МИНОБРНАУКИ РФ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ИНСТИТУТ РАДИОТЕХНИКИ,

ЭЛЕКТРОНИКИ И АВТОМАТИКИ (ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)

Факультет ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА ТИССУ

**Курсовая работа**

**по теме:**

***Объектно-ориентированное программирование.***

***Объект - «Рациональная дробь».***

ДИСЦИПЛИНА

“***Технология программирования***”.

Группа: **ВАИ-2-10**

Студент: **Мошкалев Д.В.**

Руководитель: **Колесникова М.Д.**

МОСКВА, 2012 г.

Руководитель: Колесникова М.Д.

Рецензент: Колесникова М.Д.

Мошкалев Д.В.

Курсовая работа по специальности 230201 «Информационные системы и технологии»: М. 2012 г., МИРЭА, факультет Информационных технологий, кафедра ТИССУ. – стр.?? рис ??, табл. ??.

*Объект «Рациональная дробь».* В курсовой работе.

©Мошкалев Д.В.

Файл: «Курсовая\_Работа\_Мошкалев\_(ВАИ-2-10)\_\_.doc», исполнитель Мошкалев Д.В., 2012г.

Оглавление

[1. Техническое задание 4](#_Toc323124740)

[4. Введение 5](#_Toc323124741)

[5. Технический и рабочий проект программной системы 5](#_Toc323124742)

[5.1. Постановка задачи 5](#_Toc323124743)

[5.2. Разработка функциональной структуры программной системы 5](#_Toc323124744)

[5.3. Реализация диалогового интерфейса 6](#_Toc323124745)

[5.4. Разработка и реализация алгоритмов решения 9](#_Toc323124746)

[7. Эксплуатационная документация 11](#_Toc323124747)

[7.1. Описание применения 11](#_Toc323124748)

[7.2. Руководство оператора 11](#_Toc323124749)

[8. Заключение 11](#_Toc323124750)

[9. Приложение 1: Распечатка текстов программы 12](#_Toc323124751)

[9.1. Файл программы 12](#_Toc323124752)

[9.2. Модуль RatFracClass 12](#_Toc323124753)

[9.3. Модуль RatFracForm 16](#_Toc323124754)

[9.4. Модуль LogForm 25](#_Toc323124755)

[9.5. Модуль AboutForm 26](#_Toc323124756)

[9.6. Файл тестовых данных 26](#_Toc323124757)

# Техническое задание

**Задание на курсовую работу по дисциплине**

**«Технология программирования»**

**Вариант №16**

**Тема:** *Объектно-ориентированное программирование. Объект - «Рациональная дробь».*

#### Содержательная задача:

* 1. Составить описание класса «Рациональная дробь» для представления дроби числителем и знаменателем целого типа. Обеспечить выполнение операций сложения, вычитания, умножения, деления, сравнения двух дробей. Результаты операции получать в виде несократимых дробей.
  2. Разработать приложение для Windows в среде Delphi, реализующее вычисление следующих дробей:

В приложение включить средства, позволяющие выбрать вычисление «нужной» функции.

#### Специальные требования:

* 1. Исходные данные для тестирования приложения подготовить в текстовых файлах.
  2. Результаты тестирования представить в элементах диалоговых форм.

# Введение

# Технический и рабочий проект программной системы

## Постановка задачи

Целью программы является:

* + реализация более точных математических вычислений с помощью представления дробного числа через отношение двух целых чисел,
  + предоставление простого программного интерфейса для выполнения вычислений
  + оформление необходимых программ в виде объекта

## Разработка функциональной структуры программной системы



Рисунок 1. Функциональная схема класса

## Реализация диалогового интерфейса

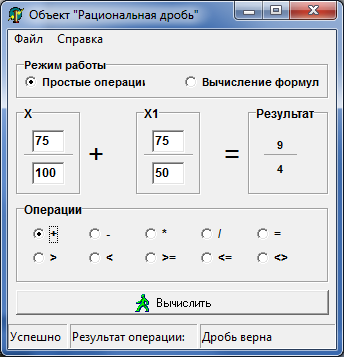


Рисунок . Режим простых операций

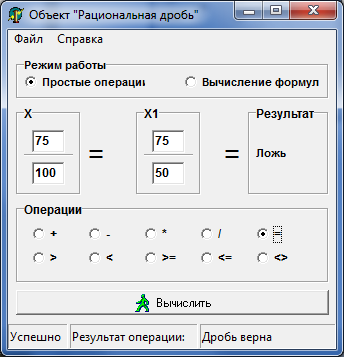


Рисунок . Режим операций сравнения

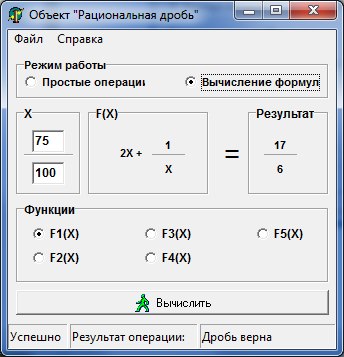


Рисунок . Режим вычисления формул

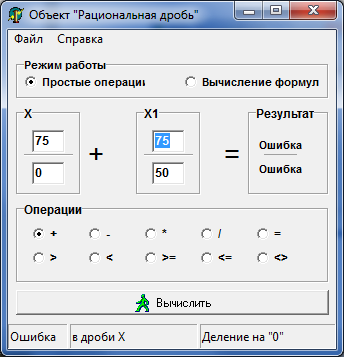


Рисунок . Реакция на ошибки в вычислениях

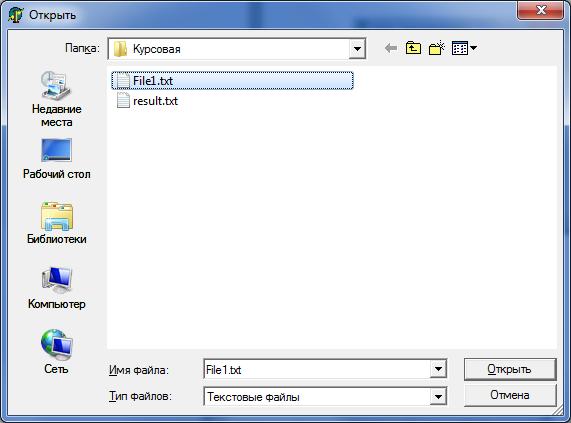


Рисунок . Диалог выбора файла исходных данных для тестов

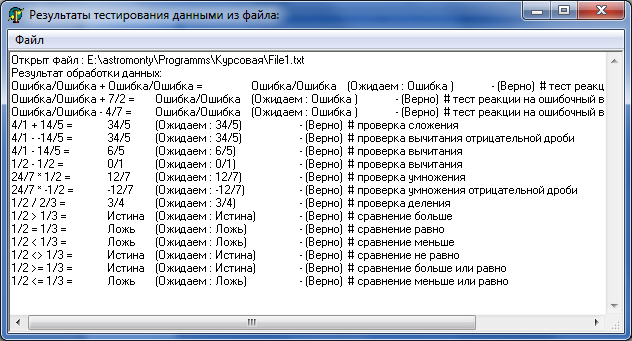


Рисунок . Форма результатов обработки файла тестовых данных

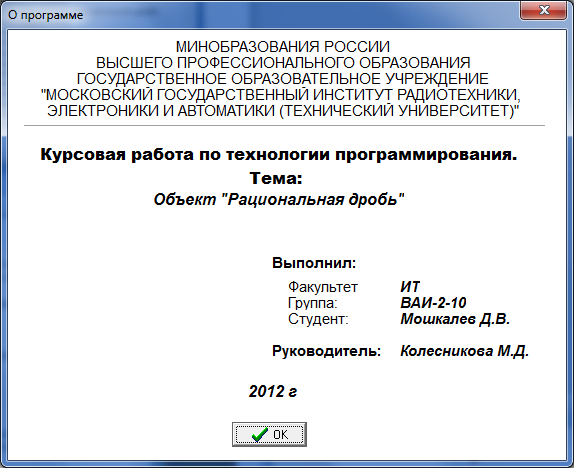


Рисунок . Форма с информацией о программе

## Разработка и реализация алгоритмов решения



Рисунок 9. Нахождение Наибольшего общего делителя бинарным алгоритмом Евклида



Рисунок 10. Приведение дроби к несократимому виду



Рисунок 11. Операции сложения, вычитания, деления, умножения

# Эксплуатационная документация

## Описание применения

## Руководство оператора

# Заключение

# Приложение 1: Распечатка текстов программы

## Файл программы

program RatFrac;

**uses**

Forms,

RatFracForm in 'RatFracForm.pas' {Form1},

RatFracClass in 'RatFracClass.pas',

AboutForm in 'AboutForm.pas' {Form2},

LogForm in 'LogForm.pas' {Form3};

*{$R \*.res}*

**begin**

Application.Initialize;

Application.CreateForm(TForm1, Form1);

Application.CreateForm(TForm2, Form2);

Application.CreateForm(TForm3, Form3);

Application.Run;

**end**.

## Модуль RatFracClass

**unit** RatFracClass;

**interface**

**uses** SysUtils;

**type**

TRationalFraction = **class**

**private**

FNumerator : **integer**; *// числитель*

FDenominator : **integer**; *// знаменатель*

FErrorFlag : **boolean**; *// флаг ошибки*

*// функции получения значений свойств*

**function** GetNumStr : **string**;

**function** GetDenomStr : **string**;

**function** GetErrorStr : **string**;

*// служебные функции*

**function** Nod( a, b : **integer**) : **integer**;

**procedure** Normalize;

**public**

*// свойства*

**property** Numerator : **integer** read FNumerator;

**property** Denominator : **integer** read FDenominator;

**property** StrNumerator : **string** read GetNumStr;

**property** StrDenominator : **string** read GetDenomStr;

**property** ErrorFlag : **boolean** read FErrorFlag;

**property** ErrorStr : **string** read GetErrorStr;

*// Конструкторы*

**constructor** Create(NumValue, DenumValue : **integer**); Overload;

**constructor** Create( Value : TRationalFraction); Overload;

**constructor** Create; Overload;

*// методы задания значений*

**procedure** SetValue(NumValue, DenumValue : **integer**); Overload;

**procedure** SetValue( Value : TRationalFraction); Overload;

*// методы операций*

**procedure** Multiply( Factor : TRationalFraction); Overload;

**procedure** Divide( Factor : TRationalFraction); Overload;

**procedure** Add( Factor : TRationalFraction); Overload;

**procedure** Sub( Factor : TRationalFraction); Overload;

*// методы сравнения*

**function** Eq( Factor : TRationalFraction ):**boolean**; Overload; //=

**function** Gt( Factor : TRationalFraction ):**boolean**; Overload; // >

**function** Lt( Factor : TRationalFraction ):**boolean**; Overload; // <

**function** Ne( Factor : TRationalFraction ):**boolean**; Overload; //<>

**function** Ge( Factor : TRationalFraction ):**boolean**; Overload; //>=

**function** Le( Factor : TRationalFraction ):**boolean**; Overload; //<=

**end**;

**implementation**

*{ TRationalFraction }*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// Конструкторы*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**constructor** TRationalFraction.Create(NumValue, DenumValue: **integer**);

**begin**

FErrorFlag := **false**;

SetValue( NumValue, DenumValue );

**end**;

**constructor** TRationalFraction.Create(Value: TRationalFraction);

**begin**

FErrorFlag := **false**;

SetValue( Value );

**end**;

**constructor** TRationalFraction.Create;

**begin**

FErrorFlag := **false**;

FNumerator := 0;

FDenominator := 1;

**end**;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Установка значений

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**procedure** TRationalFraction.SetValue(NumValue, DenumValue: **integer**);

**begin**

FNumerator := NumValue;

FDenominator := DenumValue;

Normalize;

**end**;

**procedure** TRationalFraction.SetValue(Value: TRationalFraction);

**begin**

FNumerator := Value.FNumerator;

FDenominator := Value.FDenominator ;

Normalize;

**end**;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// умножение*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**procedure** TRationalFraction.Multiply(Factor: TRationalFraction);

**begin**

FNumerator := FNumerator \* Factor.FNumerator;

FDenominator := FDenominator \* Factor.FDenominator ;

Normalize;

**end**;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// деление двух дробей*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**procedure** TRationalFraction.Divide(Factor: TRationalFraction);

**begin**

FNumerator := FNumerator \* Factor.FDenominator ;

FDenominator := FDenominator \* Factor.FNumerator;

Normalize;

**end**;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// Сложение*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**procedure** TRationalFraction.Add( Factor : TRationalFraction);

**begin**

**if**(Factor.FErrorFlag) **then** FErrorFlag := **true** **else**

**begin**

FNumerator := FNumerator \* Factor.FDenominator + Factor.FNumerator \* FDenominator ;

FDenominator := FDenominator \* Factor.FDenominator ;

Normalize;

**end**;

**end**;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// Вычитание*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**procedure** TRationalFraction.Sub( Factor : TRationalFraction);

**begin**

**if**(Factor.FErrorFlag) **then** FErrorFlag := **true** **else**

**begin**

FNumerator := FNumerator \* Factor.FDenominator - Factor.FNumerator \* FDenominator ;

FDenominator := FDenominator \* Factor.FDenominator ;

Normalize;

**end**;

**end**;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// Нормализация*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**procedure** TRationalFraction.Normalize;

**var**

a : **integer**;

**begin**

a := Nod( **abs**(FNumerator), **abs**(FDenominator ));

**if** FDenominator = 0 **then** FErrorFlag:=**true** **else** FErrorFlag := **false**;

**if**( a = 0 ) **then** **exit**;

FNumerator := FNumerator div a;

FDenominator := FDenominator div a;

**if** (FNumerator < 0) **and** (FDenominator < 0) **then**

**begin**

FNumerator := **abs**(FNumerator);

FDenominator := **abs**(FDenominator );

**end**;

**if** (FDenominator < 0) **and** (FNumerator > 0) **then**

**begin**

FDenominator := **abs**(FDenominator );

FNumerator := -FNumerator;

**end**;

**if**(FNumerator = 0) **then** FDenominator := 1;

**end**;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// Нахождение Наибольшего общего делителя*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**function** TRationalFraction.Nod( a, b : **integer**) : **integer**;

**var**

m, n, r, k: **integer**;

**begin**

m := a;

n := b;

r := 0;

**repeat**

**if** m = 0 **then** **begin** Nod := n **Shl** r; **break**; **end**; *// NOD(0,n) = n*

**if** (n = 0) **or** (m = n) **then** **begin** Nod := m **Shl** r; **break**; **end**; *// NOD(0,n) = n*

**if** (m = 1) **or** (n = 1) **then** **begin** Nod := 1 **Shl** r; **break**; **end**;

**if** (m **and** 1 = 0) **then**

**begin**

**if** ( n **and** 1 = 0) **then**

**begin**

r := r **Shl** 1;

m := m **Shr** 1;

n := n **Shr** 1;

**Continue**;

**end** **else**

**begin**

m := m **Shr** 1;

**Continue**;

**end**;

**end** **else**

**begin**

**if** ( n **and** 1 = 0) **then**

**begin**

n := n **Shr** 1;

**Continue**;

**end** **else**

**begin**

**if** (n > m) **then**

**begin**

k := m;

m := (n - m) **Shr** 1;

n := k;

**Continue**;

**end** **else**

**begin**

m := (m - n) **Shr** 1;

**Continue**;

**end**;

**end**;

**end**;

**until** **false**;

**end**;

**function** TRationalFraction.GetNumStr : **string**;

**begin**

**if**(FDenominator = 0) **then** GetNumStr := *'Ошибка'* **else**

GetNumStr := IntToStr(FNumerator);

**end**;

**function** TRationalFraction.GetDenomStr : **string**;

**begin**

**if**(FDenominator = 0) **then** GetDenomStr := *'Ошибка'* **else**

GetDenomStr := IntToStr(FDenominator );

**end**;

**function** TRationalFraction.GetErrorStr : **string**;

**begin**

**if**(FErrorFlag) **then** GetErrorStr := *'Деление на "0"'* **else**

GetErrorStr := *'Дробь верна'*;

**end**;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// Методы сравнения*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**function** TRationalFraction.Eq(Factor: TRationalFraction): **boolean**;

**begin**

**if** (FErrorFlag **or** Factor.FErrorFlag) **then** Eq:=**false** **else**

**if** FNumerator\*Factor.FDenominator = Factor.FNumerator\*FDenominator **then** Eq := **true**

**else** Eq := **false**;

**end**;

**function** TRationalFraction.Ge(Factor: TRationalFraction): **boolean**;

**begin**

**if** (FErrorFlag **or** Factor.FErrorFlag) **then** Ge := **false** **else**

**if** FNumerator\*Factor.FDenominator >= Factor.FNumerator\*FDenominator **then** Ge := **true**

**else** Ge := **false**;

**end**;

**function** TRationalFraction.Gt(Factor: TRationalFraction): **boolean**;

**begin**

**if** (FErrorFlag **or** Factor.FErrorFlag) **then** Gt:=**false** **else**

**if** FNumerator\*Factor.FDenominator > Factor.FNumerator\*FDenominator **then** Gt := **true**

**else** Gt := **false**;

**end**;

**function** TRationalFraction.Le(Factor: TRationalFraction): **boolean**;

**begin**

**if** (FErrorFlag **or** Factor.FErrorFlag) **then** Le:=**false** **else**

**if** FNumerator\*Factor.FDenominator <= Factor.FNumerator\*FDenominator **then** Le := **true**

**else** Le := **false**;

**end**;

**function** TRationalFraction.Lt(Factor: TRationalFraction): **boolean**;

**begin**

**if** (FErrorFlag **or** Factor.FErrorFlag) **then** Lt:=**false** **else**

**if** FNumerator\*Factor.FDenominator < Factor.FNumerator\*FDenominator **then** Lt := **true**

**else** Lt := **false**;

**end**;

**function** TRationalFraction.Ne(Factor: TRationalFraction): **boolean**;

**begin**

**if** (FErrorFlag **or** Factor.FErrorFlag) **then** Ne:=**false** **else**

**if** FNumerator\*Factor.FDenominator <> Factor.FNumerator\*FDenominator **then** Ne := **true**

**else** Ne := **false**;

**end**;

**END**.

## Модуль RatFracForm

**unit** RatFracForm;

**interface**

**uses**

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, Buttons, ExtCtrls, ComCtrls, Menus,

ToolWin, RatFracClass, AboutForm, LogForm;

**type**

TForm1 = **class**(TForm)

StatusBar1: TStatusBar;

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

EnterNum: TEdit;

EnterDenom: TEdit;

EnterNum2: TEdit;

EnterDenom2: TEdit;

F1: TGroupBox;

F2: TGroupBox;

F3: TGroupBox;

F4: TGroupBox;

F5: TGroupBox;

Operand1: TGroupBox;

Operand2: TGroupBox;

Operators: TGroupBox;

ResultGroup: TGroupBox;

GroupBox7: TGroupBox;

Functions: TGroupBox;

SimpleBtn: TRadioButton;

FormulsBtn: TRadioButton;

OperatorPlus: TRadioButton;

OperatorDiv: TRadioButton;

OperatorMinus: TRadioButton;

OperatorMul: TRadioButton;

OperatorGt: TRadioButton;

OperatorLt: TRadioButton;

OperatorEq: TRadioButton;

OperatorNe: TRadioButton;

OperatorGe: TRadioButton;

OperatorLe: TRadioButton;

Funct1: TRadioButton;

Funct2: TRadioButton;

Funct3: TRadioButton;

Funct4: TRadioButton;

Funct5: TRadioButton;

ResDivider: TBevel;

Bevel1: TBevel;

Bevel2: TBevel;

Bevel3: TBevel;

Bevel4: TBevel;

Bevel5: TBevel;

Bevel6: TBevel;

Bevel7: TBevel;

Bevel8: TBevel;

Bevel26: TBevel;

ResNum: TLabel;

ResDenom: TLabel;

CmpRes1: TLabel;

OperatorSymbol: TLabel;

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;

Label18: TLabel;

Label19: TLabel;

Label20: TLabel;

Label21: TLabel;

BitBtn1: TBitBtn;

**procedure** FormShow(Sender: TObject);

**procedure** FormCreate(Sender: TObject);

**procedure** FormDestroy(Sender: TObject);

**procedure** Button1Click(Sender: TObject);

**procedure** OperatorSelectClick(Sender: TObject);

**procedure** SimpleBtnClick(Sender: TObject);

**procedure** FormulsBtnClick(Sender: TObject);

**procedure** FunctSelectClick(Sender: TObject);

**procedure** N2Click(Sender: TObject);

**procedure** N3Click(Sender: TObject);

**function** CheckStringOper(InStr : **string**; **var** LogStr:**string**) : **boolean**;

**function** IntFromString(Line: **string**; **var** PosFrom: **integer**): **integer**;

**function** SetFromString(Line: **string**; **var** PosFrom: **integer**; **var** Res1 : TRationalFraction): **boolean**;

**procedure** N5Click(Sender: TObject);

**private**

X, X1, Res : TRationalFraction;

Operation : **short**;

**end**;

**var**

Form1: TForm1;

Number : **integer**;

**implementation**

*{$R \*.dfm}*

**procedure** TForm1.FormShow(Sender: TObject);

**begin**

EnterNum.Text := '1';

EnterDenom.Text := '1';

**end**;

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

**var**

a, b : **integer**;

CompResult : **boolean**;

N, Res1, Res2 : TRationalFraction;

**begin**

CompResult := **false**;

*// вводим данные с обработкой ошибок*

**try**

a := StrToInt(EnterNum.Text);

b := StrToInt(EnterDenom.Text);

X.SetValue(a, b);

**if**(Operand2.Visible) **then** **begin**

a := StrToInt(EnterNum2.Text);

b := StrToInt(EnterDenom2.Text);

X1.SetValue(a, b);

**end** **else** X1.SetValue(1,1);

**except**

**on** Exception : EConvertError do

**begin**

StatusBar1.Panels[0].Text := 'Ошибка:';

StatusBar1.Panels[1].Text := 'при вводе числа';

StatusBar1.Panels[2].Text := Exception.Message;

**exit**;

**end**

**end**;

**if**(X.ErrorFlag) **then**

**begin**

StatusBar1.Panels[0].Text := 'Ошибка';

StatusBar1.Panels[1].Text := 'в дроби X';

StatusBar1.Panels[2].Text := X.ErrorStr;

ResNum.Caption := 'Ошибка';

ResDenom.Caption := 'Ошибка';

CmpRes1.Caption := 'Ошибка';

**exit**;

**end**;

**if**(X1.ErrorFlag) **then**

**begin**

StatusBar1.Panels[0].Text := 'Ошибка';

StatusBar1.Panels[1].Text := 'в дроби X1';

StatusBar1.Panels[2].Text := X1.ErrorStr;

ResNum.Caption := 'Ошибка';

ResDenom.Caption := 'Ошибка';

CmpRes1.Caption := 'Ошибка';

**exit**;

**end**;

**case** Operation of

0 : **begin** *// +*

Res.SetValue(X);

Res.Add(X1);

**end**;

1 : **begin** *// -*

Res.SetValue(X);

Res.Sub(X1);

**end**;

2 : **begin** *// \**

Res.SetValue(X);

Res.Multiply(X1);

**end**;

3 : **begin** *// /*

Res.SetValue(X);

Res.Divide(X1);

**end**;

4 : CompResult := X.Eq(X1); *// =*

5 : CompResult := X.Gt(X1); *// >*

6 : CompResult := X.Lt(X1); *// <*

7 : CompResult := X.Ne(X1); *// <>*

8 : CompResult := X.Ge(X1); *// >=*

9 : CompResult := X.Le(X1); *// <=*

10 : **begin** *// F(x) = 2x+1/x*

Res.SetValue(X);

N := TRationalFraction.Create(2,1);

Res.Multiply(N);

Res1 := TRationalFraction.Create(1,1);

Res1.Divide(X);

Res.Add(Res1);

N.Free;

Res1.Free;

**end**;

11 : **begin** *// F(x) = (x-1)/(x-2)*

Res1 := TRationalFraction.Create(X);

N := TRationalFraction.Create(2,1);

Res1.Sub(N);

Res.SetValue(X);

N.SetValue(1,1);

Res.Sub(N);

Res.Divide(Res1);

N.Free;

Res1.Free;

**end**;

12 : **begin** *// F(x) = x\*x - 1/x*

Res1 := TRationalFraction.Create(1,1);

Res.SetValue(X);

Res.Multiply(X);

Res1.Divide(X);

Res.Sub(Res1);

Res1.Free;

**end**;

13 : **begin** *// F(x) = x/3 - 1/(1+x)*

Res1 := TRationalFraction.Create(X);

Res2 := TRationalFraction.Create(1,1);

Res.SetValue(X);

N := TRationalFraction.Create(3,1);

Res.Divide(N);

N.SetValue(1,1);

Res1.Add(N);

Res2.Divide(Res1);

Res.Sub(Res2);

Res1.Free;

Res2.Free;

N.Free;

**end**;

14 : **begin**

Res1 := TRationalFraction.Create(X);

Res.SetValue(3,1);

Res.Divide(X);

N := TRationalFraction.Create(11,1);

Res1.Divide(N);

N.SetValue(5,1);

Res1.Multiply(N);

Res.Add(Res1);

Res1.Free;

N.Free;

**end**;

**end**;

**if**(Operation <= 3) **or** (Operation >= 10 ) **then**  *// операторы вычисления*

**begin**

ResNum.Caption := Res.StrNumerator;

ResDenom.Caption := Res.StrDenominator;

**if**(Res.ErrorFlag) **then**

**begin**

StatusBar1.Panels[0].Text := *'Ошибка'*;

StatusBar1.Panels[1].Text := *'Результат операции:* ';

StatusBar1.Panels[2].Text := Res.ErrorStr;

**exit**;

**end** **else**

**begin**

StatusBar1.Panels[0].Text := *'Успешно'*;

StatusBar1.Panels[1].Text := *'Результат операции: '*;

StatusBar1.Panels[2].Text := Res.ErrorStr;

**end**;

**end** **else**

**if**(Operation < 10) **then**

**begin**

**if**(CompResult) **then**

CmpRes1.Caption := *'Истина'*

**else**

CmpRes1.Caption := '*Ложь'*;

**end**;

**end**;

**procedure** TForm1.FormCreate(Sender: TObject);

**begin**

X := TRationalFraction.Create(1,1);

X1 := TRationalFraction.Create(1,1);

Res := TRationalFraction.Create(1,1);

Operation := 0;

**end**;

**procedure** TForm1.FormDestroy(Sender: TObject);

**begin**

Res.Free;

X.Free;

X1.Free;

**end**;

*// выбираем операцию по положению галочек*

**procedure** TForm1.OperatorSelectClick(Sender: TObject);

**var**

isCompare : boolean;

**begin**

isCompare := **false**;

**if**(SimpleBtn.Checked) **then**

**begin**

**if**( OperatorPlus.Checked) **then**

**begin**

OperatorSymbol.Caption := '+';

isCompare := **false**;

Operation := 0;

**end**;

**if**( OperatorMinus.Checked) **then**

**begin**

OperatorSymbol.Caption := '-';

isCompare := **false**;

Operation := 1;

**end**;

**if**( OperatorMul.Checked) **then**

**begin**

OperatorSymbol.Caption := '\*';

isCompare := **false**;

Operation := 2;

**end**;

**if**( OperatorDiv.Checked) **then**

**begin**

OperatorSymbol.Caption := '/';

isCompare := **false**;

Operation := 3;

**end**;

**if**( OperatorEq.Checked) **then**

**begin**

OperatorSymbol.Caption := '=';

isCompare := **true**;

Operation := 4;

**end**;

**if**( OperatorGt.Checked) **then**

**begin**

OperatorSymbol.Caption := '>';

isCompare := **true**;

Operation := 5;

**end**;

**if**( OperatorLt.Checked) **then**

**begin**

OperatorSymbol.Caption := '<';

isCompare := **true**;

Operation := 6;

**end**;

**if**( OperatorNe.Checked) **then**

**begin**

OperatorSymbol.Caption := '<>';

isCompare := **true**;

Operation := 7;

**end**;

**if**( OperatorGe.Checked) **then**

**begin**

OperatorSymbol.Caption := '>=';

isCompare := **true**;

Operation := 8;

**end**;

**if**( OperatorLe.Checked) **then**

**begin**

OperatorSymbol.Caption := '<=';

isCompare := **true**;

Operation := 9;

**end**;

**if**(isCompare) **then**

**begin**

CmpRes1.Visible := **true**;

ResDenom.Visible := **false**;

ResNum.Visible := **false**;

ResDivider.Visible := **false**;

**end** **else**

**begin**

CmpRes1.Visible := **false**;

ResDenom.Visible := **true**;

ResNum.Visible := **true**;

ResDivider.Visible := **true**;

**end**;

**end**;

Button1Click(Sender);

**end**;

**procedure** TForm1.SimpleBtnClick(Sender: TObject);

**begin**

Operand2.Visible := **true**;

Operators.Visible := **true**;

Functions.Visible := **false**;

OperatorSymbol.Visible := **true**;

Operation := 0; *// возвращаем на операцию '+'*

OperatorPlus.Checked := **true**;

F1.Visible := **false**;

F2.Visible := **false**;

F3.Visible := **false**;

F4.Visible := **false**;

F5.Visible := **false**;

OperatorSelectClick(Sender);

**end**;

**procedure** TForm1.FormulsBtnClick(Sender: TObject);

**begin**

Operand2.Visible := **false**;

Operators.Visible := **false**;

Functions.Visible := **true**;

OperatorSymbol.Visible := **false**;

Operation := 10;

Funct1.Checked := **true**;

ResNum.Visible := **true**;

Resdenom.Visible := **true**;

ResDivider.Visible := **true**;

CmpRes1.Visible := **false**;

OperatorSelectClick(Sender);

FunctSelectClick(Sender);

**end**;

**procedure** TForm1.FunctSelectClick(Sender: TObject);

**begin**

**if**(Funct1.Checked) **then**

**begin**

Operation:=10;

F1.Visible := **true**;

**end** **else** F1.Visible := **false**;

**if**(Funct2.Checked) **then**

**begin**

Operation:=11;

F2.Visible := **true**;

**end** **else** F2.Visible := **false**;

**if**(Funct3.Checked) **then**

**begin**

Operation:=12;

F3.Visible := **true**;

**end** **else** F3.Visible := **false**;

**if**(Funct4.Checked) **then**

**begin**

Operation:=13;

F4.Visible := **true**;

**end** **else** F4.Visible := **false**;

**if**(Funct5.Checked) **then**

**begin**

Operation:=14;

F5.Visible := **true**;

**end** **else** F5.Visible := **false**;

Button1Click(Sender);

**end**;

**procedure** TForm1.N2Click(Sender: TObject);

**begin**

Form2.ShowModal;

**end**;

*// функция читает из выбранного файла данные, обрабатывает их и выводит на форму Form3*

**procedure** TForm1.N3Click(Sender: TObject);

**var**

s, log : **string**;

f: Textfile;

**begin**

**if**(Form3.OpenDialog1.Execute) **then**

**begin**

Form3.Memo1.Clear;

Form3.Show;

AssignFile(f, Form3.OpenDialog1.FileName ); {Assigns the Filename}

Reset(f); {Opens the file for reading}

Form3.Memo1.Lines.Add('Открыт файл : ' + Form3.OpenDialog1.FileName);

Form3.Memo1.Lines.Add('Результат обработки данных: ');

**repeat**

Readln(f, s);

CheckStringOper(s, log);

**if**(log <> '') **then** Form3.Memo1.Lines.Add(log);

**until**(Eof(f));

CloseFile(f);

**end**;

**end**;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// функция возвращает результат проверки утверждения из строки InStr*

*// символы после '#' игнорируются*

*// формат строки:*

*// 1. проверка арифметических операций:*

*// X1/Y1 S X2/Y2 = X3/Y3 или 'e' (при ошибке),*

*// где S может быть '+', '-', '\*', '/'*

*// 2. проверка операций сравнения:*

*// X1/Y1 S X2/Y2 = R , где R может быть 't', 'f',*

*// где S : '>', '<', '=', '<>', '>=', '<='*

*// X1,Y1, X2, Y2 операнды*

*// X3, Y3, R ожидаемый результат операции*

**function** TForm1.CheckStringOper(InStr : **string**; **var** LogStr :**string**) : **boolean**;

**var**

T1, T2, T3, Res1 : TRationalFraction;

i, o, k : **integer**;

b, eq, br : **boolean**;

er : **boolean**; *// признаки ошибок*

Sym, Comment : **string**;

**begin**

// создаем обьекты

T1 := TRationalFraction.Create;

T2 := TRationalFraction.Create;

T3 := TRationalFraction.Create;

Res1 := TRationalFraction.Create;

*// инициализируем переменные*

b:=**false**;

br := **false**;

er := **false**;

eq := **false**;

o := 0;

k := 1;

i := 0;

*// разбираем полученную строку*

while (i <= Length(InStr)) **and** (k < 7) do

**begin**

i := i + 1;

**if**(InStr[i] = ' ') **then** continue **else**

**if**(InStr[i] = *'#'*) **then** **break**; *// дальше идет комментарий*

**case** k of

1: **begin** *// первый аргумент*

SetFromString(InStr, i, T1);

k := 2;

**end**;

2: **begin** *// символ операции*

**case** InStr[i] of

*'+'*: o := 1;

*'-'*: o := 2;

*'\*'*: o := 3;

*'/'*: o := 4;

*'<'*: **begin**

**if**(InStr[i+1]=' ') **then** o := 5 **else** *// <*

**begin**

**if**(InStr[i+1]='=') **then** o := 9 **else** *// <=*

**if**(InStr[i+1]='>') **then** o := 7; *// <>*

i:= i + 1;

**end**;

eq := **true**;

**end**;

'>': **begin**

**if**(InStr[i+1]=' ') **then** o := 6 **else** *// >*

**begin**

**if**(InStr[i+1]='=') **then** o := 10; *// >=*

i:= i + 1;

**end**;

eq := **true**;

**end**;

'=': **begin** o := 8; eq:= **true** **end**;

**else** **break**; *// неизвестный символ*

**end**;

k := 3;

**end**;

3: **begin** *// второй аргумент*

SetFromString(InStr, i, T2);

k := 4;

**end**;

4: **begin** *// символ равенства*

**if** InStr[i] <> '=' **then** **break**; *// если не равно - ошибка*

**if**(**not** eq) **then** k := 5 **else** k := 6;

**end**;

5: **begin** *// ожидаемый результат - дробь*

**if** InStr[i] = 'e' **then** **begin** er := **true**; **break**; **end**; *// ожидаемый результат - ошибка*

SetFromString(InStr, i , T3);

k:=7; *// ввод данных завершен*

**end**;

6: **begin** *// ожидаемый результат символ*

**if** InStr[i] = 't' **then** br := **true** **else**

**if** InStr[i] = 'f' **then** br := **false**;

k:=7; *// ввод данных завершен*

**end**;

**else** **break**;

**end**;

**end**;

*// вычисляем полученное уравнение*

Res1.SetValue(T1);

**case** o of

1: **begin** Res1.Add(T2); Sym:='+'; **end**;

2: **begin** Res1.Sub(T2); Sym:='-'; **end**;

3: **begin** Res1.Multiply(T2);Sym:='\*'; **end**;

4: **begin** Res1.Divide(T2); Sym:='/'; **end**;

5: **begin** b:=T1.Lt(T2); Sym:='<'; **end**;

6: **begin** b:=T1.Gt(T2); Sym:='>'; **end**;

7: **begin** b:=T1.Ne(T2); Sym:='<>'; **end**;

8: **begin** b:=T1.Eq(T2); Sym:='='; **end**;

9: **begin** b:=T1.Le(T2); Sym:='<='; **end**;

10: **begin** b:=T1.Ge(T2); Sym:='>='; **end**;

**else**

Result := **true**;

**exit**;

**end**;

**if**(**not** Result) **then**

**Begin** *// формируем строку с отчетом*

LogStr := T1.StrNumerator + '/' + T1.StrDenominator + ' ' + Sym + ' ' + T2.StrNumerator + '/' + T2.StrDenominator + ' = ';

**if**( **not** eq) **then** *// операции + - \* /*

**begin**

b := Res1.Eq(T3);

LogStr := LogStr + Res1.StrNumerator + '/' + Res1.StrDenominator + *' (Ожидаем : '*;

**if**(er) **then**

**begin**

LogStr := LogStr + *'Ошибка ) '*;

**if**(Res1.ErrorFlag) **then**

LogStr:=LogStr + *' - (Верно)* ' + Comment

**else**

LogStr:=LogStr + *' - (Ошибка) '* + Comment;

**end** **else**

**begin**

LogStr := LogStr + T3.StrNumerator + '/' + T3.StrDenominator + ')';

**if**(b) **then**

LogStr:=LogStr + *' - (Верно) '* + Comment

**else**

LogStr:=LogStr + *' - (Ошибка) '* + Comment;

**end**;

Result := b;

**end** **else**

**begin** *// операции сравнения*

**if**(b) **then**

LogStr := LogStr + ' Истина '

**else**

LogStr := LogStr + ' Ложь ';

LogStr := LogStr + ' (Ожидаем : ';

**if**(br) **then**

**begin**

LogStr := LogStr + *' Истина) '*;

**end** **else**

**begin**

LogStr := LogStr *+ ' Ложь)* ';

**end**;

**if**(b = br) **then**

LogStr:=LogStr + ' *- (Верно)* ' + Comment

**else**

LogStr:=LogStr + ' *- (Ошибка)* ' + Comment;

Result := (b = br);

**end**;

**End** **else** LogStr:= *'Ошибка разбора строки.'*;

*// освобождаем ресурсы*

T1.Free;

T2.Free;

T3.Free;

Res1.Free;

**end**;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// Функция считывает из строки Line последовательность типа рациональная дробь и*

*// задает значение Res1*

*// считывание начинается с позиции PosFrom*

*// в случае ошибок преобразования обьект типа TRationalFraction*

*// устанавливается в сосотояние ошибки*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**function** TForm1.SetFromString(Line: **string**; **var** PosFrom: **integer**; **var** Res1 : TRationalFraction): **boolean**;

**var**

l, m : **integer**;

c : char;

**begin**

Result := **false**;

l := 0;

m := 0;

**try**

l := IntFromString(Line,PosFrom);

**except**

**on** Exception : EConvertError do

Result := Result **or** **true**;

**end**;

c := Line[PosFrom];

PosFrom:= PosFrom + 1;

**try**

m := IntFromString(Line,PosFrom);

**except**

**on** Exception : EConvertError do

Result := Result **or** **true**;

**end**;

**if**(Result) **then** m := 0; *// заставляем обьект перейти в ошибочное состояние*

**if**(c = '/') **then**

**begin**

Res1.SetValue(l,m);

Result := Result **or** **false**;

**end** **else**

**begin**

Result := Result **or** **true**;

**end**;

**end**;

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*// Функция считывает из строки число, начиная с позиции PosFrom и заканчивая*

*// концом строки, первым пробелом или символом '/' после позиции PosFrom*

*// в случае ошибки, появляется прерывание EConvertError*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**function** TForm1.IntFromString(Line: **string**; **var** PosFrom: **integer**): **integer**;

**var**

i, k, l : **integer**;

b : **boolean**;

r : **string**;

**Begin**

b := **false**;

k := 0;

l := 0;

for i := PosFrom to Length(Line) **do**

**begin**

**if**((Line[i] <> ' ') **and** (Line[i] <> '/')) **then**

**begin**

**if**(b=**false**) **then**

**begin**

l := i;

b:=**true**; *// начинается число*

**end**;

k := k + 1;

**end** **else** **if**(b) **then** **break**;

**end**;

**if**(b) **then**

**begin** // конец числа

r := Copy(Line, l, k);

PosFrom := i;

Result := StrToInt(r);

**exit**;

**end**;

Result := 0; // число не найдено (ошибка)

**End**;

**procedure** TForm1.N5Click(Sender: TObject);

**begin**

Application.Terminate;

**end**;

**end**.

## Модуль LogForm

**unit** LogForm;

**interface**

**uses**

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, Menus, Grids, ComCtrls;

**type**

TForm3 = **class**(TForm)

Memo1: TMemo;

SaveDialog1: TSaveDialog;

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

OpenDialog1: TOpenDialog;

**procedure** N2Click(Sender: TObject);

**procedure** N4Click(Sender: TObject);

**end**;

**var**

Form3: TForm3;

**implementation**

*{$R \*.dfm}*

**procedure** TForm3.N2Click(Sender: TObject);

**begin**

**if**(SaveDialog1.Execute) **then**

**begin**

Memo1.Lines.SaveToFile(SaveDialog1.FileName);

**end**;

**end**;

**procedure** TForm3.N4Click(Sender: TObject);

**begin**

Visible := **false**;

**end**;

**end**.

## Модуль AboutForm

**unit** AboutForm;

**interface**

**uses**

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls, Buttons;

**type**

TForm2 = **class**(TForm)

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;

OK: TBitBtn;

**end**;

**var**

Form2: TForm2;

**implementation**

*{$R \*.dfm}*

**end**.

## Файл тестовых данных

*# формат строки:*

*# математические операции '+', '-', '\*', '/':*

*# X1/Y1 $ X2/Y2 = X3/X3 (результат операции) или 'e' (если ожидаем ошибку в операции)*

*# логические операции '=', '<>', '>', '<', '>=', '<=' :*

*# X1/Y1 $ X2/Y2 = t (истина), f (ложь), или е (если ожидаем ошибку в операции)*

*# символы после '#' игнорируются*

12t/6 + ert/2 = e

12/t6 + 7/2 = e

65/0 - 4/7 = e

12/3 + 14/5 = 34/5 *# проверка сложения*

12/3 - 14/-5 = 34/5 *# проверка вычитания*

24/7 \* 7/14 = 12/7 *# проверка умножения*

24/7 \* -7/14 = -12/7 *#*

1/2 / 2/3 = 3/4

1/2 - 1/2 = 0/1

1/2 > 1/3 = t

1/2 = 1/3 = f

1/2 < 1/3 = f

1/2 <> 1/3 = t

1/2 >= 1/3 = t

1/2 <= 1/3 = f