0, NormWidth);

Line (323, y, 570, y); Line (70, y, 317, y);

SetLineStyle (SolidLn, 0, NormWidth);

y: =y-40;

Until (y < 110);

ly: =0;

y: =yn;

Repeat

If ly < 0 then

begin

Line (317, y, 323, y);

Str (ly: 0: 1, st);

OutTextXY (285, y+7, st);

end;

ly: =ly-dly;

SetLineStyle (DottedLn, 0, NormWidth);

Line (323, y, 570, y); Line (70, y, 317, y);

SetLineStyle (SolidLn, 0, NormWidth);

y: =y+40;

Until (y > 360);

end;

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

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

procedure groffunc;

var l, y0: integer;

y1, y2, x, y, mx, my: real;

gr, grand: string;

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

function f (x: real): real;

begin

Case number of

1: f: =a\*ln(b\*x);

2: f: =a\*sqr(x) +b\*x+c;

end;

end;

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

begin

If number=0 then OutTextXY(300, 20, 'Введите сначала данные в уравнение!!! ') else

begin

ClearDevice;

SetBKColor (black);

case number of

1: grand: =('y(x) =\*ln(\*x) ');

2: begin grand: =('y(x) =\*sqr(x) +\*x+');

str (c: 0: 2, gr); insert (gr, grand, 17); end;

end;

str (b: 0: 2, gr); insert (gr, grand, (6+number\*4));

str (a: 0: 2, gr); insert (gr, grand, 6);

OutTextXY (300, 40, grand);

y1: =0; y2: =0;

x: =m;

Repeat

y: =f (x);

if y < y1 then y1: =y;

if y > y2 then y2: =y;

x: =x+0.01;

Until (x >= n);

my: =250/abs (y2-y1);

If (abs (m) > abs (n)) then mx: =250/abs (m) else

mx: =250/abs (n);

y0: =360-abs (Round (y1\*my));

setka (y0, y2);

SetColor (blue);

Line (320, 360, 320, 90);

Line (70, y0, 590, y0);

Line (320, 90, 317, 93); Line (320, 90, 323, 93);

Line (590, y0, 587, y0-3); Line (590, y0, 587, y0+3);

OutTextXY (595, y0-5, 'x'); OutTextXY (315, 80, 'y');

OutTextXY (400, 450, 'Нажмите "Ввод" для выхода');

If Abs (m) > Abs (n) then y1: =Abs (m) else y1: =Abs (n);

SetColor (Red);

str (mass [i]: 5: 4, grand);

OutTextXY (300+Round ((250/y1) \*mass [i]), 400, grand);

Line (320+Round ((250/y1) \*mass [i]), y0, 320+Round ((250/y1) \*mass [i]), 390);

For l: =1 to i-1 do

begin

SetColor (2+l);

Line (320+Round ((250/y1) \*mass [l]), y0+10, 320+Round ((250/y1) \*mass [l]), y0-10);

end;

x: =m;

Repeat

y: =f (x);

PutPixel (320+Round (x\*mx), y0-Round (y\*my), 15);

x: =x+0.01;

Until (x >= n);

ReadLn;

pro;

end;

end;

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

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

procedure load\_file\_1;

var mistake: byte;

k: char;

st: string;

f: text;

begin

Repeat

If number = 1 then

WriteLn (' Введите промежутки [m, n] одного знака') else

WriteLn (' Введите промежутки [m, n] ');

WriteLn ('Нажмите "1" для ввода данных с клавиатуры');

WriteLn ('Нажмите "2" для ввода данных из файла');

k: =ReadKey;

Case k of

'1': begin

WriteLn (' Ввод: ');

{$I-}

ReadLn (m, n);

{$I+}

mistake: =IOResult;

If mistake <> 0 then WriteLn ('Ошибка ввода');

end;

'2': begin

WriteLn (' Нажмите "1" для указания расположения своего файла');

WriteLn (' Нажмите "2" для ввода из файла, созданного автоматически');

k: =ReadKey;

If k = '1' then begin

WriteLn ('Введите путь к файлу с расширением. txt');

ReadLn (st);

Assign (f, st);

end else

If k = '2' then assign (f, 'c: \temp\my\_stuff\m\_n. txt');

{$I-}

Reset (f);

{$I+}

mistake: =IOResult;

If mistake <> 0 then

WriteLn ('Файла не существует') else

begin

{$I-}

Read (f, m, n);

{$I+}

mistake: =IOResult; Close (f); If mistake <> 0 then

WriteLn ('Информация в файле не соответствует нужному типу') else

begin

WriteLn (m: 0: 2);

WriteLn (n: 0: 2);

end;

end; WriteLn ('Нажмите "Ввод для продолжения"'); ReadLn;

end;

end;

Until mistake = 0;

end;

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

procedure load\_file\_2;

var mistake: byte;

k: char;

st: string;

f: text;

begin

Repeat

WriteLn ('Нажмите "1" для ввода с клавиатуры');

WriteLn ('Нажмите "2" для ввода данных из файла');

k: =ReadKey;

Case k of

'1': begin

WriteLn (' Ввод: ');

If number = 1 then {$I-} ReadLn (a, b) {$I+} else

If number = 2 then {$I-} ReadLn (a, b, c) {$I-};

mistake: =IOResult;

If mistake <> 0 then WriteLn ('Ошибка ввода');

end;

'2': begin

WriteLn (' Нажмите "1" для указания расположения своего файла');

WriteLn (' Нажмите "2" для ввода из файла, созданного автоматически');

k: =ReadKey;

If k = '1' then begin

WriteLn ('Введите путь к файлу расширением. txt');

ReadLn (st);

assign (f, st);

end else

If k = '2' then assign (f, 'c: \temp\my\_stuff\a\_b\_c. txt');

{$I-}

Reset (f);

{$I+}

mistake: =IOResult;

If mistake <> 0 then

WriteLn ('Файла не существует') else

begin

If number = 1 then {$I-} Read (f, a, b) {$I+} else

{$I-} Read (f, a, b, c); {$I+}

mistake: =IOResult; Close (f); If mistake <> 0 then

WriteLn ('Информация в файле не соответствует нужному типу') else

begin

WriteLn (a: 0: 2);

WriteLn (b: 0: 2);

If number = 2 then WriteLn (c: 0: 2);

end;

end; WriteLn ('Нажмите "Ввод для продолжения"'); ReadLn;

end;

end;

Until mistake = 0;

end;

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

procedure load\_file\_3 (var E: real);

var mistake: byte;

k: char;

st: string;

f: text;

begin

Repeat

WriteLn ('Нажмите "1" для ввода данных с клавиатуры');

WriteLn ('Нажмите "2" для ввода данных из файла');

k: =ReadKey;

Case k of

'1': begin

WriteLn (' Ввод: ');

{$I-}

ReadLn (E);

{$I+}

mistake: =IOResult;

If mistake <> 0 then WriteLn ('Ошибка ввода');

end;

'2': begin

WriteLn (' Нажмите "1" для указания расположения своего файла');

WriteLn (' Нажмите "2" для ввода из файла, созданного автоматически');

k: =ReadKey;

If k = '1' then begin

WriteLn ('Введите путь к файлу с расширением. txt');

ReadLn (st);

assign (f, st);

end else

If k = '2' then assign (f, 'c: \temp\my\_stuff\E. txt');

{$I-}

Reset (f);

{$I+}

mistake: =IOResult;

If mistake <> 0 then

WriteLn ('Файла не существует') else

begin

{$I-}

Read (f, E);

{$I+}

mistake: =IOResult; Close (f); If mistake <> 0 then

WriteLn ('Информация в файле не соответствует нужному типу') else

begin

WriteLn (E: 0: 3);

end;

end; WriteLn ('Нажмите "Ввод для продолжения"'); ReadLn;

end;

end;

Until mistake = 0;

end;

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

procedure save\_file (E: real);

var k: char;

mistake: byte;

f: text;

st: string;

begin

Repeat

WriteLn (' Если хотите сохранить данные и результаты нажмите "1"');

WriteLn (' Если не хотите сохранять данные и результаты нажмите "2"');

k: =ReadKey;

Case k of

'1': begin

WriteLn (' Если хотите сохранить данные в указанные вами файлы нажмите "1"');

WriteLn (' Если хотите, чтобы сохранение произошло автоматически нажмите "2"');

k: =ReadKey;

If k = '1' then begin

Repeat

WriteLn ('Введите путь и имя файла c для сохранения промежутков [m, n] ');

ReadLn (st);

Assign (f, st);

{$I-}

ReWrite (f);

{$I+}

mistake: =IOResult;

If mistake <> 0 then WriteLn ('Файл не может быть создан') else

begin

Write (f, m: 3, n: 3); Close (f); WriteLn ('Информация сохранена. Нажмите "Ввод"'); ReadLn;

end;

Until mistake = 0;

Repeat

If number = 1 then

WriteLn ('Введите путь и имя файла для сохранения коэффициентов "a", "b"')

else

If number = 2 then

WriteLn ('Введите путь и имя файла для сохранения коэффициентов "a", "b", "c"');

ReadLn (st);

Assign (f, st);

{$I-}

ReWrite (f);

{$I+}

mistake: =IOResult;

If mistake <> 0 then WriteLn ('Файл не может быть создан') else

begin

If number = 1 then begin

Write (f, a: 3, b: 3); Close (f); WriteLn ('Информация сохранена. Нажмите "Ввод"'); ReadLn;

end else

If number = 2 then begin

Write (f, a: 3, b: 3, c: 3); Close (f); WriteLn ('Информация сохранена. Нажмите "Ввод"'); ReadLn;

end;

end;

Until mistake = 0;

Repeat

WriteLn ('Введите путь и имя файла для сохранения погрешности "Е"');

ReadLn (st);

Assign (f, st);

{$I-}

ReWrite (f);

{$I+}

mistake: =IOResult;

If mistake <> 0 then WriteLn ('Файл не может быть создан') else

begin

Write (f, E: 3); Close (f); WriteLn ('Информация сохранена. Нажмите "Ввод"'); ReadLn;

end;

Until mistake = 0;

Repeat

WriteLn ('Введите путь и имя файла для сохранения корня');

ReadLn (st);

Assign (f, st);

{$I-}

ReWrite (f);

{$I+}

mistake: =IOResult;

If mistake <> 0 then WriteLn ('Файл не может быть создан') else

begin

Write (f, mass [i]: 3); Close (f); WriteLn ('Информация сохранена. Нажмите "Ввод"'); ReadLn;

end;

Until mistake = 0;

end else

If k = '2' then begin

Assign (f, 'c: \temp\my\_stuff\m\_n. txt');

{$I-} ReWrite (f); {$I+}

mistake: =IOResult;

If mistake <> 0 then WriteLn ('Каталога для сохранения не существует') else

begin

Write (f, m, n); Close (f);

Assign (f, 'c: \temp\my\_stuff\a\_b\_c. txt');

ReWrite (f); If number = 1 then Write (f, a, b) else

Write (f, a, b, c); Close (f);

Assign (f, 'c: \temp\my\_stuff\E. txt');

ReWrite (f); Write (f, E); Close (f);

Assign (f, 'c: \temp\my\_stuff\x. txt');

ReWrite (f); Write (f, mass [i]); Close (f);

WriteLn ('Информация сохранена. Нажмите "Ввод"'); ReadLn;

end;

end;

end;

'2': mistake: =0;

end;

Until mistake = 0;

end;

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

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

procedure equation\_1;

var mistake, code\_of: byte;

E, x1, root: real;

bool\_of: boolean;

k: char;

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

begin

closegraph;

bool\_of: =false;

Repeat

number: =1;

clrscr;

WriteLn (' Уравнение вида: y(x) =a\*ln(b\*x) ');

Repeat

load\_file\_1;

If m > n then begin

WriteLn ('Введите "m" < "n" ');

WriteLn ('Нажмите "Ввод" для подолжения'); ReadLn;

end else

If (m < 0) and (n >0) or (m = 0) or (n = 0) then

begin

WriteLn ('"m" и "n" должны быть одного знака и неравные 0');

WriteLn ('Нажмите "Ввод" для продолжения'); ReadLn;

end;

Until (((m < 0) and (n < 0)) or ((m > 0) and (n > 0))) and (m <= n);

Repeat

WriteLn ('Введите коэффициенты уравнения "a", "b"');

load\_file\_2;

If m\*b <= 0 then begin

WriteLn ('попробуйте ввести "b" другого знака и неравное 0');

WriteLn ('Нажмите "Ввод" для продолжения'); ReadLn;

end;

Until m\*b > 0;

If a = 0 then begin

WriteLn ('Все "x" на промежутке [',m: 0: 1,'; ',n: 0: 1,'] - решения уравнения');

number: =0; end else

begin

Repeat

WriteLn ('Введите погрешность "E"');

load\_file\_3 (E);

If E <= 0 then begin WriteLn ('Введите "Е" больше 0');

WriteLn ('Нажмите "Ввод" для продолжения"');

end;

Until E > 0;

i: =1;

If (a\*ln(b\*m) \*(-a/sqr(m))) > 0 then begin mass [i]: =m; code\_of: =1 end else

If (a\*ln(b\*n) \*(-a/sqr(n))) > 0 then begin mass [i]: =n; code\_of: =1 end else

begin WriteLn ('Уравнение не имеет корней'); number: =0; code\_of: =0; end;

If code\_of = 1 then

begin

Repeat

x1: =mass [i] -a\*ln(b\*mass [i]) /(a/mass [i]);

root: =Abs (x1-mass [i]);

i: =i+1;

mass [i]: =x1;

Until root < E;

If (x1 < m) or (x1 > n) then

begin WriteLn ('Уравнение не имеет корней'); number: =0; code\_of: =0; end else

WriteLn ('Корнем уравнения y(x) =', a: 0: 1, '\*ln(', b: 0: 1, '\*x) является: ', x1: 5: 4);

end;

end;

WriteLn ('Нажмите "Ввод"'); ReadLn; If code\_of = 1 then save\_file (E) else

WriteLn ('Так как уравнение не имеет корней, то сохранение не выполняется');

WriteLn ('Если хотите выйти, то нажмите "ESC"');

WriteLn ('Если хотите ввести другие данные, то нажмите "Ввод"');

k: =ReadKey;

code\_of: =ord (k);

case code\_of of

27: begin

bool\_of: =true; graphica;

end;

13: bool\_of: =false;

end;

Until bool\_of;

end;

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

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

procedure equation\_2;

var mistake, code\_of: byte;

E, x1, root: real;

bool\_of: boolean;

k: char;

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

begin

closegraph;

bool\_of: =false;

Repeat

number: =2;

clrscr;

WriteLn (' Уравнение вида: y(x) =a\*x^2+b\*x+c');

Repeat

load\_file\_1;

If m > n then WriteLn ('Введите "m" < "n" ');

Until (m <= n);

WriteLn ('Введите коэффициенты уравнения "a", "b", "c"');

load\_file\_2;

If (a = 0) and (b = 0) and (c = 0) then begin

WriteLn ('Все "х" на промежутке [',m: 0: 1,'; ',n: 0: 1,'] - решения уравнения');

number: =0; end else

begin

Repeat

WriteLn ('Введите погрешность "Е"');

load\_file\_3 (E);

If E <= 0 then begin WriteLn ('Введите E > 0');

WriteLn ('Нажмите "Ввод" для продолжения');

end;

Until E > 0;

i: =1;

If (a\*sqr(n) +b\*n+c) \*(2\*a) >= 0 then begin mass [i]: =n; code\_of: =1 end else

If (a\*sqr(m) +b\*m+c) \*(2\*a) >= 0 then begin mass [i]: =m; code\_of: =1 end else

begin WriteLn ('Уравнение не имеет корней'); number: =0; code\_of: =0; end;

If code\_of = 1 then

begin

Repeat

x1: =mass [i] -((a\*sqr(mass [i]) +b\*mass [i] +c) /(2\*a\*mass [i] +b));

root: =Abs (x1-mass [i]);

i: =i+1;

mass [i]: =x1;

Until (root < E);

If (x1 < m) or (x1 > n) then

begin WriteLn ('Уравнение не имеет корней'); number: =0; code\_of: =0; end else

WriteLn ('Корнем уравнения y(x) =', a: 0: 1, '\*x^2+', b: 0: 1, '\*x+', c: 0: 1, ' является: ', x1: 0: 4);

end;

end;

WriteLn ('Нажмите "Ввод"'); ReadLn; If code\_of = 1 then save\_file (E) else

WriteLn ('Так как уравнение не имеет корней, то сохранение не выполняется');

WriteLn ('Если хотите выйти, то нажмите "ESC"');

WriteLn ('Если хотите ввести другие данные, то нажмите "Ввод"');

k: =ReadKey;

code\_of: =ord (k);

case code\_of of

27: begin

bool\_of: =true; graphica;

end;

13: bool\_of: =false;

end;

Until bool\_of;

end;

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

procedure key (p1: byte);

Var y1, y2: integer;

name: string;

i: byte;

begin

ClearDevice;

SetColor (white);

OutTextXY (250, 435, '"Ввод" - вход "z", "x" - перемещение по меню');

y1: =15;

y2: =70;

for i: =1 to 5 do

begin

Setcolor (blue);

Rectangle (16, y1-1, 251, y2-1);

RecTangle (17, y1-2, 252, y2-2);

RecTangle (18, y1-3, 253, y2-3);

SetFillStyle (1,lightblue);

Bar (15, y1, 250, y2);

case i of

1: Name: ='Cправка';

2: Name: ='y=a\*ln(b\*x) ';

3: Name: ='y=a\*x^2+b\*x+c';

4: Name: ='Построение графика';

5: Name: ='Выход';

end;

SetColor (white);

OutTextXY (45, y1+25, Name);

y1: =20+y2;

y2: =75+y2;

end;

SetColor (white);

p1: =p1-1;

Rectangle (18, 19+75\*p1, 246, 66+75\*p1);

end;

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

procedure help;

var st: string;

f: text;

y: integer;

mistake: byte;

begin

ClearDevice;

Assign (f, 'c: \temp\My\_stuff\help. asc');

{$I-}

Reset (f);

{$I+}

mistake: =IOResult; SetTextStyle (0, 0, 0);

If mistake <> 0 then OutTextXY (250, 220, 'Файла не существует') else

begin

y: =0;

Repeat

y: =15+y;

ReadLn (f, st);

OutTextXY (45, y, st);

Until EOf (f);

Close (f);

end;

OutTextXY (400, 450, 'Нажмите "Ввод" для выхода');

ReadLn; pro;

end;

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

procedure eat (p2: byte; var bool: boolean);

begin

if p2=1 then help else

if p2=2 then equation\_1 else

if p2=3 then equation\_2 else

if p2=4 then groffunc else

if p2=5 then bool: =true;

end;

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

procedure pro;

var p, code: byte;

k: char;

bool: boolean;

begin

ClearDevice;

p: =1;

key (p);

bool: =false;

repeat

SetBKColor(lightgray);

SetTextStyle (1, 0, 4); SetColor (blue);

OutTextXY (390, 130, 'МЕНЮ');

SetTextStyle (0, 0, 0);

k: =ReadKey;

code: =ord (k);

Case code of

122: begin

p: =p-1; if p=0 then p: =5;

key (p);

end;

120: begin

p: =p+1; if p=6 then p: =1;

key (p);

end;

13: eat (p, bool);

end;

until bool;

CloseGraph;

end;

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

begin

title;

number: =0;

graphica;

end.