МИНОБРНАУКИ РФ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ИНСТИТУТ РАДИОТЕХНИКИ,

ЭЛЕКТРОНИКИ И АВТОМАТИКИ (ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)

Факультет ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА:

«МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ»

**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.**

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 МОВС. – стр.46, рис 26, табл. 1.

*Объектно-ориентированное программирование: Объект «Рациональная дробь».* В курсовой работе рассмотрена концепция объектно-ориентированного программирования. Рассмотрены современные языки, в основу которых входит ООП. Более подробно рассмотрен язык Delphi. Описана реализация класса «Рациональная дробь» с использованием методов ООП языка Delphi. Создана готовая программа, реализующая взаимодействие с объектом и написаны и проведены тесты корректности вычислений.

©Мошкалев Д.В.

2012г.

Оглавление

[Техническое задание 4](#_Toc326153064)

[Введение 5](#_Toc326153065)

[1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 8](#_Toc326153066)

[1.1. Основные характеристики языка 8](#_Toc326153067)

[1.2. Рациональные дроби и их свойства 10](#_Toc326153068)

[2. ПРОЕКТНАЯ ЧАСТЬ 11](#_Toc326153069)

[2.1. Постановка задачи 11](#_Toc326153070)

[2.2. Описание входных данных 12](#_Toc326153071)

[2.3. Структура класса 12](#_Toc326153072)

[2.4. Разработка интерфейса 15](#_Toc326153073)

[3. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ 21](#_Toc326153074)

[3.1. Тестирование программы 21](#_Toc326153075)

[ЗАКЛЮЧЕНИЕ 26](#_Toc326153076)

[СПИСОК ЛИТЕРАТУРЫ 26](#_Toc326153077)

[ПРИЛОЖЕНИЕ 1. Руководство оператора 27](#_Toc326153078)

[1. Режим бинарных операций. 27](#_Toc326153079)

[2. Режим вычисления формул. 28](#_Toc326153080)

[3. Выполнение пакетного тестирования. 29](#_Toc326153081)

[4. Получение информации о программе. 30](#_Toc326153082)

[ПРИЛОЖЕНИЕ 2. Листинг программы 31](#_Toc326153083)

[1. Файл программы 31](#_Toc326153084)

[2. Модуль RatFracClass 31](#_Toc326153085)

[3. Модуль RatFracForm 35](#_Toc326153086)

[4. Модуль LogForm 43](#_Toc326153087)

[5. Модуль AboutForm 44](#_Toc326153088)

[6. Файл тестовых данных 45](#_Toc326153089)

# 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

Объектно-ориентированное программирование является наиболее прогрессивной и бурно развивающейся технологией разработки программ.

ООП возникло в результате развития идеологии процедурного программирования, где данные и подпрограммы (процедуры, функции) их обработки формально не связаны. Для дальнейшего развития объектно-ориентированного программирования часто большое значение имеют понятия события (так называемое событийно-ориентированное программирование) и компонента (компонентное программирование, КОП).

Формирование компонентное программирование на основе объектно-ориентированного программирования началось так же как начиналось формирование модульного от процедурного программирования: процедуры сформировались в модули — независимые части кода до уровня сборки программы, так объекты сформировались в компоненты — независимые части кода до уровня выполнения программы. Взаимодействие объектов происходит посредством сообщений. Результатом дальнейшего развития ООП, по-видимому, будет агентно-ориентированое программирование, где *агенты* — независимые части кода на уровне выполнения. Взаимодействие агентов происходит посредством изменения *среды*, в которой они находятся.

Языковые конструкции, конструктивно не относящиеся непосредственно к объектам, но сопутствующие им для их безопасной (исключительные ситуации, проверки) и эффективной работы, инкапсулируются от них в аспекты (в аспектно-ориентированном программировании). Субъектно-ориентированное программирование расширяет понятие объект посредством обеспечения более унифицированного и независимого взаимодействия объектов. Может являться переходной стадией между ООП и агентным программирование в части самостоятельного их взаимодействия.

Первым языком программирования, в котором были предложены принципы объектной ориентированности, была Симула. В момент своего появления (в 1967 году), этот язык программирования предложил поистине революционные идеи: объекты, классы, виртуальные методы и др., однако это всё не было воспринято современниками как нечто грандиозное. Тем не менее, большинство концепций были развиты Аланом Кэйем и Дэном Ингаллсом в языке Smalltalk. Именно он стал первым широко распространённым объектно-ориентированным языком программирования.

В настоящее время количество прикладных языков программирования, реализующих объектно-ориентированную парадигму, является наибольшим по отношению к другим парадигмам. В области системного программирования до сих пор применяется парадигма процедурного программирования, и общепринятым языком программирования является язык C. Хотя при взаимодействии системного и прикладного уровней операционных систем заметное влияние стали оказывать языки объектно-ориентированного программирования. Например, одной из наиболее распространенных библиотек мультиплатформенного программирования является объектно-ориентированная библиотека Qt, написанная на языке C++. В области прикладного программирования для Windows часто используются язык Delphi. Его главным достоинством является большая скорость разработки приложений для Windows за счет большой библиотеки готовых модулей.

ООП ориентировано на разработку крупных программных комплексов, разрабатываемых командой программистов (возможно, достаточно большой). Проектирование системы в целом, создание отдельных компонент и их объединение в конечный продукт при этом часто выполняется разными людьми, и нет ни одного специалиста, который знал бы о проекте всё.

Объектно-ориентированное проектирование состоит в описании структуры и поведения проектируемой системы, то есть, фактически, в ответе на два основных вопроса:

* Из каких частей состоит система.
* В чём состоит ответственность каждой из частей.

Выделение частей производится таким образом, чтобы каждая имела минимальный по объёму и точно определённый набор выполняемых функций (обязанностей), и при этом взаимодействовала с другими частями как можно меньше.

Дальнейшее уточнение приводит к выделению более мелких фрагментов описания. По мере детализации описания и определения ответственности выявляются данные, которые необходимо хранить, наличие близких по поведению агентов, которые становятся кандидатами на реализацию в виде классов с общими предками. После выделения компонентов и определения интерфейсов между ними реализация каждого компонента может проводиться практически независимо от остальных (разумеется, при соблюдении соответствующей технологической дисциплины).

Большое значение имеет правильное построение иерархии классов. Одна из известных проблем больших систем, построенных по ООП-технологии — так называемая *проблема хрупкости базового класса*. Она состоит в том, что на поздних этапах разработки, когда иерархия классов построена и на её основе разработано большое количество кода, оказывается трудно или даже невозможно внести какие-либо изменения в код базовых классов иерархии (от которых порождены все или многие работающие в системе классы). Даже если вносимые изменения не затронут интерфейс базового класса, изменение его поведения может непредсказуемым образом отразиться на классах-потомках. В случае крупной системы разработчик базового класса не просто не в состоянии предугадать последствия изменений, он даже не знает о том, как именно базовый класс используется и от каких особенностей его поведения зависит корректность работы классов-потомков.

Основные понятия ООП:

* Абстрагирование — это способ выделить набор значимых характеристик объекта, исключая из рассмотрения незначимые. Соответственно, абстракция — это набор всех таких характеристик.
* Инкапсуляция — это свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе и скрыть детали реализации от пользователя.
* Наследование — это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс — потомком, наследником или производным классом.
* Полиморфизм — это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.
* Класс является описываемой на языке терминологии (пространства имён) исходного кода моделью ещё не существующей сущности (объекта). Фактически он описывает устройство объекта, являясь своего рода чертежом. Говорят, что объект — это **экземпляр** класса. При этом в некоторых исполняющих системах класс также может представляться некоторым объектом при выполнении программы посредством динамической идентификации типа данных. Обычно классы разрабатывают таким образом, чтобы их объекты соответствовали объектам предметной области.
* Сущность в адресном пространстве вычислительной системы, появляющаяся при создании экземпляра класса или копирования прототипа (например, после запуска результатов компиляции и связывания исходного кода на выполнение).
* Прототип — это объект-образец, по образу и подобию которого создаются другие объекты. Объекты-копии могут сохранять связь с родительским объектом, автоматически наследуя изменения в прототипе; эта особенность определяется в рамках конкретного языка.

Многие современные языки специально созданы для облегчения объектно-ориентированного программирования. Однако следует отметить, что можно применять техники ООП и для не-объектно-ориентированного языка и наоборот, применение объектно-ориентированного языка вовсе не означает, что код автоматически становится объектно-ориентированным.

Современный объектно-ориентированный язык предлагает, как правило, следующий обязательный набор синтаксических средств:

* Объявление классов с полями (данными — членами класса) и методами (функциями — членами класса).
* Механизм расширения класса (наследования) — порождение нового класса от существующего с автоматическим включением всех особенностей реализации класса-предка в состав класса-потомка. Большинство ООП-языков поддерживают только единичное наследование.
* Полиморфные переменные и параметры функций (методов), позволяющие присваивать одной и той же переменной экземпляры различных классов.
* Полиморфное поведение экземпляров классов за счёт использования виртуальных методов. В некоторых ООП-языках все методы классов являются виртуальными.
* Конструкторы, деструкторы, финализаторы.
* Свойства (аксессоры).
* Индексаторы.
* Интерфейсы (например, в Java используются также как альтернатива множественному наследованию — любой класс может реализовать сколько угодно интерфейсов).
* Переопределение операторов для классов.
* Средства защиты внутренней структуры классов от несанкционированного использования извне. Обычно это модификаторы доступа к полям и методам, типа public, private, обычно также protected, иногда некоторые другие.

Часть языков целиком построена вокруг объектных средств — в них любые данные являются объектами, любой код — методом какого-либо класса, и невозможно написать программу, в которой не использовались бы объекты. Примеры подобных языков — Smalltalk, Python, Java, C#, Ruby, AS3. Другие языки (иногда используется термин «гибридные») включают ООП-подсистему в исходно процедурный язык. В них существует возможность программировать, не обращаясь к объектным средствам. Классические примеры — C++, Delphi и Perl.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## Основные характеристики языка

**Delphi**— императивный, структурированный, объектно-ориентированный язык программирования, диалект Object Pascal. Начиная со среды разработки Delphi 7.0, в официальных документах Borland стала использовать название Delphi для обозначения языка Object Pascal. Начиная с 2007 года уже язык Delphi (производный от Object Pascal) начал жить своей самостоятельной жизнью и претерпевал различные изменения, связанные с современными тенденциями (например, с развитием платформы .NET) развития языков программирования: В языке Object Pascal классы — это специальные типы данных, используемые для описания объектов. Соответственно объект, имеющий тип какого-либо класса, является экземпляром (instance) этого класса или переменной этого типа.

Delphi представляет следующие новые свойства и усовершенствования:

* **Новые расширения языка**. В Delphi в язык Object Pascal включены динамические массивы, методы обработки переполнения, установка значения параметров по умолчанию, и многое другое;
* **Менеджер Проекта.** Новый менеджер проекта позволяет Вам объединять проекты, которые работают вместе в одну проектную группу. Это позволяет организовать как работу взаимозависимых проектов, таких как однозадачные и многозадачные приложения или dll, так и совместную работу исполняемых программ;
* **Новый проводник**. Новый проводник содержит выполняемые классы, навигацию по модулям, и браузер кода. Проводник кода делает создание классов проще. Также проводник позволяет быстро перемещаться через файлы модуля, а так же между интерфейсом и реализацией;
* **Закрепляемые окна инструментов.** IDE (Интегрированная Среда разработки) содержит более перенастраиваемую конфигурацию окон инструментов, которые можно закреплять с редактором кода;
* **Улучшенная отладка.** Интегрированный отладчик имеет много новых свойств, включая удаленную и многопроцессорную отладку, просмотр кода центрального процессора, инспекторов, усовершенствованные точки прерывания, отладчик специфических подменю и закрепленных окон;
* **Усовершенствования Activex;**
* **Усовершенствования VCL.** Иерархия объектов Delphi была расширена, чтобы включить новый компонент для Nt Service приложений. Кроме того, новый компонент выполняемого списка (на Стандартной странице палитры), позволяет централизовать управление меню и команд от кнопок. Управление VCL расширено, чтобы поддерживать drag-and-drop перетаскивания, обеспечивать дополнительный контроль над размещением окна, и многое другое.

Delphi – это комбинация нескольких важнейших технологий:

* высокопроизводительный компилятор в машинный код;
* объектно-ориентированная модель компонент;
* визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов;
* масштабируемые средства для построения баз данных.

Компилятор, встроенный в Delphi, обеспечивает высокую производительность, необходимую для построения приложений в архитектуре «клиент-сервер». Он предлагает легкость разработки и быстрое время проверки готового программного блока, характерного для языков четвертого поколения. Кроме того, Delphi обеспечивает быструю разработку без необходимости писать вставки на Си или ручного написания кода (хотя это возможно).

В процессе построения приложения разработчик выбирает из палитры компонент готовые компоненты как художник, делающий крупные мазки кистью. Еще до компиляции он видит результаты своей работы – после подключения к источнику данных их можно видеть отображенными на форме, можно перемещаться по данным, представлять их в том или ином виде. В этом смысле проектирование в Delphi мало чем отличается от проектирования в интерпретирующей среде, однако после выполнения компиляции мы получаем код, который исполняется в 10-20 раз быстрее, чем то же самое, сделанное при помощи интерпретатора. Кроме того, в Delphi компиляция производится непосредственно в машинный код, в то время как существуют компиляторы, превращающие программу в так называемый p-код, который затем интерпретируется виртуальной p-машиной. Это не может не сказаться на фактическом быстродействии готового приложения.

В стандартную поставку Delphi входят основные объекты, которые образуют удачно подобранную иерархию базовых классов. Но если возникнет необходимость в решении какой-то специфической проблемы на Delphi, то лучше просмотреть список свободно распространяемых или коммерческих компонент, разработанных третьими фирмами, количество этих компонент в настоящее время составляет несколько тысяч. Событийная модель в Windows всегда была сложна для понимания и отладки. Но именно разработка интерфейса в Delphi является самой простой задачей для программиста.

Объекты БД в Delphi основаны на SQL и включают в себя полную мощь Borland Database Engine. В состав Delphi также включен Borland SQL LINK, поэтому доступ к СУБД Oracle, Sybase, Informix и Interbase происходит с высокой эффективностью. Кроме того, Delphi включает в себя локальный сервер Interbase для того, чтобы можно было разработать расширяемые на любые внешние sql-сервера приложения в офлайновом режиме. Разработчик в среде Delphi, проектирующий информационную систему для локальной машины (к примеру, небольшую систему учета медицинских карточек для одного компьютера), может использовать для хранения информации файлы формата .dbf (как в dbase или clipper) или .db (paradox). Если же он будет использовать локальный interbase for windows (это локальный SQL-сервер, входящий в поставку), то его приложение безо всяких изменений будет работать и в составе большой системы с архитектурой клиент-сервер.

## Рациональные дроби и их свойства

**Рациональная дробь** - это число, представленное в виде дроби, например

где ***a*** - числитель, ***b*** - знаменатель. ***a*** и ***b*** могут представлять собой целые числа, а также переменные.

Над рациональными дробями возможны следующие основные действия:

* Сложение
* Вычитание
* Умножение
* Деление
* Сокращение

Если ***,*** тогда дробь можно сократить, разделив числитель и знаменатель на .

# ПРОЕКТНАЯ ЧАСТЬ

## Постановка задачи

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.

## Описание входных данных

Входные данные могут быть представлены как в виде текстовых файлов, содержащих тестовые примеры, так и введены вручную в формы диалогового окна.

Из тестового файла данные считываются построчно. Строка должна содержать две дроби – операнды, символ операции между ними (+,-,\*,/,>,<,=,<>,>=,<=), символ «=» и результат операции. В качестве результата операции могут быть:

* символ «е», означающий ошибку,
* символы «t» или «f», означающие «**true**» и «false» - результат операции сравнения
* дробь – результат арифметической операции (+,-,\*,/)

Дробь должна быть записана в виде .

Например: -1/2 + 1/3 = -1/6.

## Структура класса

Схема класса изображена на ***Рисунок 1*** .



Рисунок . Структурная схема класса

Класс содержит основные публичные свойства:

* Numerator – числитель, значение целого типа. Свойство доступно только для чтения.
* Denominator – знаменатель, значение целого типа. Свойство доступно только для чтения.
* ErrorFlag – признак ошибки (деление на 0), значение логического типа. Свойство доступно только для чтения.

Конструкторы класса - перегружаемые функции:

* Create() - без параметров. Устанавливает значение числителя равное 0, знаменателя - 1.
* Create(A,B : integer) – с параметрами – целыми числами. A – числитель, B –знаменатель.
* Create(T : TRationalFraction) – копирующий конструктор. Параметр - экземпляр класса.

**Методы класса:**

Установка значений дроби:

* SetValue(A,B : integer) – Устанавливается значение дроби из двух целых чисел.

Здесь A – числитель, B –знаменатель.

* SetValue(T : TRationalFraction) – Копирование значений дроби из другого экземпляра класса.

Математические операции. Результаты операций модифицируют значение текущего экземпляра класса:

* Add(A,B : integer) – операция сложения. Дробь представлена целыми числами.
* Add(T : TRationalFraction) – Прибавляет к значению дроби значение дроби другого экземпляра класса.
* Sub(A,B : integer) – операция вычитания. Дробь представлена целыми числами.
* Sub(T : TRationalFraction) – Вычитает из значения дроби значение дроби другого экземпляра класса.
* Multiply(A,B : integer) – операция умножения. Дробь представлена целыми числами.
* Multiply(T : TRationalFraction) – Умножает дробь на значение дроби другого экземпляра класса.
* Divide(A,B : integer) – операция деления. Дробь представлена целыми числами.
* Divide(T : TRationalFraction) – Делит дробь на значение дроби другого экземпляра класса.

Логические операции. Логические операции возвращают значение логического типа. Текущий объект располагается с левой стороны от знака операции, операнд - с правой:

* Eq(T : TRationalFraction) – Возвращает результат логической операции «равно»
* Ne(T : TRationalFraction) – Возвращает результат логической операции «не равно»
* Gt(T : TRationalFraction) – Возвращает результат логической операции «больше»
* Ge(T : TRationalFraction) – Возвращает результат логической операции «больше или равно»
* Lt(T : TRationalFraction) – Возвращает результат логической операции «меньше»
* Le(T : TRationalFraction) – Возвращает результат логической операции «меньше или равно»

На ***Рисунок 2*** изображена схема состояний объекта типа TRationalFraction.

После создания, объект переходит в состояние ожидания операции. При выполнении математических операций и операций присваивания объект выполняет сокращение дроби, если необходимо и проверяет успешность выполнения операции. В случае отсутствия ошибок, переходит в состояние ожидания операции.

В случае обнаружения ошибки объект переходит в ошибочное состояние, которое передаётся всем объектам, которые получают его значение после перехода в состояние ошибки. Объект, находящийся в ошибочном состоянии можно только удалить операцией Free() или инициализировать новым значением.



Рисунок . Схема состояний объекта

При выполнении логических операций состояние объектов не меняется. Логические операции вызываются с помощью соответствующих методов класса. После завершения операции, объект возвращается в состояние ожидания.

В состав класса входят несколько свойств, предназначенных для отображения состояний объекта в текстовых полях формы:

* StrNumerator – возвращает значение числителя в текстовом виде.
* StrDenomerator – возвращает значение знаменателя в текстовом виде.
* ErrorStr– возвращает сообщение об ошибке в текстовом виде.

Этим свойствам соответствуют приватные методы класса:

* GetNumStr – преобразование числителя в текстовый вид
* GetDenomStr – преобразование знаменателя в текстовый вид
* GetErrorStr – формирование сообщения об ошибке

## 

## Разработка интерфейса

Интерфейс программы представлен в виде трех форм: Главной формы (Рисунок 3), Формы О программе (Рисунок 4) и формы результатов тестирования (Рисунок 5).

Главная форма предназначена для выполнения ручных вычислений.

Форма состоит из следующих управляющих элементов:

* Переключатель режимов работы. Имеет два положения: Бинарные операции и Вычисление формул.
* Переключатель Операций. Количество положений зависит от режима работы: в режиме бинарных операций 10 положений, в режиме вычисления формул 5 положений.
* Окно ввода значений операнда X – это окно не зависит от режима работы
* Окно ввода значений операнда X1 – это окно доступно только в режиме бинарных операций.
* Окно результата – вид этого окна зависит от типа операции: в режиме вычисления формул и при вычислении математических операций – это дробь, при вычислении логических операций – это слова «Истина» или «Ложь»
* Знак операции – зависит от типа операции и доступен только в режиме бинарных операций.
* Окно отображения формул – доступно только в режиме вычисления формул и располагается на места окна ввода операнда X1
* Строка состояний операции. Здесь выводятся сообщения об ошибке ввода данных и ошибках вычисления, а так же сообщения об успешности операции.
* Главное меню. Содержит две опции: Файл и Справка. В опции Файл содержатся команды «Открыть тестовый файл» и закрытия программы. В опции Справка содержится команда открытия окна «О программе».

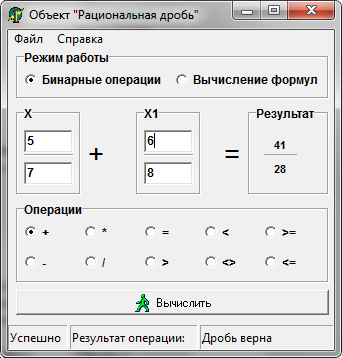


Рисунок 3. Главная форма программы

Форма «О программе» содержит сведения о теме проекта, разработчике и руководителе проекта. Форма открывается по команде «Справка -> О программе» главного меню.

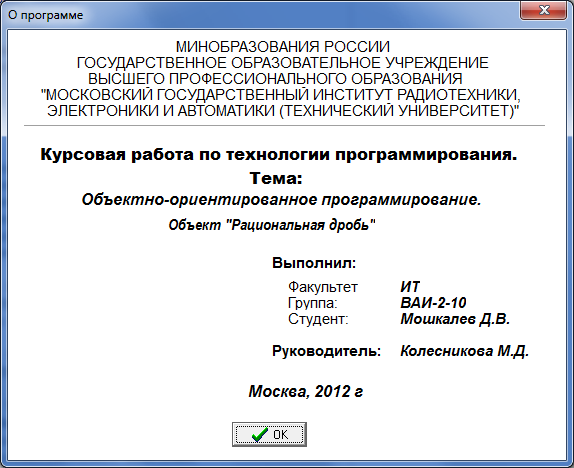


Рисунок 4. Форма "О программе"

Форма «Результаты тестирования данными из файла» предназначена для контроля результатов автоматического тестирования программы по заранее подготовленным данным.

Форма открывается в немодальном режиме после выбора опции главного меню «Открыть тестовый файл». Результаты тестирования будут отображены на компоненте Memo.

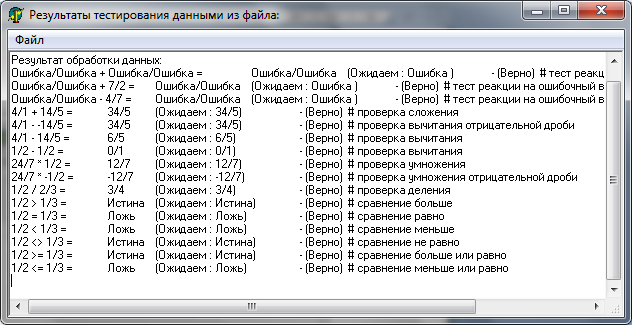


Рисунок 5. Форма «результаты тестирования данными из файла»

Свойства объектов форм программы сведены в ***Таблица 1.*** Свойства объектов

Таблица 1. Свойства объектов

| *№* | *Объект* | *Тип Объекта* | *Наименование свойств объекта* | *Значение свойства объекта* |
| --- | --- | --- | --- | --- |
| **Главная форма** | | | | |
| 1 | Form1 | TForm1 | Caption | Объект «Рациональная дробь» |
| OnCreate | FormCreate |
| OnShow | FormShow |
| 2 | BitBtn1 | TBitBtn | Caption | Вычислить |
| OnClick | Calculate |
| 3 | F1 | TGroupBox | Caption | F(X) |
| Visible | False |
| 4 | Label2 | TLabel | Caption | 2X+ |
| 5 | Label3 | TLabel | Caption | 1 |
| 6 | Label4 | TLabel | Caption | X |
| 7 | F2 | TGroupBox | Caption | F(X) |
| Visible | False |
| 8 | Label6 | TLabel | Caption | X-1 |
| 9 | Label7 | TLabel | Caption | X+2 |
| 10 | F3 | TGroupBox | Caption | F(X) |
| Visible | false |
| 11 | Label10 | TLabel | Caption | 2 |
| 12 | Label11 | TLabel | Caption | - |
| 13 | Label5 | TLabel | Caption | X |
| 14 | Label8 | TLabel | Caption | 1 |
| 15 | Label9 | TLabel | Caption | X |
| 16 | F4 | TGroupBox | Caption | F(X) |
| Visible | false |
| 17 | Label12 | TLabel | Caption | - |
| 18 | Label13 | TLabel | Caption | 1 |
| 19 | Label14 | TLabel | Caption | 1+X |
| 20 | Label15 | TLabel | Caption | X |
| 21 | Label16 | TLabel | Caption | 3 |
| 22 | F5 | TGroupBox | Caption | F(X) |
| Visible | false |
| 23 | Label17 | TLabel | Caption | + |
| 24 | Label18 | TLabel | Caption | 5X |
| 25 | Label19 | TLabel | Caption | 11 |
| 26 | Label20 | TLabel | Caption | 3 |
| 27 | Label21 | TLabel | Caption | X |
| 28 | Functions1 | TRadioGroup | Caption | Функции |
| Items | F1(X) |
| 28 | Functions1 | TRadioGroup | Items | F2(X)  F3(X)  F4(X)  F5(X) |
| Visible | False |
| OnClick | OperatorSelectClick |
| 29 | Label1 | TLabel | Caption | = |
| 30 | MainMenu1 | TMainMenu | Items | Файл  Открыть тестовый файл  Закрыть программу  Справка  О Программе |
| 31 | ModeSelect | TRadioGroup | Items | Бинарные операции  Вычисление формул |
| OnClick | ModeSelectClick |
| 32 | Operand1 | TGroupBox | Caption | X |
| 33 | EnterDenom | TEdit | Text | 1 |
| OnChange | OperatorSelectClick |
| 33 | EnterNum | TEdit | Text | 1 |
| OnChange | OperatorSelectClick |
| 34 | Operand2 | TGroupBox | Caption | X1 |
| 33 | EnterDenom2 | TEdit | Text | 1 |
| OnChange | OperatorSelectClick |
| 33 | EnterNum2 | TEdit | Text | 1 |
| OnChange | OperatorSelectClick |
| 34 | Operators1 | TRadioGroup | Caption | Операции |
| Columns | 5 |
| 34 | Operators1 | TRadioGroup | Items | +  -  \*  /  =  >  <  <>  >=  <= |
| OnClick | OperatorSelectClick |
| 35 | OperatorSymbol | TLabel | Caption | + |
| 36 | ResultGroup | TGroupBox | Caption | Результат |
| 37 | CmpRes1 | TLabel | Caption | 1 |
| Visible | false |
| 38 | ResDenom | TLabel | Caption | 1 |
| 39 | ResNum | TLabel | Caption | 1 |
| 40 | StatusBar | TStatusBar | Panels | 0  1  2 |
| **Форма «О программе»** | | | | |
| 1 | Form2 | TForm2 | Caption | О программе |
| Visible | false |
| 2 | Label1 | TLabel | Caption | Курсовая работа по технологии программирования |
| 3 | Label2 | TLabel | Caption | Тема: |
| 4 | Label3 | TLabel | Caption | Объектно-ориентированное программирование |
| 5 | Label4 | TLabel | Caption | Выполнил |
| 6 | Label5 | TLabel | Caption | Группа |
| 7 | Label6 | TLabel | Caption | Факультет |
| 8 | Label7 | TLabel | Caption | Студент |
| 9 | Label8 | TLabel | Caption | Руководитель |
| 10 | Label9 | TLabel | Caption | ИТ |
| 11 | Label10 | TLabel | Caption | ВАИ-2-10 |
| 12 | Label11 | TLabel | Caption | Мошкалев Д.В. |
| 13 | Label12 | TLabel | Caption | Колесникова М.Д. |
| 14 | Label13 | TLabel | Caption | Москва, 2012г |
| 15 | Label14 | TLabel | Caption | МИНОБРАЗОВАНИЯ РОССИИ |
| 16 | Label15 | TLabel | Caption | ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧЕРЕЖДЕНИЕ |
| 17 | Label16 | TLabel | Caption | ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ |
| 18 | Label17 | TLabel | Caption | МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ИНСТИТУТ РАДИОТЕХНИКИ |
| 19 | Label18 | TLabel | Caption | ЭЛЕКТРОНИКИ И АВТОМАТИКИ (ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ) |
| 20 | Label19 | TLabel | Caption | Объект «Рациональная дробь» |
| 21 | OK | TBitBtn | Caption | OK |
| ModalResult | mrOk |
| **Форма «Результаты тестирования данными из файла»** | | | | |
| 1 | Form3 | TForm3 | Caption | Результаты тестирования данными из файла |
| Visible | false |
| 2 | MainMenu1 | TMainMenu | Items | Сохранить  Закрыть |
| 3 | OpenDialog1 | TOpenDialog | DefaultExt | \*.txt |
| 4 | SaveDialog1 | TSaveDialog | DefaultExt | \*.log |
| 5 | Memo1 | TMemo | Align | alClient |

# ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

## Тестирование программы

Тестирование оценивает корректность выполнения приложения. Программа тестируется с целью обнаружения ошибок, а также с целью проверки соответствия всем требованиям поставленной задачи.

1. Тестирование операций сложения и вычитания. Алгоритм работы программы в режиме бинарных операций подробно описан в руководстве пользователя "**1** ***Режим бинарных операций***.

Входные данные:

Эталонные данные:

Результат работы программы представлен на ***Рисунок 6*** и ***Рисунок 7***.

|  |  |
| --- | --- |
| Рисунок . Результаты тестирования операции сложения | Рисунок . Результат тестирования операции вычитания |

1. Тестирование операций умножения и деления

Входные данные:

Эталонные данные: Для умножения: . Для деления:

Результат работы программы представлен на Рисунок 8 и Рисунок 9.

|  |  |
| --- | --- |
| Рисунок . Результат тестирования операции умножения | Рисунок . Результат тестирования операции деления |

1. Тестирование операций сравнения

Входные данные:

Эталонные данные:

Результат работы программы представлен на рисунках ***Рисунок*** 10***, Рисунок*** 11***, Рисунок*** 12***, Рисунок*** 13***, Рисунок*** 14и ***Рисунок*** 15***.***

|  |  |
| --- | --- |
| Рисунок . Результат тестирования операции равно | Рисунок .Результат тестирования операции не равно |
| Рисунок . Результат тестирования операции больше | Рисунок . Результат тестирования операции больше или равно |
| Рисунок . Результат тестирования операции меньше | Рисунок . Результат тестирования операции меньше или равно |

1. Тестирование вычисления функций

Входные данные:

Эталонные данные:

Результат работы программы представлен на рисунках **Ошибка! Источник ссылки не найден.Ошибка! Источник ссылки не найден.**Рисунок 18Рисунок 19 и Рисунок 20***.***

|  |  |
| --- | --- |
| Рисунок . Результат тестирования функции a) | Рисунок . Результат тестирования функции b) |
| Рисунок . Результат тестирования функции c) | Рисунок . Результат тестирования функции d) |
|  |  |
| Рисунок . Результат тестирования функции e) |  |

1. Тестирование данными из файла. Алгоритм работы программы в режиме пакетного тестирования описан в Руководстве оператора, стр. 29 Выполнение пакетного тестирования.

Входные данные: Файл «File1.txt»

Эталонные данные: находятся в тестовом файле.

Результат работы программы представлен на Рисунок 21.

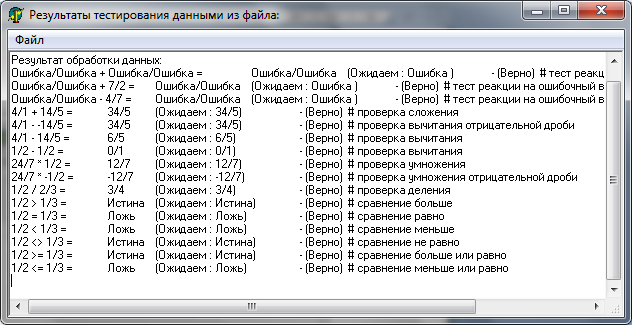


Рисунок . Результаты тестирования с использованием файла тестовых данных File1.txt

# ЗАКЛЮЧЕНИЕ

Объектно-ориентированное программирование – удобный и мощный инструмент разработки, как готовых программных продуктов, так и библиотечных модулей.

Однажды разработанный и отлаженный класс может использоваться без изменений в огромном количестве приложений, экономя время на разработку и увеличивая качество получившегося продукта. Причем программисты, использующие объекты этого класса могут пользоваться им как «черным ящиком» - не зная подробностей реализации класса.

Класс, разработанный в этой работе, может использоваться в математических вычислениях повышенной точности, т.е. в тех случаях, когда представление чисел в виде чисел с плавающей точкой имеет недостаточную точность. Дальнейшее усовершенствование и изменение функционала класса под конкретную задачу лучше всего делать, используя механизм наследования.

Что касается усовершенствования программы, демонстрирующей функционал класса «рациональная дробь», то тут можно так усовершенствовать разбор тестовых файлов, чтобы можно было задавать в тестах большие формулы с использованием дробей. Можно добавить возможность создания формул для расчетов в реальном времени, а не использовать запрограммированные ранее.

# СПИСОК ЛИТЕРАТУРЫ

* *Нил Дж. Рубенкинг.* Язык программирования Delphi для «чайников». Введение в Borland Delphi 2006 = Delphi for Dummies. — М.: Диалектика, 2007. — 336 с. — ISBN 0-7645-0179-8
* *Хавьер Пашеку.* Программирование в Borland Delphi 2006 для профессионалов = Delphi for .NET Developer’s Guide. — М.: Вильямс, 2006. — 944 с. — ISBN 0-672-32443-X
* *А. Н. Вальвачев, К. А. Сурков, Д. А. Сурков, Ю. М. Четырько.* Программирование на языке Delphi. Учебное пособие. — 2005.
* К. И. Шахгельдян Курс «Объектно-ориентированное программирование»
* А. С. Усов «Внутри Объектной Технологии: Теория, Архитектура, Концепции»

# ПРИЛОЖЕНИЕ 1. Руководство оператора

### Режим бинарных операций.

Внешний вид окна в режиме бинарных операций показан на **Ошибка! Источник ссылки не найден.**. На ***Рисунок 23*** показан внешний вид окна при выполнении логических операций. На ***Рисунок 24*** показан внешний вид программы при обнаружении ошибки.

**Алгоритм работы программы в режиме бинарных операций**:

1. Для выполнения операции установить переключатель режима работы в положение "Бинарные операции".
2. Установить переключатель операций в положение, соответствующее требуемой операции. Для алгебраических операций : "+"(сложение), "-"(вычитание), "\*"(умножение), "/"(деление). Для операций сравнения: "="(равно), "<>"(не равно), "<"(меньше), ">"(больше), "<="(меньше или равно), ">=" (больше или равно).
3. Установить значение числителя и знаменателя первого операнда в блоке "X". Верхнее окно ввода в блоке "X" соответствует числителю, нижнее - знаменателю.
4. Установить значение числителя и знаменателя второго операнда в блоке "X1". Верхнее окно ввода в блоке "X1" соответствует числителю, нижнее - знаменателю.
5. По состоянию панели состояния оценить успешность выполнения операции. ( успешным является значение "Успешно, Результат операции: Дробь верна")
6. Получить результат выбранной операции с дробями в блоке "Результат".

Для вычислительных операций ("**+**"(сложение), "**-**"(вычитание), "**\***"(умножение), "**/**"(деление)), в блоке "Результат" будет отображаться дробь, вычисленная в соответствии с выбранной на шаге 2 операции.

Для операций сравнения ("**=**"(равно), "**<>**"(не равно), "**<**"(меньше), "**>**"(больше), "**<=**"(меньше или равно), "**>=**" (больше или равно)), в блоке "Результат" будут отображаться значения "**ИСТИНА**" или "**ЛОЖЬ**" в зависимости от результата сравнения в соответствии с выбранной на шаге 2 операции.

|  |  |
| --- | --- |
|  |  |
| Рисунок . Режим бинарных операций, операция сложения | Рисунок . Режим операций сравнения, операция "Равно" |
|  |  |
| Рисунок . Реакция на ошибки в вычислениях, деление на "0" во втором операнде |  |

### Режим вычисления формул.

В соответствии с техническим заданием пункт 1.2, программа выполняет вычисления по пяти заранее заданным формулам:

Внешний вид окна в режиме вычисления формул показан на ***Рисунок 25.*** Ошибки, возникающие в программе в процессе вычисления отображаются аналогично показанному на ***Рисунок 24.*** Алгоритм работы программы в режиме вычисления формул:

1. Для выполнения операции установить переключатель режима работы в положение "вычисление формул".
2. Установить переключатель функции в положение, соответствующее требуемой функции.
3. Проверить, что в блоке "F(X)" появилась требуемая формула.
4. Установить значение числителя и знаменателя операнда в блоке "X". Верхнее окно ввода в блоке "X" соответствует числителю, нижнее - знаменателю.
5. По состоянию панели состояния оценить успешность выполнения операции. ( успешным является значение "Успешно, Результат операции: Дробь верна")
6. Получить результат выбранной операции с дробями в блоке "Результат".

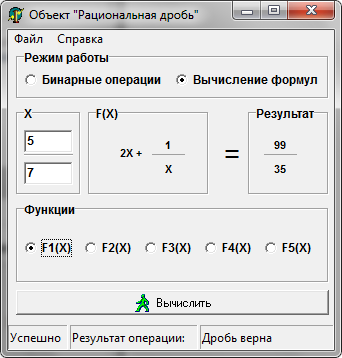


Рисунок 25. Режим вычисления формул

### Выполнение пакетного тестирования.

В соответствии с техническим заданием пункт 2, программа выполняет тестирование на основе данных, полученных из файла. Пример тестового файла приведен в ПРИЛОЖЕНИИ 2. Листинг программы, Файл тестовых данных.

Алгоритм выполнения тестирования:

1. Для выполнения операции выбрать в пункте меню "Файл" опцию "Открыть текстовый файл".
2. В открывшемся диалоговом окне "Открыть" выбрать текстовый файл, содержащий тестовые данные и Нажать кнопку "Открыть".
3. В открывшемся окне "Результаты тестирования данными из файла" получить результат проведенных тестов.(Рисунок 26).
4. Для сохранения результатов тестирования в файл выбрать в пункте меню "Файл" опцию "Сохранить".
5. В открывшемся диалоговом окне "Сохранить как" указать имя файла, в который необходимо сохранить результаты тестирования.

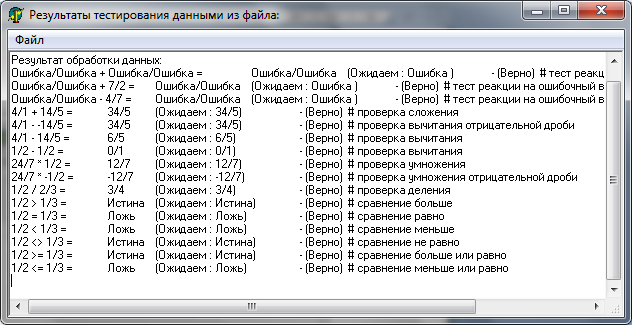


Рисунок 26. Форма результатов обработки файла тестовых данных

### 

### Получение информации о программе.

1. Для получения информации о программе выбрать в пункте меню "Справка" опцию "О программе".
2. В открывшемся диалоговом окне "О программе" находится информация о проекте.
3. Для закрытия окна "О программе" нажать кнопку "ОК" или "x".

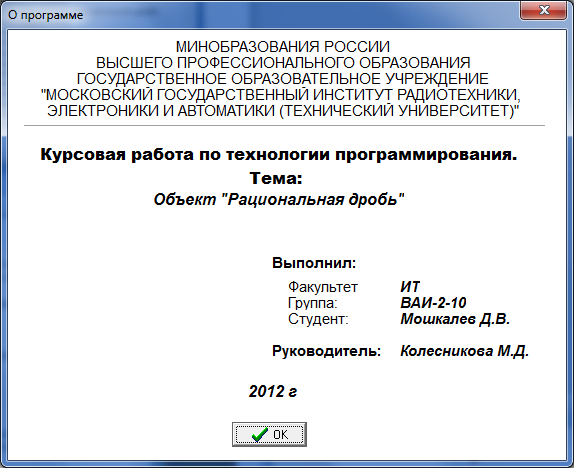


Рисунок 27. Форма с информацией о программе

# ПРИЛОЖЕНИЕ 2. Листинг 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, res: **integer**;

**begin**

m := a;

n := b;

r := 0;

res := 0;

**repeat**

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

**if** (n = 0) **o**r (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 + 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*

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

*// Операнд равен*

**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**;

*// текущий объект больше или равен операнду*

**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**;

*//текущий объект больше операнда*

**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**;

*//текущий объект меньше или равен операнду*

**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**;

*//текущий объект меньше операнда*

**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**;

*//текущий объект не равен операнду*

**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, **Var**iants, **Class**es, Graphics, Controls, Forms,

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

ToolWin, RatFrac**Class**, AboutForm, LogForm;

**type**

TForm1 = **class**(TForm)

MainMenu1: TMainMenu;

N1: TMenuItem;

N2: TMenuItem;

N3: TMenuItem;

N4: TMenuItem;

N5: TMenuItem;

N6: TMenuItem;

StatusBar1: TStatusBar;

BitBtn1: TBitBtn;

F1: TGroupBox;

F2: TGroupBox;

F3: TGroupBox;

F4: TGroupBox;

F5: TGroupBox;

Operand1: TGroupBox;

Operand2: TGroupBox;

ResultGroup: TGroupBox;

ResDivider: TBevel;

Bevel1: TBevel;

Bevel26: TBevel;

EnterNum: TEdit;

EnterDenom: TEdit;

EnterNum2: TEdit;

EnterDenom2: TEdit;

CmpRes1: TLabel;

Label1: TLabel;

ResNum: TLabel;

ResDenom: TLabel;

OperatorSymbol: TLabel;

Bevel2: TBevel;

Bevel3: TBevel;

Bevel4: TBevel;

Bevel5: TBevel;

Bevel6: TBevel;

Bevel7: TBevel;

Bevel8: TBevel;

Label2: TLabel;

Label3: TLabel;

Label4: TLabel;

Label6: TLabel;

Label7: TLabel;

Label5: 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;

Operators1: TRadioGroup;

Functions1: TRadioGroup;

ModeSelect: TRadioGroup;

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

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

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

**procedure** ModeSelectClick(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);

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

private

Operation : short; *// режим вычислений*

*{ Private declarations }*

public

*{ Public declarations }*

**end**;

**var**

Form1: TForm1;

Number : **integer**;

**implementation**

*{$R \*.dfm}*

*// Функция выполняет вычисления всех операций в*

*// зависимости от переменной Operation*

*// входные данные перечитываются из Элементов формы*

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

**var**

a, b : **integer**;

CompResult : **boolean**;

X, X1, Res, N, Res1, Res2 : TRationalFraction;

**begin**

CompResult := **false**;

*// создаём рабочие объекты*

Res := TRationalFraction.Create;

X := TRationalFraction.Create;

X1 := TRationalFraction.Create;

*// 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** *// сообщаем об ошибке в панель статуса*

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** *// сообщаем об ошибке в панель статуса*

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** *// сообщаем об ошибке в панель статуса*

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 : **begin** *// =*

CompResult := X.Eq(X1);

**end**;

5 : **begin** *// >*

CompResult := X.Gt(X1);

**end**;

6 : **begin** **// <**

CompResult := X.Lt(X1);

**end**;

7 : **begin** **// <>**

CompResult := X.Ne(X1);

**end**;

8 : **begin** *// >=*

CompResult := X.Ge(X1);

**end**;

9 : **begin** *// <=*

CompResult := X.Le(X1);

**end**;

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.Add(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**; *//F(x) = 3/x + 5 \* x/11*

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** *// сообщаем об ошибке в панель статуса*

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

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

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

**exit**;

**end** **else**

**begin** *// сообщаем об успехе в панель статуса*

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**;

// освобождаем ресурсы

X.Free;

X1.Free;

Res.Free;

**end**;

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

**begin**

Operation := 0; // +

**end**;

*// функция-обработчик сообщений о выборе операции на форме*

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

**var**

isCompare : bool;

**begin**

isCompare := **false**;

**if** (ModeSelect.ItemIndex = 0) **then**

**begin**

OperatorSymbol.Caption := Operators1.Items.Strings[Operators1.ItemIndex];

Operation := Operators1.ItemIndex;

**if**(Operation >= 4) **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** **else**

**begin**

Operation := Functions1.ItemIndex + 10;

F1.Visible := **false**;

F2.Visible := **false**;

F3.Visible := **false**;

F4.Visible := **false**;

F5.Visible := **false**;

**case** Functions1.ItemIndex **of**

0: F1.Visible := **true**;

1: F2.Visible := **true**;

2: F3.Visible := **true**;

3: F4.Visible := **true**;

4: F5.Visible := **true**;

**end**;

**end**;

Calculate(Sender);

**end**;

*// функция - обработчик выбора режима работы(простые операции/функции)*

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

**begin**

**if** ModeSelect.ItemIndex = 0 **then**

**begin**

Operand2.Visible := **true**;

Operators1.Visible := **true**;

Functions1.Visible := **false**;

OperatorSymbol.Visible := **true**;

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

Operators1.ItemIndex := 0;

F1.Visible := **false**;

F2.Visible := **false**;

F3.Visible := **false**;

F4.Visible := **false**;

F5.Visible := **false**;

**end** **else**

**begin**

Operand2.Visible := **false**;

Operators1.Visible := **false**;

Functions1.Visible := **true**;

OperatorSymbol.Visible := **false**;

Operation := 10;

Functions1.ItemIndex := 0;

ResNum.Visible := **true**;

Resdenom.Visible := **true**;

ResDivider.Visible := **true**;

CmpRes1.Visible := **false**;

**end**;

OperatorSelectClick(Sender);

**end**;

*// функция выбирает математическую функцию для вычислений*

*// (устанавливает значение переменной Operation)*

*// и переключает видимость у элементов формы в зависимости от выбранной функции*

**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.ClearLog;

Form3.Show;

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

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

Form3.AddLogString(*'Открыт файл :* ' + Form3.OpenDialog1.FileName);

Form3.AddLogString(*'Результат обработки данных:* ');

**repeat**

Readln(f, s);

CheckStringOper(s, log);

**if**(log <> '') **then**

Form3.AddLogString(log);

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

CloseFile(f);

**end**;

**end**;

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

*// 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 опперации*

**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)) **do**

**begin**

i := i + 1;

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

**if**(InStr[i] = '#') **then**

**begin** *// дальше идет коментарий*

Comment := Copy(InStr, i, Length(InStr) - i + 1);

break;

**end**;

**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**; **end** **else** *// 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**;

**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 + #$9 + Res1.StrNumerator + '/' + Res1.StrDenominator +

#$9 + '*(Ожидаем* : ';

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

**begin**

LogStr := LogStr + *'Ошибка )* ';

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

LogStr:=LogStr + #$9 + '- *(Верно*)' + #$9 + Comment

**else**

LogStr:=LogStr + #$9 + '- *(Ошибка)*' + #$9 + Comment;

**end** **else**

**begin**

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

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

LogStr:=LogStr + #$9 + #$9 + *'- (Верно)'* + #$9 + Comment

**else**

LogStr:=LogStr + #$9 + #$9 + '- *(Ошибка)'* + #$9 + Comment;

**end**;

Result := b;

**end** **else**

**begin** *// операции сравнения*

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

LogStr := LogStr + #$9 + *'Истина* '

**else**

LogStr := LogStr + #$9 + *'Ложь* ';

LogStr := LogStr + #$9 + '(*Ожидаем* : ';

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

**begin**

LogStr := LogStr + *'Истина*)';

**end** **else**

**begin**

LogStr := LogStr + *'Ложь*) ';

**end**;

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

LogStr:=LogStr + #$9 + '- (*Верно*)' + #$9 + Comment

**else**

LogStr:=LogStr + #$9 + '- (*Ошибка*)' + #$9 + 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;

*//пробуем прочитатьь первое значение*

**try**

l := IntFromString(Line,PosFrom);

**except**

**on** **Exception** : EConvertError **do**

Result := Result or **true**;

**end**;

*// читаем символ "/"*

c := Line[PosFrom];

PosFrom:= PosFrom + 1;

*//пробуем прочитатьь второе значение*

**try**

m := IntFromString(Line,PosFrom);

**except**

**on** **Exception** : EConvertError **do**

Result := Result or **true**;

**end**;

**if**(Result) **then** m := 0; *// если ошибка, то 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**;

*// выход из программы*

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

**begin**

Application.Terminate;

**end**;

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

**begin**

EnterNum.Text := '1';

EnterDenom.Text := '1';

EnterNum2.Text := '1';

EnterDenom2.Text := '1';

OperatorSelectClick(Sender);

**end**;

**end**.

## Модуль LogForm

**unit** LogForm;

**interface**

**uses**

Windows, Messages, SysUtils, **Var**iants, **Class**es, 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);

**procedure**AddLogString(Log:string);

**procedure** ClearLog();

**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**;

**procedure** TForm3.AddLogString(Log: **string**);

**begin**

Memo1.Lines.Add(Log);

**end**;

**procedure** TForm3.ClearLog;

**begin**

Memo1.Clear;

**end**;

**end**.

## Модуль AboutForm

**unit** AboutForm;

**interface**

**uses**

Windows, Messages, SysUtils, **Var**iants, **Class**es, 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;

Label14: TLabel;

Label15: TLabel;

Label16: TLabel;

Label17: TLabel;

Label18: TLabel;

Label19: TLabel;

Bevel1: TBevel;

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 *# тест реакции на ошибочный ввод числа*

12/t6 + 7/2 = e *# тест реакции на ошибочный ввод числа*

65/0 - 4/7 = e *# тест реакции на ошибочный ввод числа*

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

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

12/3 - 14/5 = 6/5 *# проверка вычитания*

1/2 - 1/2 = 0/1 *# проверка вычитания*

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

24/7 \* -7/14 = -12/7 *# проверка умножения отрицательной дроби*

1/2 / 2/3 = 3/4 *# проверка деления*

1/2 > 1/3 = t *# сравнение больше*

1/2 = 1/3 = f *# сравнение равно*

1/2 < 1/3 = f *# сравнение меньше*

1/2 <> 1/3 = t *# сравнение не равно*

1/2 >= 1/3 = t *# сравнение больше или равно*

1/2 <= 1/3 = f *# сравнение меньше или равно*