МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ(МАИ)
(ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)

Факультет №3
«СИСТЕМЫ УПРАВЛЕНИЯ, ИНФОРМАТИКА И ЭЛЕКТРОЭНЕРГЕТИКА»
каф. 308
«Информационные системы»

Пояснительная записка к курсовой работе
по теории чисел

Выполнил студент
1 курса,
группы 03-119,
Злобин Д.В.

Преподаватель:
доцент, к.т.н.
Гридин А.Н.

Москва 2010

Задание

Разработать и отладить программу на языке Pascal (Delphi), в операционной системе Windows 7 Ultimate, выполняющую следующие функции:

1. Формирование подмножества натуральных чисел с заданными параметрами.
2. Факторизация числа с опциями.
3. Нахождение наименьшего общего кратного (НОК) и наибольшего общего делителя (НОД) заданной совокупности чисел.
4. Нахождение рациональных решений алгебраического уравнения с целочисленными коэффициентами.
5. Представление рациональной дроби в виде цепной
6. Представление цепной дроби в виде рациональной.

Содержание

Введение

Дуальность природы (единство и борьба противоположностей, положительное и отрицательное, притяжение и отталкивание, аморфное и структурированное и т.п.) отражается и в математике, где выделяются непрерывные образования (например, множество точек на отрезке линии, на плоскости, в объеме, множество действительных чисел и т.п.) и отдельные (конечные) объекты (множество натуральных чисел, особые точки функций, логические переменные, операторы и операнды и т.п.).

Область математики, которая занимается изучением структур финитного (конечного) характера, в настоящее время обычно называют дискретной математикой в отличие от классической математики, которая в основном занимается изучением свойств объектов непрерывного характера.

В общем случае дискретная математика охватывает все произвольные дискретные структуры: алгебраические системы, графы (включая и бесконечные графы), конечные группы, вычислительные среды и проч..

Свойства изучаемого дискретной математикой объекта приводят к ряду особенностей, отличающих дискретную математику от классической.

Прежде всего, это отказ от таких понятий классической математики, как предел и непрерывность, а отсюда и малопригодность многих ее мощных средств анализа.

Другими особенностями являются:

•проблемы алгоритмической разрешимости и построение конкретных решающих алгоритмов;

•исследование дискретных многоэкстремальных задач, где методы существенно использующие гладкость функции, мало эффективны (типичные примеры: построение нормальных минимальных дизъюнктивных форм; определение условий, ограничивающих полный перебор и т.п.)

Еще одна особенность дискретной математики связана с методами ее изучения. В настоящее время при изучении классической математики в высшей школе (исключая, естественно, подготовку математиков-профессионалов) имеет место склонность к «рецептурному»методу (решение задач по существующим алгоритмам или, в других случаях, по более или менее сложным моделям).

Изучение же дискретной математики, связанной, и весьма тесно, с проблемами управления и развития информационных технологий, часто направлено на создание моделей и эффективных алгоритмов. В такой ситуации математика нужна, прежде всего, как метод мышления, как язык, как средство формулирования и организации понятий. Такое владение математикой требует большей культуры: понимания важности точных формулировок и умения обходиться без них там, где это целесообразно; умения понять, что просто, что сложно, а что невозможно, ощущения связи между может быть далекими идеями и понятиями.

Таким образом, цель изучения дискретной математики состоит не только в освоении определенного набора понятий и приемов решения задач, а и в существенном повышении культуры пользования математическим аппаратом в вышеприведенном смысле.

Теория чисел — это одно из направлений математики, которое иногда называют «высшей арифметикой». Данная наука изучает натуральные числа и некоторые сходные с ними объекты, рассматривает различные свойства (делимость, разложимость, взаимосвязи и так далее), алгоритмы поиска чисел, а также определяет ряд достаточно интересных наборов натуральных чисел.

Теория чисел среди математических дисциплин выделяется скорее психологической установкой, чем предметом «целые числа». Более сильное утверждение было бы неверным: в теоретико-числовых работах исследуются и алгебраические, и трансцендентные числа; или, вообще, не числа, а скажем, аналитические функции очень специального вида {ряды Дирихле, модулярные формы); или геометрические объекты {решетки, схемы над Z). Прежде всего, целые числа образуют первичную материю математики вообще (точнее, одну из двух первичных материй; Вторая — это «фигуры», геометрия).

История элементарной теории чисел поэтому столь длинна, как история всей математики, а историю современной математики можно было бы условно начинать с того времени, когда «числа» и «фигуры» прочно объединились в идее координатизации, которая по замечанию И. Р. Шафаревича лежит в основе алгебры. Далее, целые числа как универсум идеи дискретного являются также универсумом

любых логических конструкций, в том числе любых математических рассуждений, оформленных зкак таковые. Мы подчеркиваем, что математика как акт индивидуального творчества, конечно, к логике не сводится, но в коллективном сознании нашей эпохи существует в виде потенциально завершимой огромной и точной логической конструкции. Если этот образ постоянно размывается его, так сказать, нежизненностью, то и восстанавливающие его тенденции сильны; сейчас к ним добавилась компьютерная реальность с ее чрезвычайно жесткими требованиями к логической структуре математической продукции в виде программного обеспечения. Пониманием того, что свойства целых чисел суть свойства дискретного вообще и, стало быть, свойства мира математических рассуждений, в частности, мы обязаны математике двадцатого века, в первую очередь Гёделю. При желании, это донимание может быть оформлено внутри математики в виде теоремы о том, что задача доказуемости внутри любой формальной системы равносильна задаче о разрешимости в целых числах подходящего диофантова уравнения. Этот парадоксальный факт — свидетельство того, что теория чисел, будучи малой частью математического знания, в потенции все это знание содержит. Недаром Карл Фридрих Гаусс любил говорить, что математика — царица наук, а теория чисел — царица математики.

Сама же написанная программа включает в себя набор из нескольких основных операций, которые могут понадобиться при решении более сложных задач, как из теории чисел, так и из других разделов математики.

1.2.Описание программы

DMC.exe

1. Назначение

Выполняет следующие функции:

1. Формирование подмножества натуральных чисел, объединенных общими делителями и остатком среди чисел данной размерности.
2. Факторизация числа и формирование множества его делителей и их суммы.
3. Нахождение наименьшего общего кратного (НОК) и наибольшего общего делителя (НОД) заданной совокупности чисел.
4. Нахождение рациональных решений алгебраического уравнения с целочисленными коэффициентами с использованием схемы Горнера.
5. Представление рациональной дроби в виде цепной.
6. Представление цепной дроби в виде рациональной.

1.2.Оборудование и ПО

ОС Microsoft Windows 7 Ultimate, среда программирования Borland Delphi 7.

Аппаратная часть:

Процессор: Intel Core i7-920,

Видеокарта: GeForce GTX 275

Оперативная память: Kingston 3x2Gb RAM.

Математическая основа решения, алгоритмы.

1. Numerator

Эта программа выполняет формирование подмножества натуральных чисел, объединённых общими делителями и остатком среди чисел данной размерности. Для этого сначала ищется наименьшее общее кратное (НОК) делителей, далее, находится 1-е число среди необходимой размерности, которое делит-

ся на НОК с заданным остатком. Затем, к этому числу мы прибавляем НОК и получаем 2-е число и так далее, пока не дойдем до границ размерности.

2.Factorizator

Эта программа выполняет факторизацию числа, то есть разложение его на простые сомножители, а также формирует множество этих сомножителей и считает их сумму. Для начала ищем простые числа, на которые делится заданное число, проверяем кол-во повторений (то есть степень этого простого числа). Далее находим все делители числа и составляем из них множество. Вычисляем сумму делителей.

3.NOD_NOK

Эта программа находит наименьшее общее кратное (НОК) и наибольший общий делитель (НОД) заданной совокупности чисел, используя алгоритм Евклида. Для этого сначала мы считаем по этому алгоритму НОД 2х чисел - находим максимальное из двух, делим на 2-е с остатком, затем делим второе на полившийся остаток и так далее, пока не остаток не станет равным 0. Остаток, предшествующий остатку, равному 0 и будет НОДом. НОК находится перемножением двух исходных чисел и деление их на НОД. Далее, мы находим НОД и НОК следующего числа с НОД и НОК предыдущей двойки. Продолжаем да тех пор, пока не найдем НОД и НОК всей совокупности.

4.Superhorner

Эта программа находит рациональные решения алгебраического уравнения с целочисленными коэффицентами с использованием схемы Горнера. Для этого нужно ввести старшую степень неизвестного , коэффиценты при них и свободный член. Далее, свободный член раскладывается на рациональные сомножители, которые в свою очередь подставляются в исходное уравнение. Для упрощения этой проверки используется схема Горнера. Заключается она в том, что к коэф. при старшей степени прибавляем коэффицент старшей степени,умноженный на выбранный сомножитель, + коэффицент n-1 степени + коэффицент n-1 степени, умноженный на выбранный сомножитель и т.д. Если выполняется равенство, следовательно, этот сомножитель и является одним из корней исходного уравнения.

5.Expressor

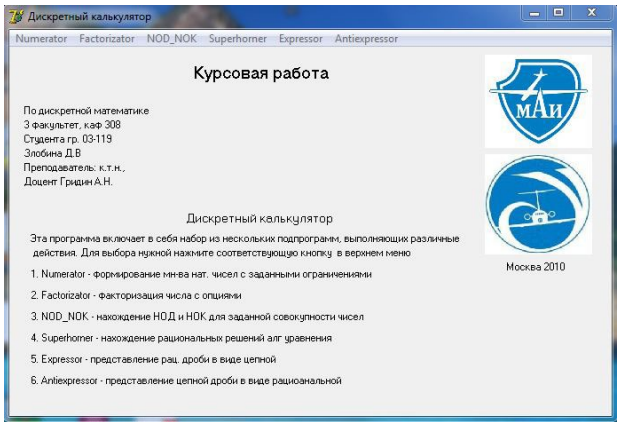
Эта программа представляет рациональную дробь в виде цепной. Для этого сначала выделяется целая часть из исходной дроби, затем остаток представляем в виде «обратной» дроби(например, было 3/5, стало 1/ (5/3)), выделяем целую часть из получившегося знаменателя и т.д., пока не останется дробь, «переворот» которой ничего не даст. Целые части и знаменатели записываются через запятую в квадратных скобках, это есть цепная дробь.

6. Antiexpressor

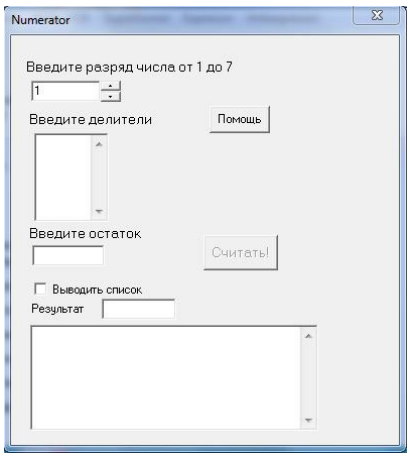
Эта программа представляет цепную дробь в виде рациональной. Она выполняет операцию, обратную той которая используется в программе Expressor, тем самым, «собирая» рациональную дробь.

Интерфейс программы.

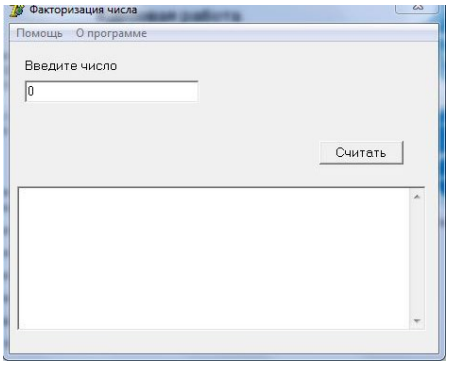
1. Основная программа



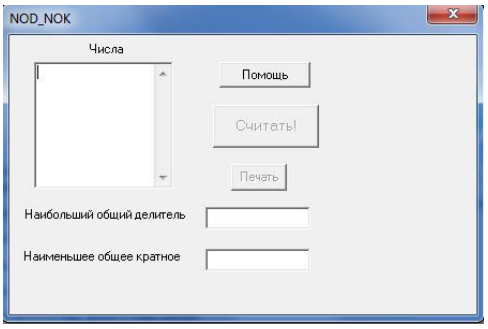
2. Numerator



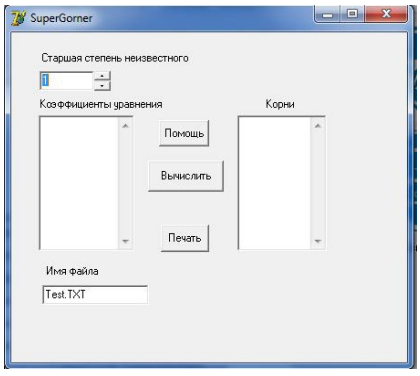
3. Factorizator



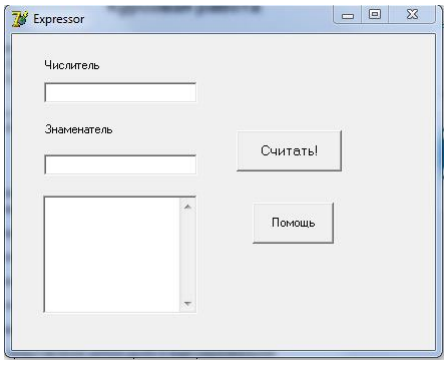
4. NOD_NOK



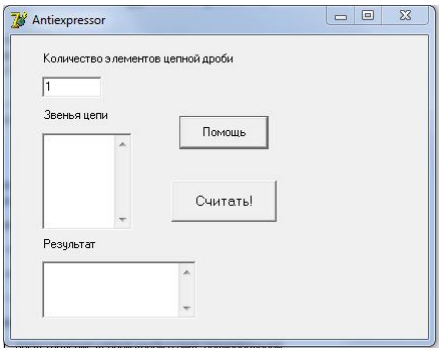
5. Superhorner



6. Expressor



7. Antiexpressor



Тесты

1.Numerator

а) Корректные

1)Разрядность: 2
  Делители: 12,10
  Остаток: 1
  Результат: Чисел 1.
  61

2) Разрядность: 3
  Делители: 11, 13
  Остаток: 7
  Результат: Чисел 6.
 150, 293, 436, 579, 722, 865

б) Некорректные

1) Разрядность: 2
  Делители: 10
  Остаток: 12
  Сообщение об ошибке: «Остаток должен быть меньше делителя»

2)Разрядность 2
  Делители: -2
  Сообщение об ошибке: «Делитель должен быть больше 0»

2. Factorizator

а) Корректные

1)Число 123
   Результат:
   123 = 3^1 *   41^1

Кол-во делителей T(123)= 4

Множество делителейD(123)= { 1, 3, 41, 123 }

Сумма делителей S(123)= 168

2) Число 123
   Результат:
   4781 = 7^1 *   683^1

Кол-во делителей T(123)= 4

Множество делителей D(123)= { 1, 7, 683, 4781 }

Сумма делителей S(123)= 5472

б) Некорректные

1) Число 0
Сообщение об ошибке: «Число должно быть больше 0»

2) Число 2000000000
Сообщение об ошибке: «Число должно быть меньше 1000000000»

3.NOD_NOK

а) Корректные

1)Числа 11, 12
Результат:
НОД= 1
НОК= 132

2) Числа 3, 7, 5
Результат:
НОД= 1
НОК= 105

б) Некорректные

1) Числа 0, 10
Сообщение об ошибке: «Число должно быть не меньше 1»

2) Число 1, 2, 4, 6, 5, 9, 12, 13
Сообщение об ошибке: «Количество чисел должно быть меньше 6»

4.Superhorner

а) Корректные

1)Степень: 4
Коэффиценты: 1, 2, -11, 4, 4
Результат:
1, 2

2) Степень: 3
Коэффиценты: 1, 17, 58, -24
Результат:
-12

б) Некорректные

1) Степень: 11
Сообщение об ошибке: «Максимальная степень неизвестного не больше 10»

2) Степень: 3

Коэффициенты: 1, 17
Сообщение об ошибке: «Введите еще 2 коэффицента уравнения»

5. Expressor
а) Корректные

1)Числитель: 123
Знаменатель: 456
Результат:
[0, 3, 1, 2, 2, 2, 2]

2) Числитель: 17
Знаменатель: 49
Результат:
[0, 2, 1, 7, 2]

б) Некорректные

1) Числитель: 17
Знаменатель: 0
Сообщение об ошибке: «Знаменатель должен быть больше 0»

6. Antiexpressor
а) Корректные

1)Кол-во звеньев: 7
Звенья
[0, 3, 1, 2, 2, 2, 2]
Результат
41/152

2) Кол-во звеньев:4
Звенья
[4, 2, 1, 7]
Результат
100/23

б) Некорректные

1) Кол-во звеньев:4
Звенья
[4, 0, 2, 1,]

Сообщение об ошибке: «Элементы цепи должны быть больше 0»

Заключение

Была разработана программа, выполняющая следующие функции:

1. Формирование подмножества натуральных чисел, объединенных общими делителями и остатком среди чисел данной размерности.
2. Факторизация числа и формирование множества его делителей и их суммы.
3. Нахождение наименьшего общего кратного (НОК) и наибольшего общего делителя (НОД) заданной совокупности чисел.
4. Нахождение рациональных решений алгебраического уравнения с целочисленными коэффициентами с использованием схемы Горнера.
5. Представление рациональной дроби в виде цепной.
6. Представление цепной дроби в виде рациональной.

Программа написана на языке Delphi, ОС Microsoft Windows 7 Ultimate.

Аппаратная часть:

Процессор: Intel Core i7-920,

Видеокарта: GeForce GTX 275

Оперативная память: Kingston 3x2Gb RAM.

Проведенные тесты показали работоспособность программы.

К плюсам программы можно отнести нетребовательность к ресурсам компьютера (тестировалась на более слабом оборудовании), простоту в обращении.

Программа имеет четкую структуру: главная программа содержит описание и пункты в меню в соответствующие подпрограммы. Интерфейс прост и интуитивно понятен. В каждой подпрограмме есть кнопка «Помощь», которая описывает работу с ней.

Недостатком программы является ограниченность в оперировании с числами большой разрядности.

Для устранения этого недостатка в будущем, необходимо будет использовать более эффективные алгоритмы и средства написания, специально предназначенные для использования в вычислительных машинах.

Список используемых источников:

1. В.И. Николаев, В. Я. Пашкин “Основы дискретной математики”, 1999г.

Приложения

Листинг программы.

```
//Main programm

unit KPUnit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, Menus, StdCtrls, jpeg, ExtCtrls;

type
  TfrmKP = class(TForm)
    MainMenu1: TMainMenu;
    MmNum: TMenuItem;
    MmFac: TMenuItem;
    MmSuperGorner: TMenuItem;
    MmExpressor: TMenuItem;
    MmAntiExpresor: TMenuItem;
    MmNOD_NOK: TMenuItem;
    Image1: TImage;
    Image2: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    procedure MmNumClick(Sender: TObject);
    procedure MmFacClick(Sender: TObject);
    procedure MmExpressorClick(Sender: TObject);
    procedure MmSuperGornerClick(Sender: TObject);
    procedure MmNOD_NOKClick(Sender: TObject);
    procedure MmAntiExpresorClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmKP: TfrmKP;

implementation

uses dm002Unit, DM003Unit, DM004Unit, DM007Unit,
DM005Unit, dm001Unit,
  DM008Unit;

{$R *.dfm}

procedure TfrmKP.MmNumClick(Sender: TObject);
var
 frmDM001: TfrmDM001;
begin
  frmDM001:=TfrmDM001.Create(Self);
    frmDm001.Show;
end;

procedure TfrmKP.MmFacClick(Sender: TObject);
var
  Form1: TForm1;
begin
 Form1:=TForm1.Create(Self);
    Form1.Show;
end;

procedure TfrmKP.MmSuperGornerClick(Sender: TObject);
var
  frmSuperGorner: TfrmSuperGorner;
begin
 frmSuperGorner:=TfrmSuperGorner.Create(Self);
 frmSuperGorner.Show;
end;

procedure TfrmKP.MmExpressorClick(Sender: TObject);
var
  Form2: TForm2;
begin
 Form2:=TForm2.Create(Self);
 Form2.Show;
end;

procedure TfrmKP.MmNOD_NOKClick(Sender: TObject);
var
  frmNumer: TfrmNumer;
begin
  frmNumer:=TfrmNumer.Create(Self);
  frmNumer.Show;
end;

procedure TfrmKP.MmAntiExpresorClick(Sender: TOb-
ject);
var
  Antiexpressor: TAntiexpressor;
begin
  Antiexpressor:=TAntiexpressor.Create(Self);
  Antiexpressor.Show;
end;

end.



//Numerator


unit dm001Unit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, ComCtrls, StdCtrls;

type
  TfrmDM001 = class(TForm)
    edtNumDigit: TEdit;
    udNumDigit: TUpDown;
    lblNumDigit: TLabel;
    LblDiv: TLabel;
    edtlost: TEdit;
    Lbllost: TLabel;
    btnRun: TButton;
    mmResultList: TMemo;
    cbViewList: TCheckBox;
    lblResult: TLabel;
    edtResult: TEdit;
    btnHelp: TButton;
    mmInp: TMemo;
    procedure edtDivKeyPress(Sender: TObject; var
Key: Char);
    procedure btnRunClick(Sender: TObject);
    procedure edtlostExit(Sender: TObject);
```

```
    procedure edtNumDigitChange(Sender: TObject);
    procedure btnHelpClick(Sender: TObject);
    procedure btnExitClick(Sender: TObject);
    procedure mmInpExit(Sender: TObject);
    procedure mmInpKeyPress(Sender: TObject; var Key:
Char);

  private
    { Private declarations }
    function power(const Base, Exponent: integer):
integer;
  public
    { Public declarations }
  end;

var
  frmDM001: TfrmDM001;

implementation

uses HelpUnit;

{$R *.dfm}

function TfrmDM001.power(const Base, Exponent: inte-
ger): integer;
var
i: integer;
begin
  result:=1;
  for i:=1 to Exponent do
    result:=result*Base;
end;

procedure TfrmDM001.edtDivKeyPress(Sender: TObject;
var Key: Char);
begin
  if not (Key in ['0'..'9', #8]) then begin
    Key:=#0;
    Beep;
  end;
end;


procedure TfrmDM001.edtNumDigitChange(Sender: TOb-
ject);
begin
  if (length(edtNumDigit.Text)>0) and
     (length(edtLost.Text)>0) then
    btnRun.Enabled:=true
  else
    btnRun.Enabled:=false;
end;




procedure TfrmDM001.edtlostExit(Sender: TObject);
var
i:integer;
begin
  if (length(edtLost.Text)>0) then begin
    For I:=1 to mmInp.Lines.Count-1 do begin
    if StrToInt(edtLost.Text)>=StrToInt(mmInp.
Lines[i])
    then
    begin
      MessageDlg('Остаток должен быть меньше
делителя', mtError, [mbOK], 0);
      edtLost.SetFocus;
    end;
    end;
    if StrToInt(edtLost.Text)>=StrToInt(mmInp.
Lines[0])
    then
    begin
      MessageDlg('Остаток должен быть меньше
делителя', mtError, [mbOK], 0);
      edtLost.SetFocus;
    end;
  end;
end;

procedure TfrmDM001.btnRunClick(Sender: TObject);
var
nDigit, nLost: integer;
nMin, nMax: integer;
nCount: integer;
Stl,Finl:integer;
Dig: array of integer;
I, Max, Min, J, NOK: integer;
P: Int64;
Bul, mBul:Boolean;
begin
  edtResult.Text:=IntToStr(0);
  mmResultList.Lines.Clear;
  nDigit:=StrToInt(edtNumDigit.Text);
  nLost:=StrToInt(edtLost.Text);
  if nDigit<=1 then
    begin
    nMin:=0;
    nMax:=9;
    end
  else
    begin
    nMin:=power(10, nDigit-1);
    nMax:=nMin*10-1;
    end;

   if nlost>= nMin then mBul:=true;

   For I:=1 to mmInp.Lines.Count-1 do begin
      if StrToInt(mmInp.Lines[I])> nMax then begin
      MessageDlg('Делитель должен быть меньше
наибольшего числа данной разрядности', mtError,
[mbOK], 0);
      end;
      end;

  NOK:=0;
  if  mmInp.Lines.Count-1<=6 then begin
    Setlength(Dig,mmInp.Lines.Count);

      Dig[0]:=StrToInt(mmInp.Lines[0]);
      Min:=Dig[0];
      Max:=Dig[0];
      P:=Dig[0];
    For I:=1 to mmInp.Lines.Count-1 do begin
      Dig[I]:=StrToInt(mmInp.Lines[I]);
      P:=P*Dig[I];
      if Dig[I]<Min then
      Min:=Dig[I];
      if Dig[I]>Max then
      Max:=Dig[I];
    end;
    For J:=Min downto 1 do begin
      if Min mod j=0 then begin
        Bul:=true;
        For I:=0 to mmInp.Lines.Count-1 do begin
          if Dig[I] mod J<>0 then begin
            Bul:=false;
            Break;
         end;
        end;
      if Bul then  begin

       Break;
      end;
      end;
    end;
    J:=Max;
    While J<=P do begin
      Bul:=true;
      For I:=0 to mmInp.Lines.Count-1 do begin
        if J mod Dig[I]<>0 then begin
          Bul:=false;
         Break;
        end;
      end;
     if Bul then  begin
      NOK:=J;
      Break;
```

```
     end;
     J:=J+Max;
   end;
   end;

   If NOK > nMax then begin
    mmResultList.text:=' ';
    end
   else begin

   if mBul then
      begin
      Stl:=nMin mod NOK;
      Finl:=nMax mod NOK;
      nMin:=nMin+NOK-Stl+nLost;
      nMax:=nMax-NOK+nlost-Finl;
      end
   else
   begin
  Stl:=nMin mod NOK;
  If Stl>nLost then
    nMin:=nMin+NOK-Stl+nLost
  else
    nMin:=nMin+NOK-Stl;

  Finl:=nMax mod NOK;
  If Finl<nLost then
    nMax:=nMax-NOK+nLost-Finl
  Else
    nMax:=nMax-Finl+NOK;
  end;

  nCount:=(nMax-nMin+NOK) div NOK;
  edtResult.Text:=IntToStr(nCount);
  if cbViewList.Checked then begin
    try
      mmResultList.Lines.BeginUpdate;
      Screen.Cursor:=crAppStart;
      while nMin <= nMax do begin
        mmResultList.Lines.Append(IntToStr(nMin));
        nMin:=nMin+NOK;
      end;
    finally
      Screen.Cursor:=crDefault;
      mmResultList.Lines.EndUpdate;
    end;
  end;
  end;
end;


procedure TfrmDM001.btnHelpClick(Sender: TObject);
begin
ShowMessage('Эта программа определяет мощность
множества с заданными параметрами'+ #13#10 +
'Для этого введите разряд чисел, делители и остаток
от деления и нажмите"Считать!".');
end;

procedure TfrmDM001.btnExitClick(Sender: TObject);
begin
  ((Sender as TButton).Owner as TForm).Close;
end;

procedure TfrmDM001.mmInpExit(Sender: TObject);

var
i:integer;
begin

{For I:=1 to mmInp.Lines.Count do begin
    if StrToInt(edtLost.Text)>=StrToInt(mmInp.
Lines[i]) then begin
    if (length(edtLost.Text)>0) and
     (length(mmInp.Lines[i])>0) then begin
    if StrToInt(edtLost.Text)>=StrToInt(mmInp.
Lines[i]) then
      edtLost.Text:=IntToStr(StrToInt(mmInp.
Lines[i])-1);
  end;
  end; }
  For I:=1 to mmInp.Lines.Count do begin
  if (length(mmInp.Lines[i])>0) then begin
    If StrToInt(mmInp.Lines[i])<=0 then begin
      MessageDlg('Делитель должен быть больше 0',
mtError, [mbOK], 0);
    end;
  end;
  end;

end;

procedure TfrmDM001.mmInpKeyPress(Sender: TObject;
var Key: Char);
begin
   if not (Key in ['0'..'9', #8, #13]) then begin
    Key:=#0;
    Beep;
  end;
end;

end.


// Factorizator

unit dm002Unit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls, Menus;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Label2: TLabel;
    Button1: TButton;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    Memo1: TMemo;
    procedure Button1Click(Sender: TObject);
    procedure Edit1KeyPress(Sender: TObject; var Key:
Char);
    procedure FormCreate(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure N1Click(Sender: TObject);
    procedure Edit1Exit(Sender: TObject);
    //procedure Edit1Change(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

//uses dm002Unit;

{$R *.dfm}



procedure TForm1.Edit1KeyPress(Sender: TObject; var
Key: Char);
begin
    case Key of
      '0'..'9':               ; //
      #8      :               ; // <Backspace>
      #13     : Button1.SetFocus;

      else Key :=Chr(0);
    end;

end;
```

```
procedure TForm1.Button1Click(Sender: TObject);


function pow(a,x:longint):longint;
var
t,i:longint;
begin
t:=a;
for i:=1 to x-1 do
t:=t*a;
pow:=t;
end;  {pow}

var
   numb, powers: array [1..100] of integer;
    ch: integer;
   c1: longint;
   n: longint;
   n1: longint;
   i: longint;
   h,k: longint;
   sum: longint;
   T:longint;
begin

   memo1.text := ' ';
   ch := StrToInt(Edit1.Text);
   if ch=0 then
   MessageDlg('Число должно быть больше 0', mtError,
[mbOK], 0)
   else
   begin

   c1:=ch;
   n:= 1;
   n1:= 0;
   while ch <> 1 do
   begin
      i:= 2;
      while ch mod i <> 0 do
      Inc(i);
      Inc(n1);
      if n1 = 1 then
      begin
         numb[n]:= i;
         powers[n]:= 1;
      end
      else if numb[n] = i then Inc(powers[n])
      else
      begin
         Inc(n);
         numb[n]:= i;
         powers[n]:= 1;
      end;
      ch:= ch div i;
   end;

   memo1.text := memo1.text+ ' ' + IntToStr(c1)+' =
';
   k:=1;
   T:=1;
   for i:= 1 to n do
   begin
     memo1.text := memo1.text+ ' '  +
IntToStr(numb[i])+'^' + IntToStr(powers[i]);
     k:=k*((pow(numb[i],powers[i]+1) - 1) div
(numb[i] - 1));
     t:=t*(powers[i]+1);
     if i <> n then
        begin
        memo1.text := memo1.text+ ' ' +' * '+' ';
        end;
   end;
   memo1.text := memo1.text+ chr(13) + chr(10)+
chr(13) + chr(10);


   memo1.text := memo1.text + ' Количество делителей'
+ 'T(' + IntToStr(c1)+')= '+IntToStr(T)+ chr(13) +
chr(10)+ chr(13) + chr(10);

   memo1.text := memo1.text+ ' Множество делителей '
+ 'D(' + IntToStr(c1)+')= {';

   for h:=1 to c1 do
      begin
      if c1 mod h=0 then
         begin
         memo1.text := memo1.text + ' ' +
IntToStr(h)+', ' ;
         end;
      end;
   memo1.text := memo1.text +'}'+ chr(13) + chr(10)+
chr(13) + chr(10);

   memo1.text := memo1.text+ ' Сумма делителей ' +
'S(' + IntToStr(c1)+')= ';
      sum:=0;
   for h:=1 to c1 do
      begin
      if c1 mod h=0 then
         begin
         sum:=sum+h;
         end;
      end;
   memo1.text := memo1.text +IntToStr(sum);
end;

end;


procedure TForm1.FormCreate(Sender: TObject);
begin
memo1.text := '';
end;

procedure TForm1.N2Click(Sender: TObject);
begin
ShowMessage('МАИ, 3 факультет, 2010
год'+#13#10+'ДМДЗ308.03, гр 03-119, каф
308,'+#13#10+'Студент: Злобин Д.В.,' +#13#10+
'Преподаватель: к.т.н. Гридин.А.Н');
        exit;
end;

procedure TForm1.N1Click(Sender: TObject);
begin
ShowMessage('Эта программа выполняет факторизацию
чисел, находит все делители числа, их сумму и
количество'+ #13#10 +'Для этого введите число от 1 до
1000000000 в поле и нажмите "Считать!');
end;

procedure TForm1.Edit1Exit(Sender: TObject);
begin
//if (StrToInt(Edit1.Text) <> 0) then
begin
  if (StrToInt(Edit1.Text) > 1000000000) or
(StrToInt(Edit1.Text) < 0) then
    begin
    ShowMessage('Число должно быть меньше
1000000000');
    Edit1.SetFocus;
    end;
end;
end;

end.



// NOD_NOK

unit DM003Unit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
```

```
Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TfrmNumer = class(TForm)
    mmInp: TMemo;
    lblInp: TLabel;
    edtNOD: TEdit;
    edtNOK: TEdit;
    btnResult: TButton;
    btnHelp: TButton;
    lblNOD: TLabel;
    lblNOK: TLabel;
    btnToFile: TButton;
    procedure mmInpKeyPress(Sender: TObject; var Key:
Char);
    procedure btnResultClick(Sender: TObject);
    procedure mmInpExit(Sender: TObject);
    procedure mmInpChange(Sender: TObject);
    procedure btnHelpClick(Sender: TObject);
    procedure btnExitClick(Sender: TObject);
    procedure btninfClick(Sender: TObject);
    procedure btnToFileClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmNumer: TfrmNumer;

implementation

uses CreateUnit, HelpUnit;

{$R *.dfm}

procedure TfrmNumer.mmInpKeyPress(Sender: TObject;
var Key: Char);
begin
if not (Key in ['0'..'9', #8,#13,#10]) then begin
    Key:=#0;
    Beep;
  end;
end;

procedure TfrmNumer.btnResultClick(Sender: TObject);
Var
  Dig: array of integer;
  I, Max, Min, J, NOD, NOK: integer;
  P: Int64;
  Bul:Boolean;
begin
  NOD:=0;
  NOK:=0;
  if  mmInp.Lines.Count-1<=6 then begin
    Setlength(Dig,mmInp.Lines.Count);
    try
      Dig[0]:=StrToInt(mmInp.Lines[0]);
      Min:=Dig[0];
      Max:=Dig[0];
      P:=Dig[0];
    For I:=1 to mmInp.Lines.Count-1 do begin
      Dig[I]:=StrToInt(mmInp.Lines[I]);
      P:=P*Dig[I];
      if Dig[I]<Min then
      Min:=Dig[I];
      if Dig[I]>Max then
      Max:=Dig[I];
    end;
    For J:=Min downto 1 do begin
      if Min mod j=0 then begin
        Bul:=true;
        For I:=0 to mmInp.Lines.Count-1 do begin
          if Dig[I] mod J<>0 then begin
            Bul:=false;
            Break;
         end;
        end;
      if Bul then  begin
        NOD:=J;
        Break;
      end;
      end;
    end;
    J:=Max;
    While J<=P do begin
      Bul:=true;
      For I:=0 to mmInp.Lines.Count-1 do begin
        if J mod Dig[I]<>0 then begin
          Bul:=false;
         Break;
        end;
      end;
     if Bul then  begin
      NOK:=J;
      Break;
     end;
     J:=J+Max;
   end;
   edtNOD.Text:=IntToStr(NOD);
   edtNOK.Text:=IntToStr(NOK);
//   if cbPrint.Checked then begin
  finally
    Dig:=nil;
  end;
  end else begin
    MessageDlg('Количество чисел должно быть не
больше 6', mtError, [mbOK], 0);
  end;
end;

procedure TfrmNumer.mmInpExit(Sender: TObject);
var
I:integer;
begin
  if (length(mmInp.Lines.Text)>0) and (mmInp.Lines.
Count>1) then begin
    For I:=0 to mmInp.Lines.Count-1 do begin
      if length(mmInp.Lines[I])>0 then begin
        if StrToInt(mmInp.Lines[I])<1 then begin
          MessageDlg('Число должно быть не меньше 1',
mtError, [mbOK], 0);
          mmInp.Lines[I]:='1';
          mmInp.SetFocus;
          break;
        end;
        if StrToInt(mmInp.Lines[I])>100000  then be-
gin
          MessageDlg('Число должно быть меньше
100000', mtError, [mbOK], 0);
          mmInp.Lines[I]:='100000';
          mmInp.SetFocus;
          break;
        end;
      end else begin
        MessageDlg('Строка должна быть заполнена',
mtError, [mbOK], 0);
        mmInp.Lines[I]:='1';
        mmInp.SetFocus;
        break;
      end
    end;
  end;
end;

procedure TfrmNumer.mmInpChange(Sender: TObject);
begin
  if (length(mmInp.Lines.Text)>0) and (mmInp.Lines.
Count>1)then begin
    btnToFile.Enabled:=True;
    btnResult.Enabled:=True;
  end else begin
    btnResult.Enabled:=False;
    btnToFile.Enabled:=False;
  end;
end;

procedure TfrmNumer.btnHelpClick(Sender: TObject);
begin
ShowMessage('Эта программа находит НОК и НОД заданной
```

```
совокупности чисел'+ #13#10 +
'Для этого в соответсвующем поле введите числа и
нажмите "Считать!".'+ #13#10 +
'Также можно сохранить результаты в отдельном файле с
помощью кнопки "Печать".');
end;
procedure TfrmNumer.btnExitClick(Sender: TObject);
begin
  ((Sender as TButton).Owner as TForm).Close;
end;

procedure TfrmNumer.btninfClick(Sender: TObject);
var
  frmCreate: TfrmCreate;
begin
  frmCreate:=TfrmCreate.Create(Application);
  try
    frmCreate.ShowModal;
  finally
    frmCreate.Free;
  end;
end;
procedure TfrmNumer.btnToFileClick(Sender: TObject);
var
slText: TStringList;
tStr: string;
i: Integer;
begin
  slText:=TStringList.Create;
  try
    tStr:='×èñëà: ';
    for i:=0 to mmInp.Lines.Count-1 do begin
      if i>0 then
        tStr:=tStr+', ';
      tStr:=tStr+mmInp.Lines[i];
    end;
    slText.Append(tStr);
    slText.Append('НОД='+edtNOD.Text);
    slText.Append('НОК='+edtNOK.Text);
    slText.SaveToFile('DM003File.txt');
  finally
    slText.Free;
  end;
end;

end.

//SuperHorner

unit DM005Unit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls;

type
  TfrmSuperGorner = class(TForm)
    edtPower: TEdit;
    mmInp: TMemo;
    mmResult: TMemo;
    btnResult: TButton;
    lblPower: TLabel;
    lblInp: TLabel;
    lblResult: TLabel;
    btnHelp: TButton;
    btnPrint: TButton;
    edtTest: TEdit;
    lblTest: TLabel;
    udPower: TUpDown;
    procedure mmInpKeyPress(Sender: TObject; var Key:
Char);
    procedure btnResultClick(Sender: TObject);
    procedure btnExitClick(Sender: TObject);
    procedure btnPrintClick(Sender: TObject);
    procedure btnHelpClick(Sender: TObject);


  private
    { Private declarations }
  public
    { Public declarations }
    function  NOD(a:integer;b:integer):integer;
  end;

var
  frmSuperGorner: TfrmSuperGorner;

implementation

uses HelpUnit, CreateUnit, DM004Unit;

{$R *.dfm}
function  TfrmSuperGorner.NOD(a:integer;b:integer):i
nteger;
var
  T:integer;
begin
  a:=ABS(a);
  b:=ABS(b);
  if a>b then begin
    T:=a;
    a:=b;
    b:=T;
  end;
  WHILE b mod a<>0 do begin
    T:=a;
    a:=b mod a;
    b:=T;
  end;
  result:=a;
end;
procedure TfrmSuperGorner.mmInpKeyPress(Sender: TOb-
ject; var Key: Char);
begin
  if not (Key in ['0'..'9', #8,#13,#10,'-']) then
begin
    Key:=#0;
    Beep;
  end;
end;

procedure TfrmSuperGorner.btnResultClick(Sender: TO-
bject);
const
  K=11;
var
  a:array[1..K] of real;
  x, i, n, L, j, p, T, R, M, s:integer;
  y, result:extended;
  Bol:boolean;
begin
  mmResult.Lines.Clear;
  n:=StrToInt(edtPower.Text);
  n:=n+1;
  if n>mmInp.Lines.Count then begin
    MessageDlg('Введите '+IntToStr(n-mmInp.Lines.
Count)+' коэффицента(-ов) уравнения', mtError,
[mbOK], 0);
    mmInp.SetFocus;
  end else begin
    try
      mmResult.Lines.BeginUpdate;
      Screen.Cursor:=crAppStart;
      if (length(mmInp.Lines[0])>0)and
         (StrToInt(mmInp.Lines[0])<>0)then begin
        for i:=1 to n do
          a[i]:=StrToFloat(mmInp.Lines[i-1]);
        for T:=n downto 2 do
          if a[T]<>0 then
            break;
        y:=Abs(a[T]);
        if T<2 then begin
          mmResult.Lines.Append('0'+'
'+IntToStr(n-T));
        end else begin
          if T<n then
            if T=n-1 then
              mmResult.Lines.Append('0')
```

```
            else
               mmResult.Lines.Append('0'+'
'+IntToStr(n-T));
          p:=1;
          While (p<=ABS(a[1])) and (n>1) do begin
            if trunc(a[1]) mod p=0 then begin
              i:=1;
              While (i<=y) and (n>1)do begin
                if (trunc(y) mod i=0)and ((i<>p)
or(p=1)) then begin
                  x:=-i;
                  for L:=1 to 2 do begin
                    result:=a[1];
                    for j:=2 to n do begin
                           result:= result*x/p;
                           result:= result + a[j];
                    end;{forj}
                    if result=0 then begin
                      R:=NOD(x,p);
                      x:=x div R;
                      M:=p div R;
                      s:=1;

                      if s=1 then begin
                        if M>1 then
                          mmResult.Lines.Append(IntTo
Str(x)+'/'+IntToStr(M))
                        else
                          mmResult.Lines.
Append(IntToStr(x));
                      end else begin
                        if M>1 then
                          mmResult.Lines.Append(IntTo
Str(x)+'/'+IntToStr(M)+'    '+IntToStr(s))
                        else
                          mmResult.Lines.
Append(IntToStr(x)+'    '+IntToStr(s));
                      end ;
                    end;
                    x:=(-1)*x;
                  end;{forl}
                end;{if}
                i:=i+1;
              end;
            end;{if}
            p:=p+1;

          end;{while}
         end;

      end else begin
        if length(mmInp.Lines[0])=0 then begin
           MessageDlg('Введите коэффицент старшей
степени', mtError, [mbOK], 0);
           mmInp.SetFocus;
        end else begin
           MessageDlg('Коэффицент старшей степени
должен быть отличен от нуля', mtError, [mbOK], 0);
           mmInp.SetFocus;
        end;
      end;
    finally
      mmResult.Lines.EndUpdate;
      Screen.Cursor:=crDefault;
    end;{finally}
    if length(mmResult.Lines[0])=0  then
      mmResult.Lines.Append('Íåò êîðíåé');
  end;
end;

procedure TfrmSuperGorner.btnExitClick(Sender: TOb-
ject);
begin
 //frmSuperGorner.Close;
 ((Sender as TButton).Owner as TForm).Close;
end;

procedure TfrmSuperGorner.btnPrintClick(Sender: TOb-
ject);
var
slText: TStringList;
tStr: string;
i:integer;
begin
  slText:=TStringList.Create;
  try
    for i:=0 to mmResult.Lines.Count-1 do
      slText.Append(mmResult.Lines[i]);
    tStr:=TimeToStr(Time);
    tStr:=tStr+'   '+DateToStr(Date);
    slText.Append(tStr);
    slText.SaveToFile(edtTest.Text);
  finally
    slText.Free;
  end;
end;

procedure TfrmSuperGorner.btnHelpClick(Sender: TOb-
ject);

begin
ShowMessage('Эта программа находит целочисленные
решения алгебраического уравнения, используя схему
Горнера'+ #13#10 +
'Для этого введите степень уравнения и коэффициенты по
убыванию степеней и нажмите "Считать".'+ #13#10 +
'Также можно сохранить результаты в отдельном файле с
помощью кнопки "Печать".');
end;



end.


//Expressor

unit DM007Unit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TForm2 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Button2: TButton;
    Memo1: TMemo;
    Button1: TButton;
    procedure Edit1KeyPress(Sender: TObject; var Key:
Char);
    procedure Edit2KeyPress(Sender: TObject; var Key:
Char);
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}


procedure TForm2.Edit2KeyPress(Sender: TObject; var
Key: Char);
begin
case Key of
      '0'..'9':              ;
      #8     :               ; // <Backspace>
      #13    : Button2.SetFocus; // <Enter>
```

```
      else Key :=Chr(0);      end;
end;

procedure TForm2.Button2Click(Sender: TObject);



var
   a,b,t:integer;
begin

a := StrToInt(Edit1.Text);
b := StrToInt(Edit2.Text);
if b<=0 then
begin
ShowMessage('Знаменатель должен быть больше 0');
end
else
begin

memo1.text := ' [ ';
while (a mod b>0) do
   begin
   memo1.text := memo1.text + IntToStr(a div b)+ ',
';
   a:=a mod b;
   t:=b;
   b:=a;
   a:=t;
   end;

memo1.text := memo1.text + IntToStr(a div b)+ ' ]';
end;
end;

procedure TForm2.Edit1KeyPress(Sender: TObject; var
Key: Char);
begin


    case Key of
      '0'..'9':              ;
      #8      :              ; // <Backspace>
      #13     : Edit2.SetFocus; // <Enter>


      else Key :=Chr(0);

end;

end;
procedure TForm2.Button1Click(Sender: TObject);
begin
ShowMessage('Эта программа преобразовывает обычные
дроби в цепные.'+ #13#10 +
'Для этого в соответсвующих полях введите числитель и
знаменатель.'
+ #13#10 +'и нажмите "Считать!".');
end;

end.

// Antiexpressor

unit DM008Unit;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TAntiexpressor = class(TForm)
    Label1: TLabel;
    Help: TButton;
    Run: TButton;
    mmInp: TMemo;
    Memo1: TMemo;
    edtPower: TEdit;
    Label2: TLabel;
    Label3: TLabel;
    procedure Memo1KeyPress(Sender: TObject; var Key:
Char);
    procedure edtPowerKeyPress(Sender: TObject; var
Key: Char);
    procedure RunClick(Sender: TObject);
    procedure HelpClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Antiexpressor: TAntiexpressor;

implementation

{$R *.dfm}

uses HelpUnit, CreateUnit;

procedure TAntiexpressor.edtPowerKeyPress(Sender: TO-
bject; var Key: Char);
begin

    case Key of
      '0'..'9':              ;
      #8      :              ; // <Backspace>
      #13     : Memo1.SetFocus; // <Enter>
      else Key :=Chr(0);
end;

end;

procedure TAntiexpressor.Memo1KeyPress(Sender: TOb-
ject; var Key: Char);
begin
if not (Key in ['0'..'9', #8, #13]) then begin
    Key:=#0;
    Beep;
end;
end;

procedure TAntiexpressor.RunClick(Sender: TObject);
var s: array [0..100] of integer;
   a,b,t,
   i, n, j:integer;
   bul:boolean;
begin

Memo1.Lines.Clear;
 n:=StrToInt(edtPower.Text);

       for i:=0 to n-1 do
       begin
         s[i]:=StrToInt(mmInp.Lines[i]);
       end;

for j:= 1 to n-1 do
  begin
  if  StrToInt(mmInp.Lines[j])<=0 then
    begin
    bul:=false;
    MessageDlg('Элементы цепной дроби, кроме первого,
должны быть>0', mtError, [mbOK], 0);
    mmInp.SetFocus;
    break;
    end
  else
    begin
    bul:=true;
    a:=1;
    b:=s[n-1];
    for i:= n-1 downto 1 do
      begin
```

```
      t:=s[i-1]*b+a;
      a:=b;
      b:=t;
      end;


   // memo1.text:=memo1.Text+ #13#10+IntToStr(b) +  '
/ '+ IntToStr(a);
    end;

end;
if bul then
 memo1.text:=memo1.Text+ #13#10+IntToStr(b) +  ' / '+
IntToStr(a)
 else
 memo1.Text:=' ';
end;


procedure TAntiexpressor.HelpClick(Sender: TObject);
begin
ShowMessage('Эта программа переводит цепные дроби в
обыкновенные'+ #13#10 +
'Для этого введите элементы цепной дроби и нажмите
"Считать!".');

end;

end.
```