МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение высшего профессионального образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТОРОЕНИЯ»

Человеко-машинный интерфейс

Методические указания

Санкт-Петербург

2007

Составители: доцент, канд.техн.наук В.П.Попов;
доцент, канд.техн.наук Н.В.Соловьев

В методических указаниях изложена последовательность действий разработчика прикладных программ в форме Windows-приложений, а также рассмотрены стандартные компоненты интерфейсов прикладных программ, используемые в среде визуально-ориентированного программирования Delphi. Приведен пример выполнения задания.

В качестве лабораторных работ студенту предлагается изучить и подробно описать в отчете свойства и реакции на события одного из заданных компонентов и разработать программу в форме Windows-приложения с полностью оформленным интерфейсом пользователя, решающую заданную преподавателем математическую или физическую задачу.

Методические указания предназначены для студентов, обучающихся по направлению «Информатика и вычислительная техника» и могут использоваться при изучении дисциплин «Человеко-машинный интерфейс», «Программирование на языках высокого уровня» и «Технология программирования».

Лабораторная работа №1

Стандартные компоненты интерфейсов прикладных программ

Цель работы: Ознакомить студентов с основными свойствами и поддерживаемыми событиями стандартных компонентов, применяемых для создания интерфейсов при разработке Windows-приложений в среде визуального программирования на объектно-ориентированном языке высокого уровня.

Общие сведения

В настоящее время существует довольно большое количество различных сред визуального программирования Windows-приложений, базирующихся на объектно-ориентированных языках высокого уровня. Например, среда Delphi, в основе которой лежит язык Pascal 7.0, или Visual Basic. Существуют и специализированные среды, предназначенные для создания приложений в той или иной области, например, среда Microsoft Access для разработки систем управления базами данных или среда Matlab для проектирования инженерных приложений.

Для максимально быстрой разработки Windows-приложений все среды визуального программирования объединяют в себе следующие элементы:

- высокопроизводительный компилятор с языка программирования высокого уровня в EXE-файлы;
- объектно-ориентированная модель компонентов, предназначенных как для создания стандартных интерфейсных элементов, так и для разработки новых классов объектов;
- быстрая среда разработки, содержащая полный набор визуальных средств, поддерживающих как создание пользовательских интерфейсов, так и обработку корпоративных данных, с использованием библиотеки визуальных и невизуальных компонентов;
- интегрированный отладчик, позволяющий контролировать выполнение отлаживаемого приложения в различных режимах, следить за значениями пере-

менных и оперативно изменять эти значения непосредственно в процессе отладки;

- контекстно-ориентированную справочную систему, позволяющую получить исчерпывающую информацию по конкретным элементам среды или библиотеки классов, включая их свойства и процедуры обработки событий;
- библиотеку классов, инкапсулирующих различные функции Windows API, предназначенные для создания стандартных и дополнительных интерфейсных элементов, управления базами данных, поддержки протоколов обмена и технологии мультимедиа.

Далее в качестве визуальной среды разработки Windows-приложений будет рассматриваться базовая версия широко известного продукта фирмы Borland – среды программирования Delphi.

Структура среды разработки IDE

Среда разработки в Delphi состоит из ряда элементов, всегда присутствующих на экране: главное окно с панелью инструментов и палитры компонентов, инспектор объектов, окно для визуального создания формы, редактор исходного текста программы, и утилит, которые становятся доступными в определенных случаях: дизайнер меню, графический редактор, справочная система и т.д. Типичное отображение состояния среды разработки на экране дисплея приведено на рисунке 1.

Главное окно содержит базовые команды типа File, Edit, Search, View, Compile, Run, Debug, Options, Tools, Help, инструментальные кнопки для быстрого доступа к командам и палитру компонентов.

Состав базовых команд в основном соответствует общепринятым:

- в меню File находятся команды для выполнения операций с проектами, модулями и отдельными файлами;
- команды меню Edit позволяют помимо редактирования работать с областью обмена данными, отменять действия и управлять отображением компонентов;
- меню Search предоставляет команды для поиска и замены указанных символов и строк;

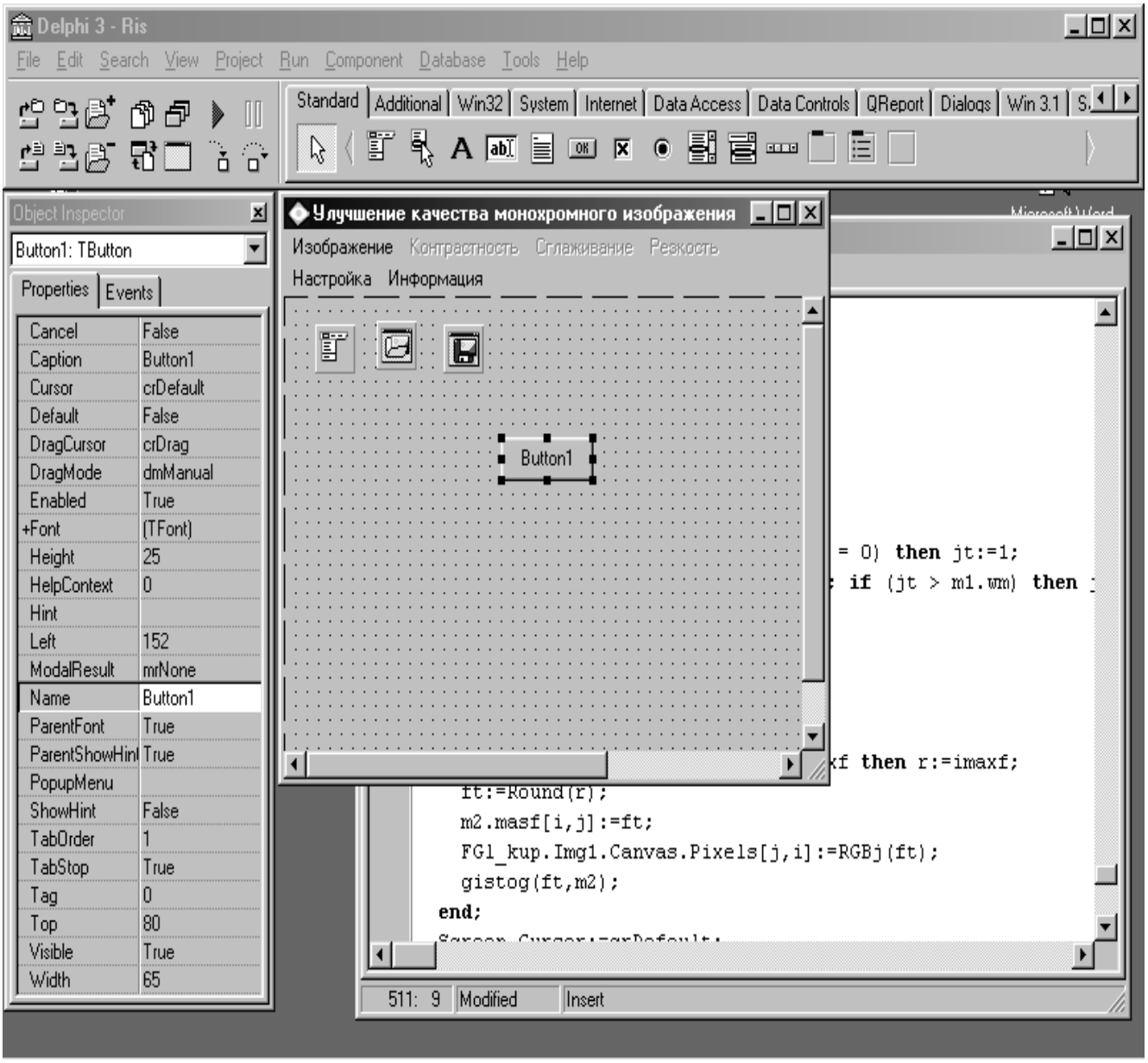


Рисунок 1 – Среда разработки Delphi

- в меню View содержатся команды для отображения различной информации и вызова информационных утилит;
- в меню Compile находятся команды для компиляции и сборки проектов, проверки синтаксиса и получения статистической информации о проекте;
- меню Run содержит команды для выполнения приложений в различных режимах, задания параметров командной строки и команд встроенного отладчика Debug;
- в меню Options сосредоточены команды задания различных параметров проекта, среды разработки и управления библиотекой компонентов;

- в меню Tools содержатся команды вызова различных утилит;
- меню Help предназначено для отображения различной информации справочного характера.

Палитра компонентов позволяет выбирать как визуальные, так и другие компоненты, которые будут присутствовать в разрабатываемой форме Windows-приложения. Компоненты сгруппированы по области их использования на отдельных страницах.

Основные страницы, присутствующие во всех версиях среды, следующие:

- стандартные компоненты, реализующие такие интерфейсные элементы среды Windows, как главное меню, локальное меню, текстовый элемент, строка редактирования, многострочный редактор, кнопки, списки;
- дополнительные компоненты, включающие в себя графические кнопки, редактор с шаблонным вводом, закладки, графические образы и др.;
- диалоговые панели для ввода и вывода файлов как общего назначения, так и специального, например, растровых изображений в bmp-формате;
- системные компоненты, предназначенные для управления системой, например, таймер, управление мультимедийными устройствами, управление обменом данными между приложениями;
- компоненты управления базами данных;
- компоненты для отображения данных;
- компоненты для формирования отчетов в базах данных.

Среда позволяет создавать новые компоненты и модифицировать существующие, что открывает перед разработчиками практически неограниченные возможности по разработке элементов интерфейсов с необходимыми свойствами и поведением.

Инспектор объектов позволяет устанавливать свойства объектов, расположенных в форме, и назначать методы – обработчики событий, на которые реагирует объект.

Менеджер проектов позволяет добавлять и удалять файлы, входящие в проект, перемещаться по файлам с исходным текстом и формам, входящим в проект.

Браузер объектов – графическое средство для просмотра иерархии объектов, входящих в стандартную библиотеку и составляющих данное приложение, а также просмотра предопределенных констант, процедур, типов данных, переменных, свойств и методов – унаследованных, виртуальных, защищенных, общих, открытых и закрытых.

Редактор используется для непосредственного написания кода в виде исходного текста программы на языке Pascal 7.0, причем код, создаваемый инспектором объектов и средой Delphi, тоже помещается в окно редактора и доступен для изменений.

Дизайнер меню предназначен для создания и модификации главного и локального меню. Имеется возможность сохранять и загружать меню в виде шаблонов, что позволяет использовать меню несколькими приложениями.

Графический редактор предназначен для создания и редактирования графических растровых изображений, иконок, курсоров и файлов ресурсов. Редактор поддерживает операции с областью обмена данными, различные шрифты, стили линий, кистей и т.п.

Встроенный отладчик позволяет выполнять пошаговую трассировку кода, назначать различные точки останова, следить за изменением значений переменных различного типа, узнавать результат выражений, просматривать стек вызова и многое другое.

Порядок выполнения лабораторной работы

1. Выбрать среду визуального программирования (рекомендуемые среды: Delphi, Visual Basic, Visual C++).
2. Выбрать объект из палитры интерфейсных компонентов (рекомендуемые страницы: стандартные, дополнительные, диалоговые компоненты).
3. Изучить свойства выбранного объекта и обрабатываемые им события.
4. Написать исполняемый код для одного из событий.
5. Отладить полученную программу.
6. Заполнить отчет по лабораторной работе.

Содержание отчета

1. Описание применения выбранного из палитры компонентов объекта.
2. Описание свойств объекта.
3. Описание событий, обрабатываемых объектом.
4. Текст исполняемого кода и результат обработки события.

Список рекомендуемых интерфейсных компонентов:

1. Главное меню (MainMenu)
2. Локальное меню (PoupMenu)
3. Статический текст (Label)
4. Строка редактирования (Edit)
5. Многостроковый редактор (Memo)
6. Стандартная кнопка (Buton)
7. Кнопка независимой фиксации (CheckBox)
8. Кнопка зависимой фиксации (RadioButton)
9. Список (ListBox)
10. Комбинированный список (ComboBox)
11. Полоса прокрутки (ScrollBar)
12. Группа (GroupBox)
13. Графическая кнопка (BitBtn)
14. Таблица (StringGrid)
15. Закладки (Tabs)
16. Статусная строка (StatusBar)
17. Открытие файла (OpenDialog)
18. Сохранение файла (SaveDialog)
19. Выбор шрифта (FontDialog)
20. Выбор цвета (ColorDialog)

Лабораторная работа №2

Разработка алгоритма задачи и текста прикладной программы

Цель работы: Основываясь на знаниях и навыках, полученных при изучении дисциплин «Алгоритмизация и структурное программирование» и «Программирование на языках высокого уровня» разработать алгоритм поставленной задачи с учетом особенностей выбранной среды визуального программирования.

Лабораторная работа представляет собой первый этап разработки алгоритма выполнения поставленной задачи и компоновки пользовательского интерфейса прикладной программы.

Приведенные ниже варианты заданий не выходят за рамки школьного курса физики и математики, а также аналитической геометрии и матричной алгебры, которые изучаются студентами на первом курсе ВУЗа. Информацию по методам решения можно самостоятельно найти в соответствующей литературе.

Среда программирования выбирается студентом самостоятельно из известных пакетов, поддерживающих визуальное программирование в стиле Windows, например, Visual Basic, Delphi, Visual C++.

Варианты задания:

1. Умножение матриц $C = A * B$.
2. Нахождение точек пересечения окружности и прямой линии.
3. Решение квадратного уравнения $ax^2 + bx + c = 0$.
4. Дальность полета и высота подъема тела при броске под углом.
5. Умножение комплексных чисел $a_1 + ib_1 + a_2 + ib_2 = a_3 + ib_3$.
6. Нахождение точек пересечения двух окружностей.
7. Решение системы вида $\begin{cases} y = \dfrac{a}{x} + b \\ y = cx + d \end{cases}$
8. Нахождение точки пересечения двух прямых.
9. Время движения тела при скольжении по наклонной плоскости.
10. Нахождение точек пересечения овала и прямой линии.

11.Нахождение обратного комплексного числа $\overline{a+ib}=\frac{a-ib}{a^2+b^2}$.

12.Суммарное сопротивление при различных соединениях.

13.Скорости тел после соударения (абсолютно упругий и неупругий удары).

14.Решение системы из двух линейных уравнений $\begin{cases} a_{11}x+a_{12}y=b_1 \\ a_{21}x+a_{22}y=b_2 \end{cases}$.

15.Расчет кпд кипятильника.

16.Нахождение точек пересечения гиперболы и прямой линии.

17.Суммарная емкость при различных соединениях.

18.Сложение матриц $C=A+B$.

19.Объем, давление и температура при различных термодинамических процессах

20.Нагревание и охлаждение, включая плавление и парообразование тел с учетом теплоемкости.

В процессе выполнения студент должен:

- выполнить описание поставленной задачи и выбранного метода решения,
- составить список исходных данных с указанием типа, возможного диапазона, значения по умолчанию;
- составить список выходных данных с указанием типа, возможного диапазона;
- разработать схему алгоритма и исполняемый модуль;
- разработать тестовые примеры.

При большом объеме входных и выходных данных необходимо предусмотреть возможность их ввода из файла, редактирования и вывода в файл. Пример выполнения данной работы в среде визуального программирования C++ Builder 6 приведен в Приложении А, в среде Delphi – в Приложении Д.

Содержание отчета

1. .Формулировка задания и математическая модель.
2. Информационная модель (список исходных и выходных данных с указанием типа, возможного диапазона, значения по умолчанию).
3. Схема алгоритма решения поставленной задачи и текст исполняемого модуля с комментариями.
4. Тестовые примеры.

Лабораторная работа №3

Компоновка интерфейса прикладной программы

Цель работы: Получить навыки по разработке интерфейсных форм прикладных программ для многооконного приложения в среде визуального программирования.

Общие сведения

При разработке интерфейса формулируются определенные правила ведения диалога. Диалог строится из сообщений. Типы сообщений следующие:

- при вводе:

а) входные данные,

б) управляющие сообщения (команды, директивы и др.);

- при выводе:

а) сообщения об ошибках (при вводе данных и в ходе решения),

б) выходные данные,

в) сообщения о состоянии системы (что произошло или происходит в системе),

г) справка (если пользователь не может ответить на запрос системы),

д) подсказка (способ подтвердить готовность принять сообщение).

По способу ведения диалога в интерфейсе, диалоговые процессы делятся на:

- диалоги, управляемые системой:

а) меню (в виде списка, строки, блока, пиктограмм),

б) готовая форма,

в) вопросы и ответы;

- диалоги управляемые пользователем на базе команд (системы запросов на ввод).

При форматировании экрана в состав полей меню обычно включают:

- заголовок меню (объект необязательный),
- основной текст меню (список объектов выбора),
- заключительный текст (объект необязательный).

Форматирование экрана при диалоге включает этапы:

- решить, **какая информация** должна появляться на экране,
- определить **основной формат** этой информации,
- решить, **где** она должна появляться на экране (т.е. определить область вывода каждого поля),
- решить, какие **средства выделения** требуются для каждого поля (т.е. атрибуты полей),
- разработать **проект размещения данных** на экране,
- оценить эффективность этого размещения,
- принять меры (обновляемое сообщение, обновление строки состояний и др.), если время реакции системы (задержки) превысят приемлемое.

В последнее время традиционная структура интерфейса стала заменяться на многооконный интерфейс (всплывающее или вытягиваемое меню), где каждому источнику информации соответствуют различные окна.

В процессе выполнения студент должен разработать внешний вид форм для отображения входных и выходных данных, а также необходимой справочной информации. Обязательно использование всплывающих подсказок. При необходимости кроме основной формы допускается использование вспомогательных форм, а также гипертекстовых ссылок в пояснениях к работе с программой. Пример выполнения данной работы в среде визуального программирования C++ Builder 6, применительно к заданию из Приложения А, приведен в Приложении Б, в среде Delphi, применительно к заданию из Приложения Г – в Приложении Д.

Содержание отчета

1. Окно основной формы интерфейса.
2. Окно информации о программе.
3. Окно пояснения по работе с программой.

Лабораторная работа №4

Разработка сценариев и реакций на события по сценарию

Цель работы: Получить навыки по формированию сценариев возможных действий пользователя при работе с прикладной программой и разработке реакций программы на эти события.

Общие сведения

Динамика диалога (его сценарий) может быть описана сетью (графом) переходов, представленной на рисунке 2. Каждая вершина графа – состояние, в котором диалог выводит сообщение или ждёт сообщение от пользователя. Дуги графа – возможные переходы из одного состояния в другое. В соответствии с рисунком 2, если диалог находится в состоянии N1 и условие А выполняется (истинно), то диалог переходит в состояниеN2. Аналогично выполняются и другие переходы.

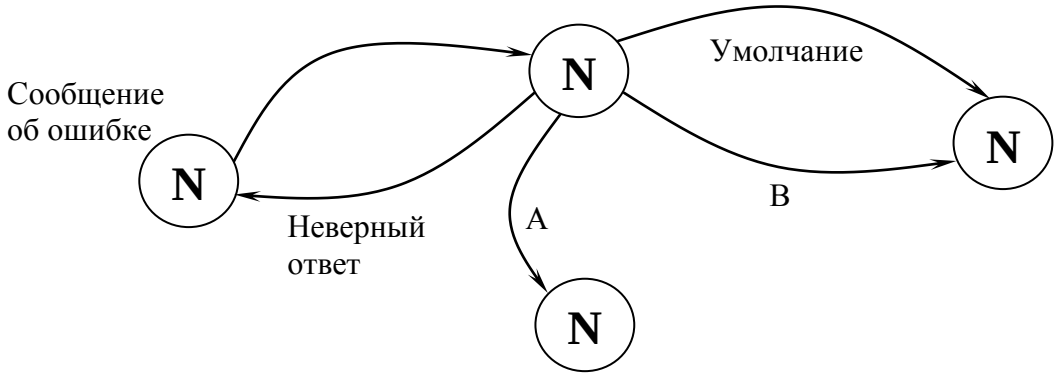


Рисунок 2

В процессе выполнения студенту необходимо учесть все возможные действия пользователя и предусмотреть для них соответствующую реакцию программы, для чего следует:

- разработать графы возможных сценариев работы программы;
- разработать процедуры обработки событий;
- произвести тестирование программы.

Пример выполнения данной работы в среде визуального программирования C++ Builder 6, применительно к заданию из Приложения А и с использованием экранных форм из Приложения Б, приведен в Приложении В, в среде Delphi, применительно к заданию из Приложения Г и с использованием экранных форм из Приложения Д – в Приложении Е.

Содержание отчета

1. Графы возможных сценариев работы программы.
2. Тексты процедур обработки событий.
3. Примеры экранных форм, полученных в результате различных сценариев работы программы.
4. Результаты тестирования.

Рекомендуемая литература

1. Коутс Р, Влейминк И. Интерфейс «человек-компьютер». – М.: Мир, 1990. – 501с.
2. Дарахвелидзе П.Г., Марков Е.П. Delphi – среда визуального программирования. – СПб.: BHV, 1996. – 352с.
3. Холзнер С. Visual C++. Учебный курс. – СПб.: Питер, 2006. – 570с.
4. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: построение и анализ. – М.: МЦНМО, 2004. – 856с.

Приложение А

Разработка алгоритма выполнения поставленной задачи

Необходимо написать программу для решения системы вида: $\begin{cases} y = \frac{a}{x} + b \\ y = cx + d \end{cases}$

Для решения этой системы уравнений необходимо вычесть из первого уравнения второе $\frac{a}{x} + b - cx - d = 0$ и умножить правую и левую часть на *x*, причем необходимо учесть что $x \neq 0$.

$$a + bx - cx^2 - dx = 0$$

$$cx^2 + (d - b)x - a = 0$$

В результате получается стандартное квадратное уравнение, дискриминант которого $D = (d - b)^2 + 4ac$.

Возможные значения *x* находятся по формуле:

$$x_{1,2} = \frac{-(d - b) \pm \sqrt{(d - b)^2 + 4ac}}{2c}$$

значения $y_{1,2}$ находится путем подстановки $x_{1,2}$ в любое уравнение исходной системы.

Схема алгоритма представлена на рисунке 1.

Исходными данными являются коэффициенты a, b, c, d, которые будут задаваться типом *double* , т.е. их значения должны будут находиться в диапазоне 1.7E-308..1.7E308

Выходными данными являются $x_{1,2}$ и $y_{1,2}$, которые так же будут задаваться типом *long double,* т.е. их значения должны будут находиться в диапазоне 3.4E-4932..3.4E+4932.

В качестве среды программирования была выбрана среда визуального программирования C++ Builder 6, позволяющая создавать Windows-приложения.

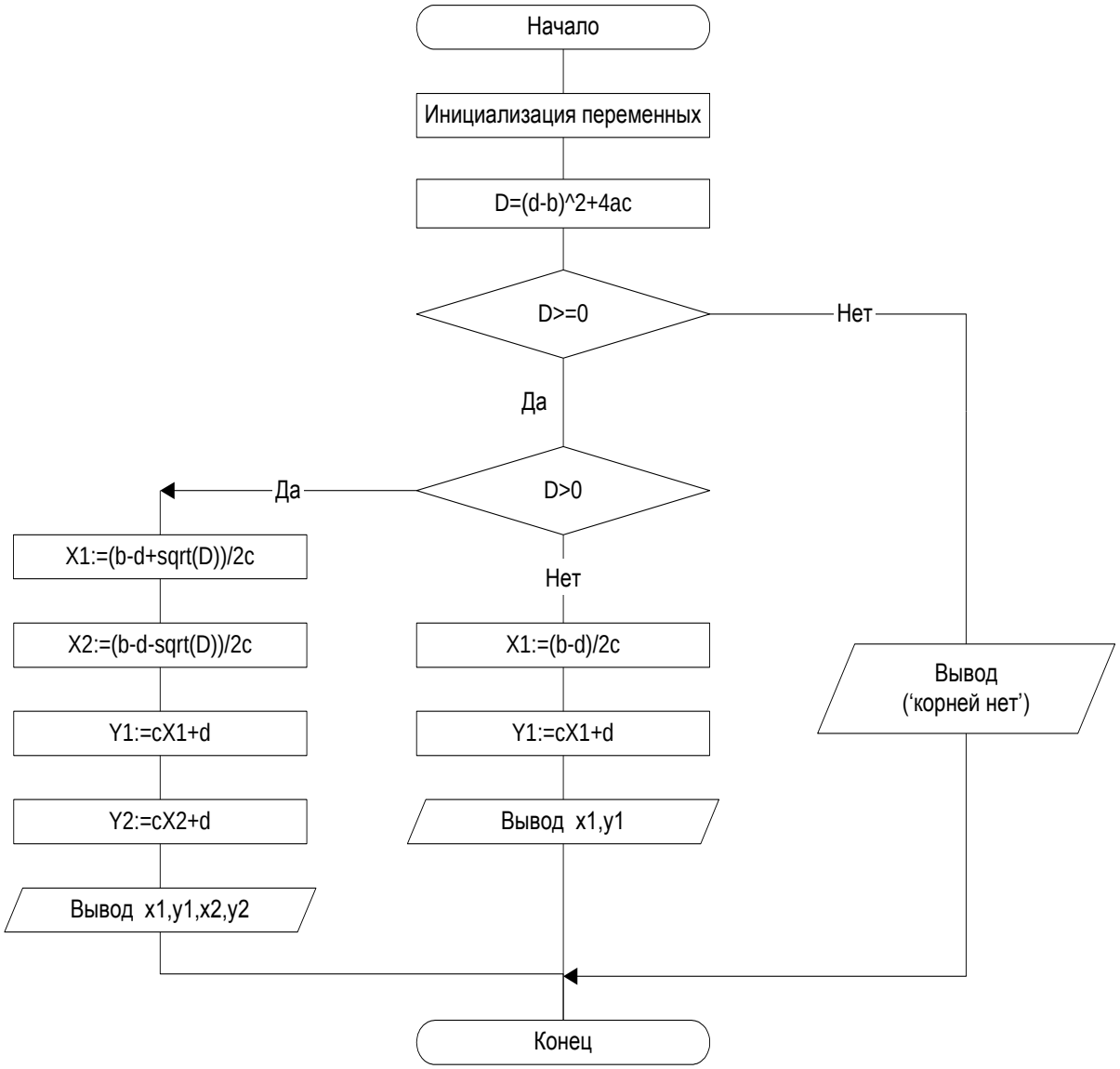


Рисунок А.1

Текст функции, выполняющей решение системы уравнений:

```
 int solution_equation(double a,double b, double c, double d,long double &resultX1,
long double &resultX2, long double &resultY1, long double &resultY2)
{
  //вычитаем из первого уравнения второе, домножаем всё на x и получаем: a + b*x
-c*x*x - d*x = 0   => -c*x*x + (b-d)*x + a = 0
  //временные переменные для преобразования типов
  long double tmp_a = a;
  long double tmp_b = b;
  long double tmp_c = c;
  long double tmp_d = d;
  //временные переменные для сохранения результатов
  long double x1 = 0, x2 = 0;
  long double y1 = 0, y2 = 0;
  //решение системы уравнений
  long double D =  (tmp_b - tmp_d)*(tmp_b - tmp_d) + 4*tmp_a*tmp_c;
```

```
 if (D < 0.0) return SQ_ERROR_NO_ROOT;
 if (D == 0.0)
  {
   //один корень
     if (tmp_c == 0) return SQ_ERROR_DIV_BY_ZERO;
     x1 = (tmp_b - tmp_d) /(2 * tmp_c );
     if (x1 == 0) return SQ_ERROR_DIV_BY_ZERO;
     y1 = tmp_c * x1 + tmp_d;
     resultX1 = x1; resultY1 = y1;
     resultX2 = x1; resultY2 = y1;
     return SQ_SUCCES_ALONE_ROOT;
  }
 if  (D > 0.0)
  {
   //два корня
     if (tmp_c == 0) return SQ_ERROR_DIV_BY_ZERO;
     x1 = (-1.0*(tmp_b - tmp_d) + sqrt(D)) /(-2 * tmp_c );
     y1 = tmp_c * x1 + tmp_d;
     x2 = (-1.0*(tmp_b - tmp_d) - sqrt(D)) /(-2 * tmp_c );
     y2 = tmp_c * x2 + tmp_d;
     if ((x1 == 0)&&(x2 != 0))
      {
       resultX1 = x2; resultY1 = y2;
       resultX2 = x2; resultY2 = y2;
       return SQ_SUCCES_ALONE_ROOT;
      }
     if ((x2 == 0)&&(x1 != 0))
      {
       resultX1 = x1; resultY1 = y1;
       resultX2 = x1; resultY2 = y1;
       return SQ_SUCCES_ALONE_ROOT;
      }
     if ((x1 == 0)&&(x2 == 0)) return  SQ_ERROR_DIV_BY_ZERO;
     resultX1 = x1; resultY1 = y1;
     resultX2 = x2; resultY2 = y2;
     return SQ_SUCCES_TWO_ROOT;
  }
}
```

Тестовые примеры:

1) Система умеет 2 решения:
Входные данные:
$a = 1$
$b = 2$
$c = 3$
$d = 4$
Выходные данные:
$x_1 = -1; y_1 = 1$
$x_2 = 0{,}33; y_2 = 5$

2)Система имеет 1 решение:
Входные данные:
$a = 2$
$b = 1$
$c = -2$
$d = 5$
Выходные данные:

$x_1 = 1; y_1 = 3$

2)Система не имеет решений:
Входные данные:
$a = -10$
$b = 1$
$c = 2$
$d = 3$
Выходные данные:
Система уравнений не имеет решения !

При большом объеме данных предусмотрен ввод данных из файла, а так же вывод результатов в файл. По умолчанию файлом для входных данных является input.ini . Вывод результатов осуществляется в файл results.ini.

Приложение Б

Компоновка форм

В программе используется четыре формы: главная форма, информация о программе, помощь и массовая обработка данных.

Главная форма представлена на рисунке Б.1.

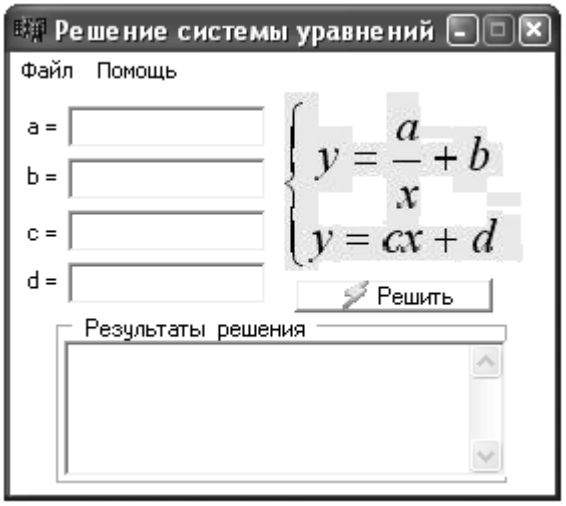


Рисунок Б.1

В главной форме происходит ввод исходных данных и вывод результатов. В этой форме использованы следующие элементы VBL:
Edit – ввод исходных данных;
Memo – вывод результатов;
Image – рисунок (исходная система);
Button – кнопка запуска решения;
Label – вспомогательная информация.

Форма с информацией о программе и авторе представлена на рисунке Б.2.

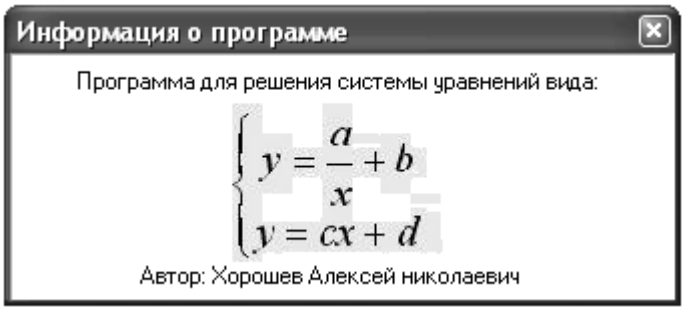


Рисунок Б.2

В этой форме использованы следующие элементы VBL:
Image – рисунок(исходная система);
Label – вспомогательная информация.

Форма, содержащая информацию, которая помогает пользователю работать с программой, представлена на рисунке Б.3.

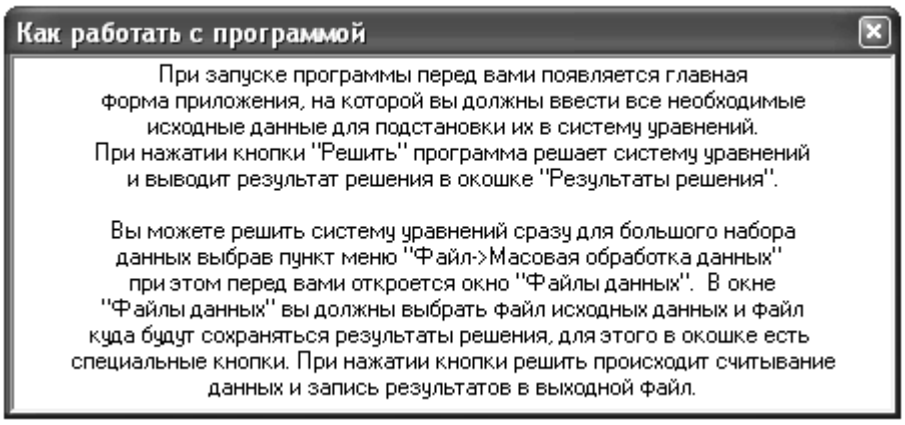


Рисунок Б.3

В этой форме использованы следующие элементы VBL:
Label – вспомогательная информация.

Форма, реализующая массовую обработку данных, представлена на рисунке Б.4.

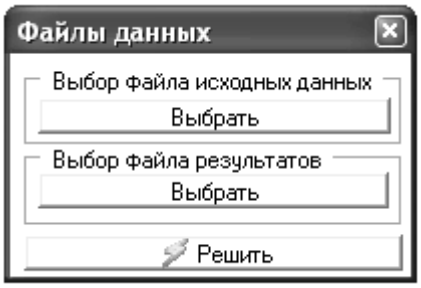


Рисунок Б.4

Этот режим работы используется при большом объеме входных и выходных данных. Ввод данных осуществляется из файла, вывод – в файл.
В этой форме использованы следующие элементы VBL:
Button – кнопка запуска решения;
Label – вспомогательная информация;
GroupBox - вспомогательная информация.

Приложение В

Разработка сценариев и реакций на события согласно сценариям.

Нажатие на кнопку “Решить”.

При нажатии на кнопку “Решить” происходит:

а) проверка правильности исходных данных, и если данные не соответствуют типу (или не введен какой либо из коэффициентов), то выводится сообщение «Все коэффициенты системы должны быть числами», как показано на рисунке В.1;

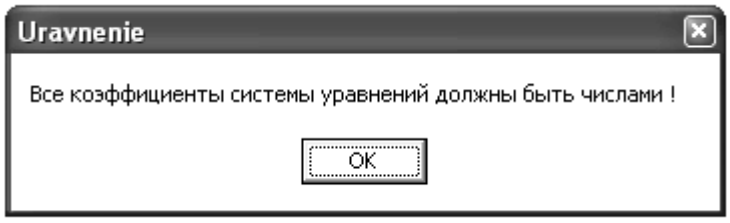


Рисунок В.1

б) решение системы уравнений (после решения отбрасываются все ответы, где $x = 0$);

в) вывод ответа в зависимости от количества найденных в пункте б решений. Если найдено 2 решения, то выводятся x_1, y_1, x_2, y_2, если найдено одно решение - x_1, y_1, если решений нет, то выводится сообщение «Система уравнений не имеет решения !»

Текст сценария:

```
void __fastcall TMainForm::BtRasschetClick(TObject *Sender)
{
MResult->Clear();
   double in_a = 0, in_b = 0, in_c = 0, in_d = 0;
   try
{
    in_a = StrToFloat(Eda->Text);
    in_b = StrToFloat(Edb->Text);
    in_c = StrToFloat(Edc->Text);
    in_d = StrToFloat(Edd->Text);
   }
  catch (...)
  {
   ShowMessage ("Все коэффициенты системы уравнений должны быть числами
!");
   MResult->Lines->Add("Все коэффициенты системы уравнений должны быть
числами !");
   return;
  }
  long double resX1,resX2,resY1,resY2;
  int reslt = solution_equation(in_a, in_b, in_c, in_d, resX1, resX2, resY1, resY2);
```

```
 if (reslt == SQ_SUCCES_TWO_ROOT)
  {
    MResult->Lines->Add("Система имеет 2 решения:");
    MResult->Lines->Add("x1 = " + FloatToStr(resX1) + "; y1 = "+ FloatToStr(resY1));
    MResult->Lines->Add("x2 = " + FloatToStr(resX2) + "; y2 = "+ FloatToStr(resY2));
  }
 if (reslt == SQ_SUCCES_ALONE_ROOT)
  {
   MResult->Lines->Add("Система имеет 1 решение:");
   MResult->Lines->Add("x = " + FloatToStr(resX1) + "; y = "+ FloatToStr(resY1));
  }
 if (reslt == SQ_ERROR_DIV_BY_ZERO)    MResult->Lines->Add("Система уравнений не имеет решения из-за деления на 0!");
 if (reslt == SQ_ERROR_NO_ROOT)        MResult->Lines->Add("Система уравнений не имеет решения !"); }
```

Массовая обработка данных.

При нажатии первой кнопки “Выбрать” открывается форма для выбора файла с входными данными, как показано на рисунке В.2.

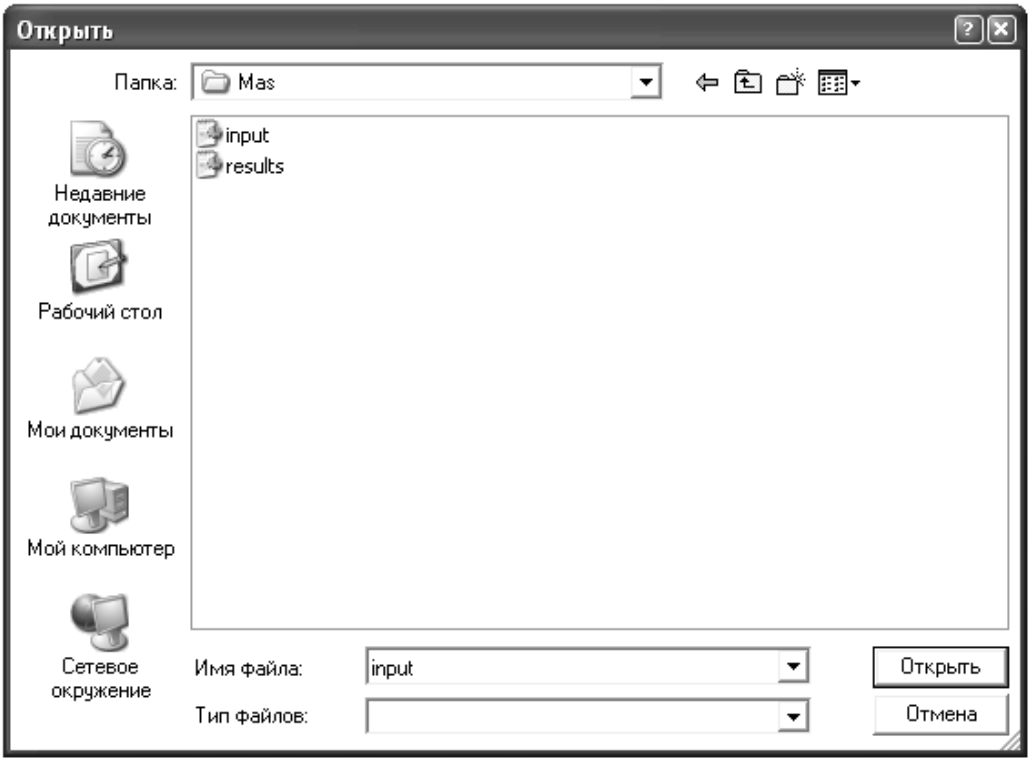


Рисунок В.2

Текст сценария:

```
void __fastcall TFormMas::SBInpClick(TObject *Sender)
{
if (OpenDialog1->Execute())
{
  if(FileExists(OpenDialog1->FileName))
```

```
  {
     NameFileOpen = OpenDialog1->FileName;
  }
 else
  ShowMessage("Такого файла не существует !");
}
}
```

При нажатии второй кнопки “выбрать” открывается форма для выбора файла с выходными данными, как показано на рисунке В.3.

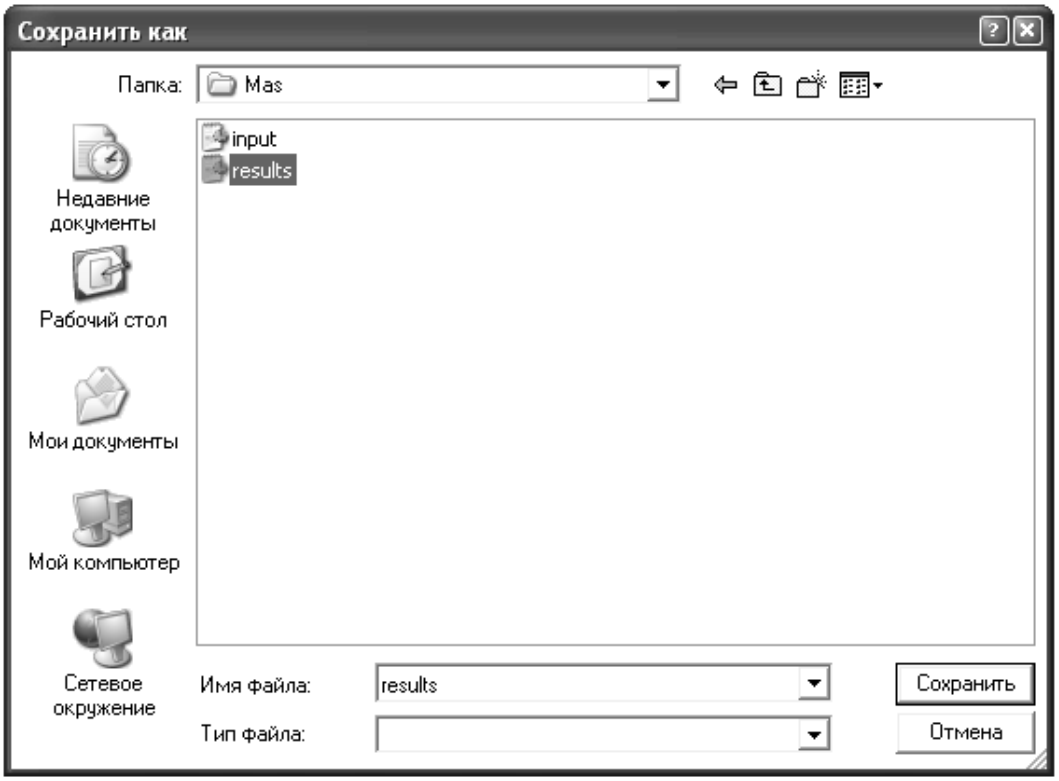


Рисунок В.3

Текст сценария:

```
void __fastcall TFormMas::SBOutClick(TObject *Sender)
{
if (SaveDialog1->Execute())
{
  NameFileSave = SaveDialog1->FileName;
}
}
```

При выборе кнопки решить происходит проверка входных данных, и если они не соответствуют заданному типу, то выводится сообщение « Указаны неправильные файлы данных !», как показано на рисунке В.4.

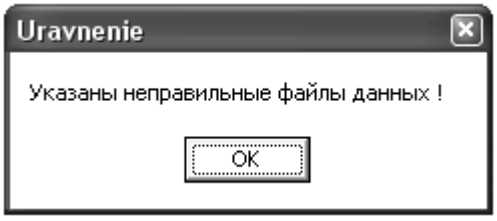


Рисунок В.4

Если не выбраны файл с исходными данными или файл для сохранения результатов, то выводится сообщение «Указаны не все файлы данных !», как показано на рисунке В.5.

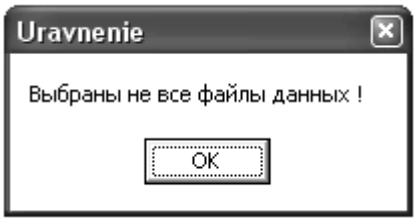


Рисунок В.5

Текст сценария.

```
void __fastcall TFormMas::SBRasschetClick(TObject *Sender)
{
try
{
if ((NameFileSave != "")&&(NameFileOpen != "")) solution_with_files();
 else ShowMessage("Выбраны не все файлы данных !");
}
catch(...)
{
 ShowMessage("Указаны неправильные файлы данных !");
 return;
}
}
```

Приложение Г

Разработка алгоритма выполнения поставленной задачи

1) Формулировка задания

Вариант 5.

Умножение комплексных чисел. $a_1 + ib_1 + a_2 + ib_2 = a_3 + ib_3$

2) Математическая модель

Произведением комплексных чисел $a+ bi$ и $c+ di$ называется комплексное число: ($ac – bd$) + ($ad + bc$) i . Это определение вытекает из двух требований:

- числа $a+ bi$ и $c+ di$ должны перемножаться, как алгебраические двучлены,
- число i обладает основным свойством: $i^2 = –1$.

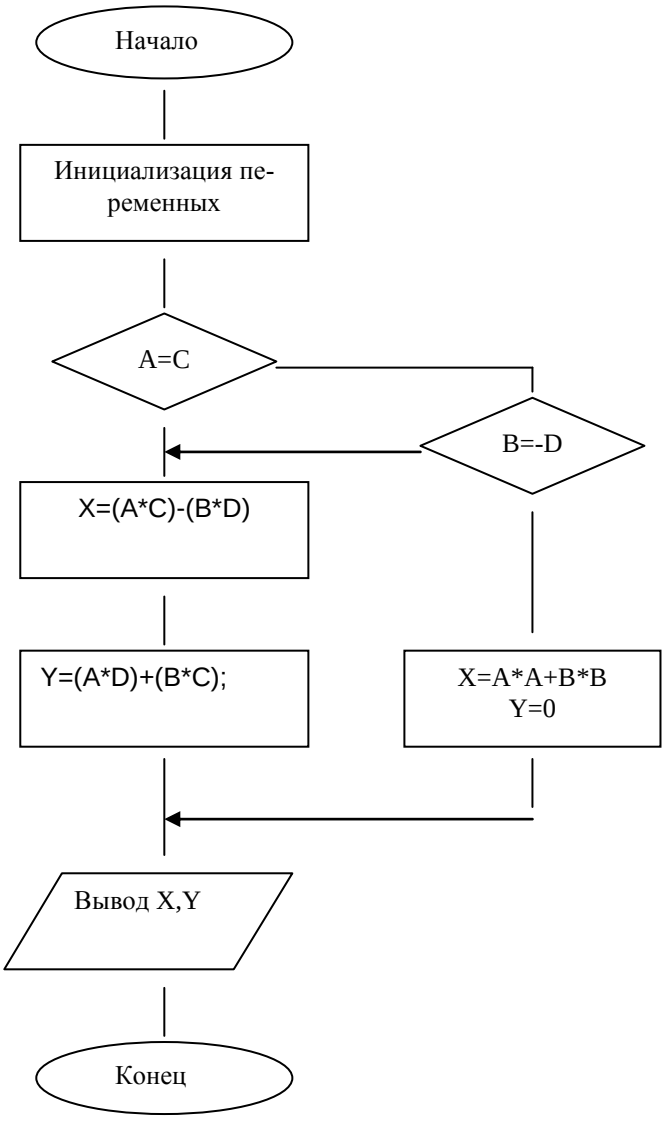
П р и м е р . ($a+ bi$)($a – bi$)= $a^2 + b^2$. Следовательно, *произведение двух сопряжённых комплексных чисел равно действительному положительному числу.*

.

3) Информационная модель

Статус данного	Содержание	Имя данного	Тип данного	Имя типа	Диапазон	Значение по умолчанию
Входные	коэффициенты первого числа	a	вещественный	Real	$5.0*10^{-324}$ .. $1.7*10^{308}$	0
		b				0
	коэффициенты второго числа	c				0
		d				0
Выходные	коэффициенты третьего числа	X	вещественный	Real	$5.0*10^{-324}$ .. $1.7*10^{308}$	0
		Y				0

4) Схема алгоритма



5) Текст исполняемого модуля

```
procedure TForm1.Button1Click(Sender: TObject);
Var a,b,c,d,X,Y: Comp;
   st: String;
   i: Integer;
begin
    If (Edit1.Text = '') Or (Edit2.Text = '') Or (Edit3.Text = '') Or (Edit4.Text = '')
     Then Begin MessageDlg(Требуется ввести все коэффициенты', mtError,
[mbOk],0);
               Exit;
          End;
     st:= Edit1.Text+Edit2.Text+Edit3.Text+Edit4.Text;
     For i:= 1 To Length(st)
     Do If Not (st[i] In ['0'..'9','-',',',])
       Then Begin MessageDlg(Введённые коэффициенты должны быть вещественными числами!',mtError,[mbOk],0);
```

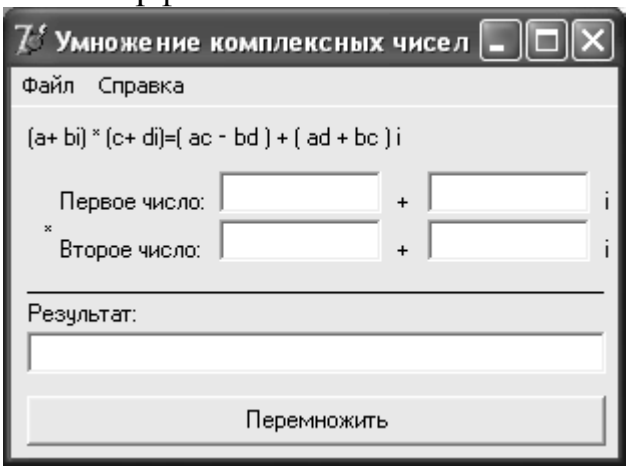
```
                Exit;
            End;
        a:= StrToFloat(Edit1.Text);
        b:= StrToFloat(Edit2.Text);
        c:= StrToFloat(Edit3.Text);
        d:= StrToFloat(Edit4.Text);
            {Случай сопряжённых чисел}
        If (a=c)and(b=-d)
        Then Begin X:=a*a+b*b;
                Edit5.Text:=FloatToStr(X);
            End
        Else Begin X:=(a*c)-(b*d);
                Y:=(a*d)+(b*c);
                Edit5.Text:= FloatToStr(X)+'+('+FloatToStr(Y)+'i)';
            end;
end;

procedure TForm4.Button3Click(Sender: TObject);
Var st,stn,stg,str: String;
   X,Y: Real;
   A: TMas;
   i,j,k: Integer;
Begin If (F1OK = False) Or (F2OK = False)
    Then Begin MessageDlg('Требуется задать имена входного и исходного файлов',mtError,[mbOk],0);
                Exit;
            End;
        AssignFile(F1,F1Name); AssignFile(F2,F2Name);
        Reset(F1); Rewrite(F2); k:= 0;
        While Not(EOF(F1))
        Do Begin ReadLn(F1,st); Inc(k);
                If st[Length(st)] <> ';'
                Then Begin MessageDlg(‘Ошибка в файле: строка '+IntToStr(k)+' не оканчивается на ";"!',mtError,[mbOk],0);
                        Exit;
                    End;
                i:= 1; j:= 1; stg:= '';
                While True
                Do Begin stn:= st[i];
                        If Not (stn[1] In ['0'..'9','-',',',';',' '])
                        Then Begin MessageDlg('Ошибка в файле: символ "'+stn+'", находящийся в строке '+IntToStr(k)+', позиция '+IntToStr(i)+' некорректен!' ,mtError, [mbOk],0);
                                Exit;
                            End;
                        If stn = ';'
```

```
                Then Begin A[j]:= StrToFloat(stg);
                        Break;
                    End;
                If stn <> ' '
                Then stg:= stg+stn
                Else Begin A[j]:= StrToFloat(stg);
                        stg:= '';
                        Inc(j);
                    End;
                Inc(i);
            End;
            str:= 'Входные данные: '+FloatToStr(A[1])+'+('+FloatToStr(A[2])+')i * '+FloatToStr(A[3])+'+('+FloatToStr(A[4])+')i; Выходные данные: ';
              {Случай сопряженных чисел}
            If (A[1]=A[3]) and (A[2]=-A[4])
            Then Begin X:=A[1]*A[1]+A[2]*A[2];
                    str:= str+FloatToStr(X);
                End
            Else Begin X:=(A[1]*A[3])-(A[2]*A[4]);
                    Y:=(A[1]*A[4])+(A[2]*A[3]);
                    str:= str+FloatToStr(X)+'+('+FloatToStr(Y)+'i)';
                end;
            WriteLn(F2,str);
        End;
    CloseFile(F1); CloseFile(F2);
    Form4.Close;
end;
```

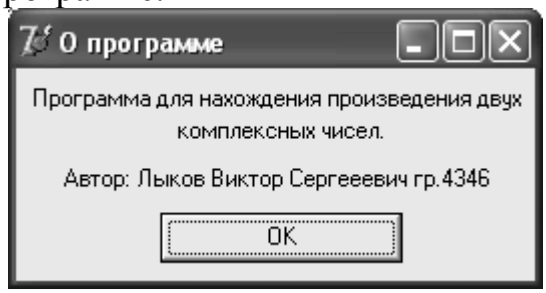
Приложение Д

Компоновка форм

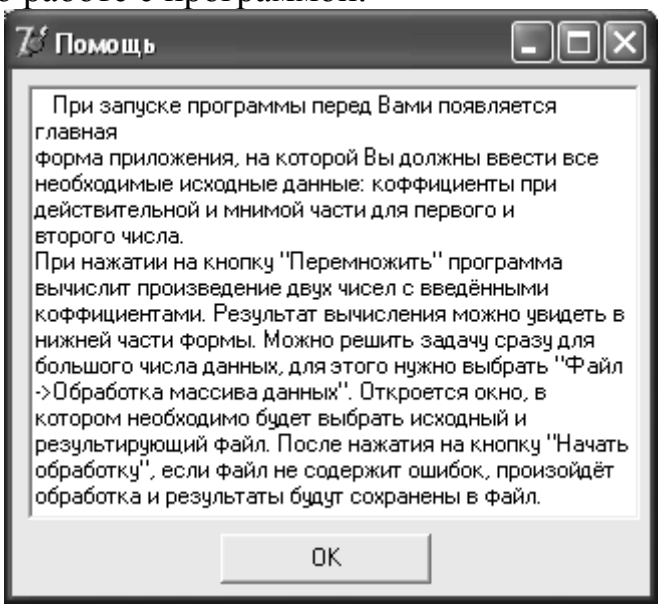
1. Окно основной формы интерфейса.



2. Окно информации о программе.



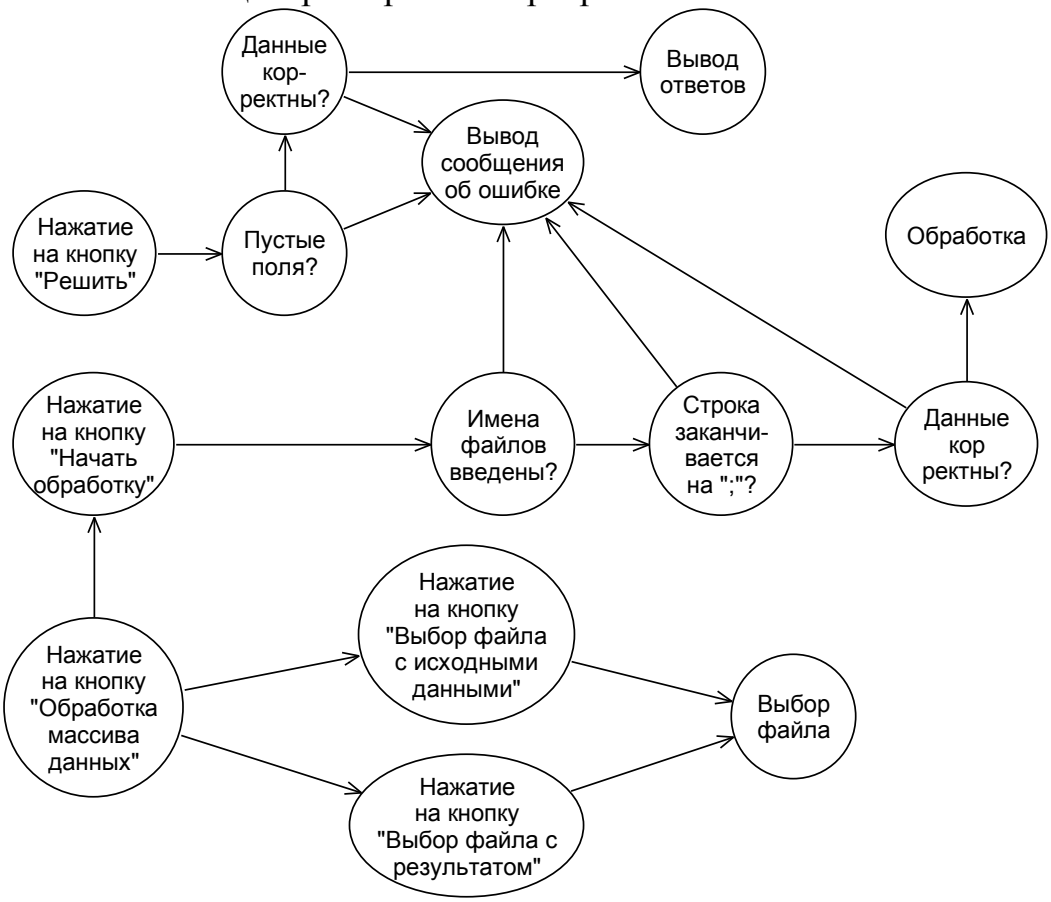
3. Окно пояснения по работе с программой.



Приложение Е

Разработка сценариев и реакций на события согласно сценариям.

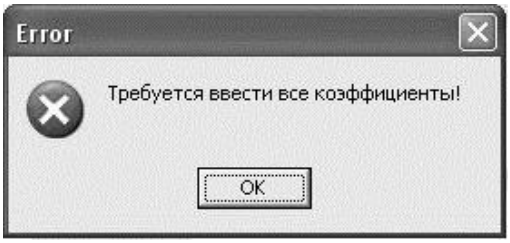
1) Графы возможных сценариев работы программы.



2) Тексты процедур обработки событий.

- Проверка на пустые поля

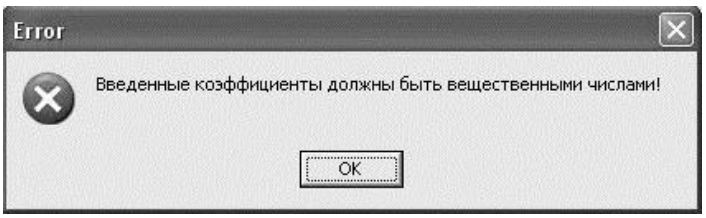
```
If (Edit1.Text = '') Or (Edit2.Text = '') Or (Edit3.Text = '') Or (Edit4.Text = '')
Then Begin MessageDlg('Требуется ввести все коэффициенты!',mtError,[mbOk],0);
        Exit;
    End;
```



- Проверка на ошибки ввода

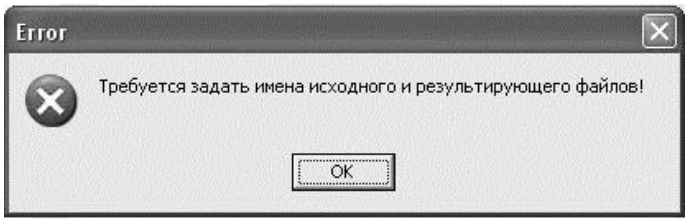
```
st:= Edit1.Text+Edit2.Text+Edit3.Text+Edit4.Text;
For i:= 1 To Length(st)
Do If Not (st[i] In ['0'..'9','-','+',',','E','e'])
    Then Begin MessageDlg('Введенные коэффициенты должны быть вещественными числами!',mtError,[mbOk],0);
```

```
                Exit;
            End;
```



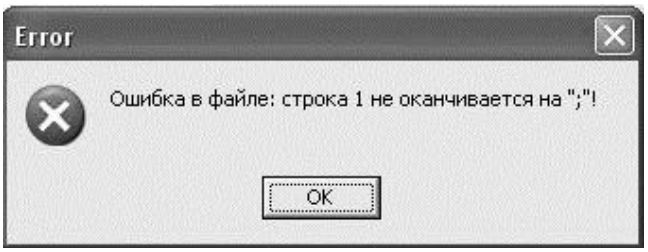
- Проверка на пустые имена файлов

```
    If (F1OK = False) Or (F2OK = False)
    Then Begin MessageDlg('Требуется задать имена исходного и результирующего файлов!',mtError,[mbOk],0);
                Exit;
            End;
```



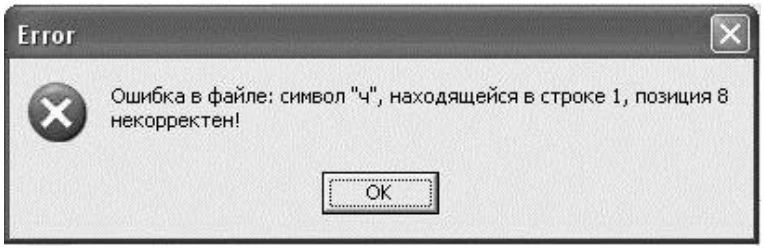
- Проверка на ошибку в файле: строка не оканчивается на «;»

```
    If st[Length(st)] <> ';'
    Then Begin MessageDlg('Ошибка в файле: строка '+IntToStr(k)+' не оканчивается на ";"!',mtError,[mbOk],0);
                Exit;
            End;
```



- Проверка на некорректные символы в файле обработки

```
    stn:= st[i];
    If Not (stn[1] In ['0'..'9','-','+',',',';',' ','E','e'])
    Then Begin MessageDlg('Ошибка в файле: символ "'+stn+'", находящийся в строке  '+IntToStr(k)+', позиция '+IntToStr(i)+' некорректен!',mtError,[mbOk],0);
                Exit;
            End;
```



3) Тестовые примеры

а) обычный случай

Входные данные:

a = 4

b = 3

c = 2

d = 5

Выходные данные:

X = -7

Y = 26

-7+26i

б) сопряжённые числа

Входные данные:

a = 4

b = 3

c = 4

d = -3

Выходные данные:

X=25

Y=0

25