Министерство образования и науки Российской Федерации

ФГБОУ ВО «СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Лесосибирский филиал

Кафедра информационных и технических систем

**Отладка и тестирование объектно-ориентированных программ**

Курсовая работа по дисциплине «Объектно-ориентированное программирование»

(ИТС 000000.010 ПЗ)

Руководитель:

\_\_\_\_\_\_\_\_И.А. Петрова

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка, дата)

Разработал:

студент группы 63-1

\_\_\_\_\_\_\_\_ А.О. Шадрин

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

Лесосибирск, 2016

Задание на курсовуюработу

Пояснительная записка курсовой работы должна содержать теоретическую часть по теме «Отладка и тестирование объектно – ориентированных программ», а также практическую часть, которая включает в себя описание следующей программы:«Электронный учебник по ООП».

Реферат

Курсовая работа представлена в виде пояснительной записки. Пояснительная записка излагается на 23 страницах машинописного текста. В ней представлены 18 рисунков.

Цель выполнения курсовой работы состоит в изучении самостоятельном поиске и систематизации информации по теме «Отладка и тестирование объектно – ориентированных программ», а также выполнение практического задания – создание приложения «Электронный учебник по ООП».

Содержание

[Введение 5](#_Toc454229869)

[1 Понятия отладки и тестирования объектно-ориентированных программ 6](#_Toc454229870)

[2 Цели и задачи тестирования программного обеспечения 8](#_Toc454229871)

[2.1 Цели тестирования: 8](#_Toc454229872)

[2.2 Задачи тестирования: 8](#_Toc454229873)

[3 Методы отладки программного обеспечения объектно-ориентированных программ 9](#_Toc454229874)

[3.1 Метод индукции. 9](#_Toc454229875)

[3.2 Метод дедукции. 10](#_Toc454229876)

[4 Этапы отладки и тестирования объектно-ориентированных программ 11](#_Toc454229877)

[4.1 Принципы тестирования. 11](#_Toc454229878)

[4.2 Этапы тестирования: 11](#_Toc454229879)

[4.3 Стратегии тестирования: 12](#_Toc454229880)

[5 Ошибка и виды ошибок 13](#_Toc454229881)

[6 Создание программы «Электронный учебник» 14](#_Toc454229884)

[6.1 Этап проектирования 14](#_Toc454229885)

[6.2 Разработка программы 16](#_Toc454229886)

[Заключение 22](#_Toc454229887)

[Библиографический список 23](#_Toc454229888)

Введение

Объектно-ориентированное программирование – это вид программирования, в котором в качестве элементов программы используются совокупности объектов, каждый из которых является экземпляром определённого класса, где классы образуют иерархию наследования.

Одним из языков объектно-ориентированного программирования является Delphi, основной область использования которого является – написание прикладного программного обеспечения.

Основные этапы жизненного цикла программного обеспечения:

1. Анализ и постановка задач.
2. Проектирование.
3. Кодирование.
4. Тестирование и отладка.
5. Документирование.
6. Эксплуатация, поддержка.

В данной курсовой работе будут описаны особенности одного из основных этапов жизненного цикла программного обеспечения - отладки и тестирования объектно-ориентированных программ.

Этап отладки и тестирования подразумевает выполнение комплексного тестирования написанной программы и объектов с целью выявления иустранения ошибок.

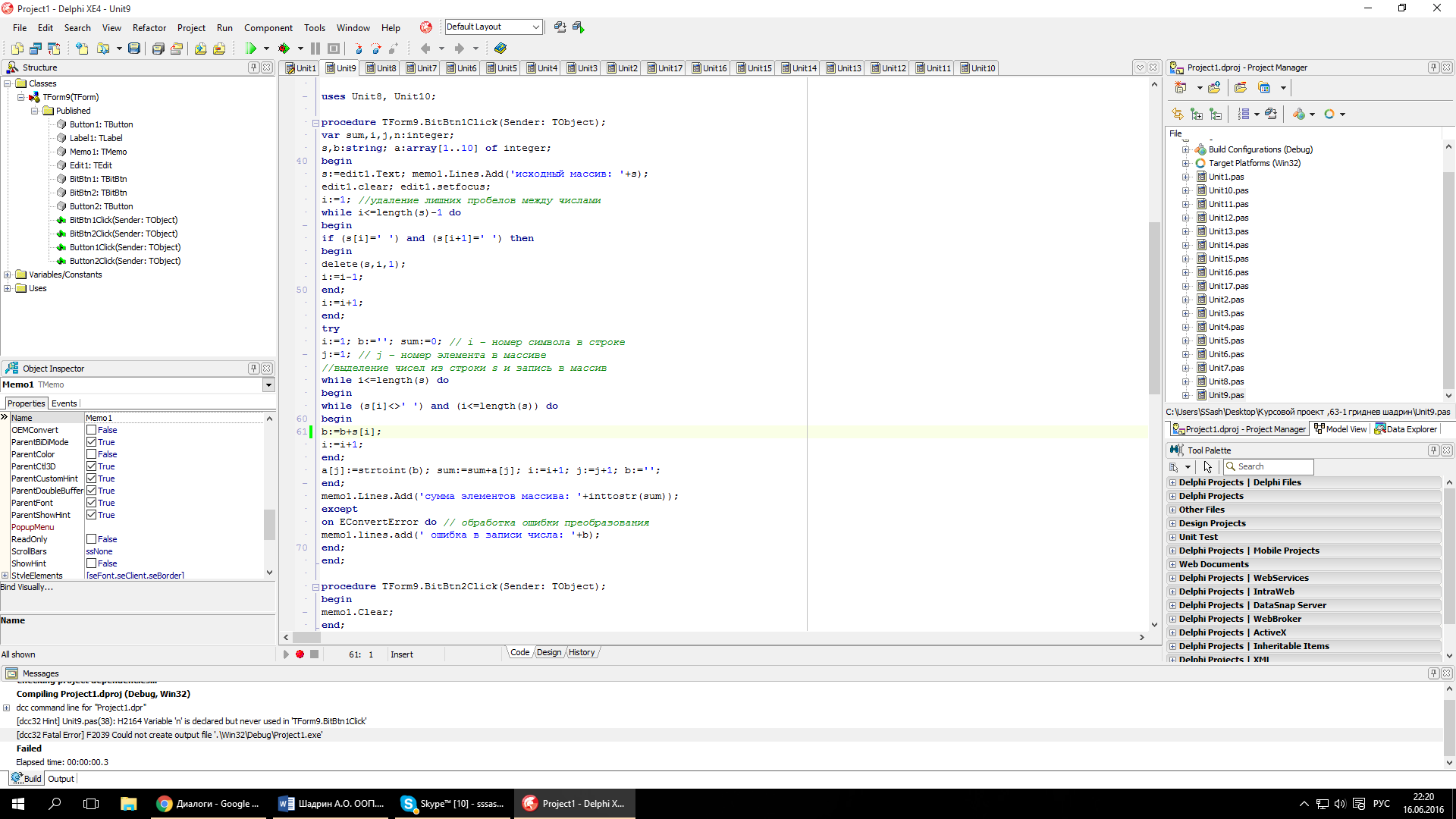


Рисунок 1 – Сообщение об ошибке в программе DelphiXE4

1 Понятия отладки и тестирования объектно-ориентированных программ

Изм.

Лист

№ докум.

Подпись

Дата

Лист

*6*

ИТС.000000.010 ПЗ

Отладка программного обеспечения – этап разработки программного обеспечения, в котором производится тестирование программы, а также поиск и устранение ошибок программы.

Тестирование программного обеспечения – этап разработки программного обеспечения, в котором производятся испытания программного продукта с целью выявить неправильную работу программы, либо убедиться в отсутствии проблем с программным продуктом.

Функциональные виды тестирования:

1. Функциональное тестирование -рассматривает заранее указанное поведение и основывается на анализе спецификаций функциональности компонента или системы в целом.
2. Тестирование безопасности- это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным.
3. Тестирование взаимодействия– это функциональное тестирование, проверяющее способность приложения взаимодействовать с одним и более компонентами или системами и включающее в себя тестирование совместимости и интеграционное тестирование.

Нефункциональные виды тестирования:

1. Тестирование производительности - определение работы программного продукта под нагрузкой.
2. Тестирование установки направленно на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.
3. Тестирование удобства пользования - это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий.
4. Тестирование на отказ и восстановление - проверяет тестируемый продукт с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками программного обеспечения, отказами оборудования или проблемами связи.
5. Конфигурационное тестирование — специальный вид тестирования, направленный на проверку работы программного обеспечения при различных конфигурациях системы.

В отличие от других видов программирования, в объектно-ориентированном программировании тестируются не функции, процедуры или другие программные модули, а производится тестирование классов и связей между ними.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

*7*

ИТС.000000.010 ПЗ

Главные понятия объектно-ориентированного программирования:

1. Инкапсуляция – объединение данных и свойственных им процедур обработки в одном объекте.
2. Класс - это шаблон, на основе которого может быть создан конкретный программный объект, он описывает свойства и методы, определяющие поведение объектов этого класса. Каждый конкретный объект, имеющий структуру этого класса, называется экземпляром класса.
3. Наследование-предусматривает создание новых классов на базе существующих и позволяет классу потомку иметь (наследовать) все свойства класса - родителя.
4. Полиморфизм-означает, что рожденные объекты обладают информацией о том, какие методы они должны использовать в зависимости от того, в каком месте цепочки они находятся.
5. Модульность- свойство программ, при котором объекты заключают в себе полное определение их характеристик, никакие определения методов и свойств не должны располагаться вне его, это делает возможным свободное копирование и внедрение одного объекта в другие.

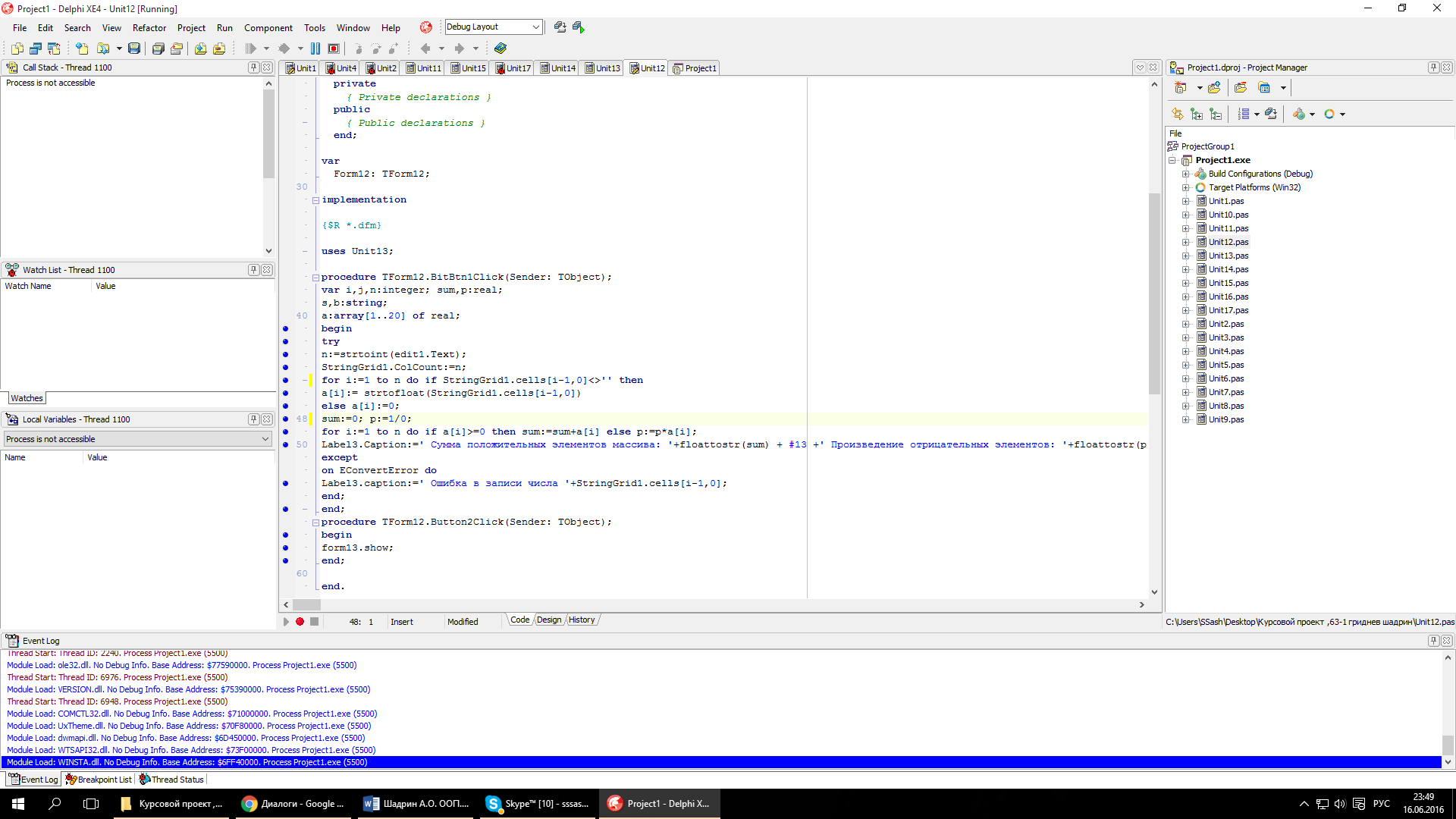


Рисунок 2 – Трассировка программы в DelphiXE4

2 Цели и задачи тестирования программного обеспечения

Изм.

Лист

№ докум.

Подпись

Дата

Лист

*8*

ИТС.000000.010 ПЗ

2.1 Цели тестирования:

1. Повысить вероятность того, что приложение, предназначенное для тестирования, будет работать правильно при любых обстоятельствах.
2. Повысить вероятность того, что приложение, предназначенное для тестирования, будет соответствовать всем описанным требованиям.
3. Провести полное тестирование приложения за короткий срок.

2.2 Задачи тестирования:

1. Проверить, что система работает в соответствии с определенными временами отклика клиента и сервера.
2. Проверить, что наиболее критические последовательности действий с системой конечного пользователя выполняются верно.
3. Проверить работу пользовательских интерфейсов
4. Проверить, что изменения в базах данных не оказывают неблагоприятного влияния на существующие программные модули.
5. При проектировании тестов свести к минимуму переработку тестов при возможных изменениях приложения.
6. Использовать инструменты автоматизированного тестирования там, где это целесообразно.
7. Проводить тестирование таким образом, чтобы не только обнаруживать, но и предупреждать дефекты.
8. При проектировании автоматизированных тестов использовать стандарты разработки таким образом, чтобы создать многократно используемые и сопровождаемые скрипты.

3 Методы отладки программного обеспечения объектно-ориентированных программ

3.1 Метод индукции.

Метод индукции - подразумевает анализ программы от частного к общему. Просматриваются ошибки и определяются данные, которые имеют к ней хоть какое-то отношение. Далее исключаются маловероятные гипотезы, пока не остается одна, которую можно уточнить и доказать.

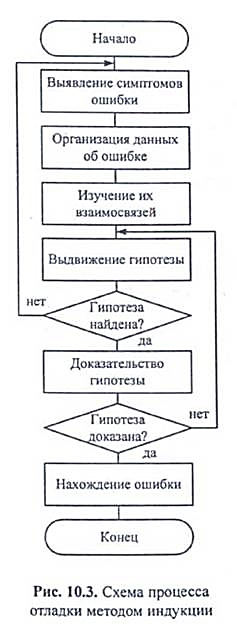


Рисунок 3 – Схема процесса отладки методом индукции

3.2 Метод дедукции.

Метод дедукции–подразумевает анализ программы от общего к частному. Выдвигается гипотеза, которая может объяснить ошибку. Далее при помощи тестов эта гипотеза проверяется и доказывается.

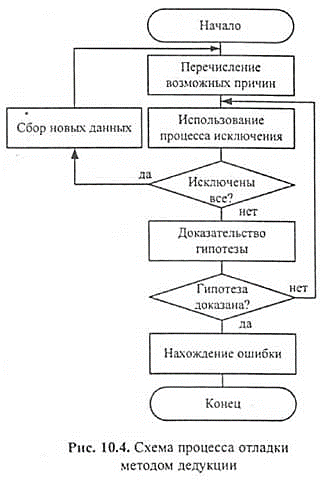


Рисунок 4 – Схема процесса отладки методом дедукции

4 Этапы отладки и тестирования объектно-ориентированных программ

4.1 Принципы тестирования.

1. Просчитать вручную программу от ввода исходных данных до выходных данных. Такие данные будут считаться эталонными.
2. При прогоне программы по тестовым данным, полученные результаты необходимо сравнивать с эталонными.
3. При проведении тестирования нужно учитывать не только правильные, но и неверные исходные данные.
4. Провести проверку на нежелательные исходные данные, чтобы в дальнейшем ограничить ввод некоторых исходных данных, например, деление на ноль, считывание данных из несуществующего файла и т.д.

4.2 Этапы тестирования:

1. Сравнение программы со схемой алгоритма.
2. Визуальный контроль программы на экране дисплея или визуальное изучение распечатки программы и сравнение ее с оригиналом на программном бланке. Первые два этапа тестирования способны устранить больше количество ошибок, как синтаксических (что не так важно), так и семантических (что очень важно, так как позволяет исключить их трудоемкий поиск в процессе дальнейшей отладки).
3. Трансляция программы на машинных язык. На этом этапе выявляются синтаксические ошибки. Компиляторы с языков Си, Паскаль выдают диагностическое сообщение о синтаксических ошибках в листинге программы (листингом называется выходной документ транслятора, сопровождающий оттранслированную программу на машинном языке – объектный модуль).
4. Редактирование внешних связей и компоновка программы. На этапе редактирования внешних связей программных модуле программа-редактор внешних связей, или компоновщик задач, обнаруживает такие синтаксические ошибки, как несоответствие числа параметров в описании подпрограммы и обращении к ней, вызов несуществующей стандартной программы.
5. Выполнение программы. После устранения обнаруженных транслятором и редактором внешних связей синтаксических ошибок переходят к следующему этапу – выполнению программы на ЭВМ на машинном языке: программа загружается в оперативную память, в соответствие с программой вводятся исходные данные и начинается счет. Проявление ошибки в процессе вода исходных данных или в процессе счета приводит к прерыванию счета и выдаче диагностического сообщения рабочей программы. Проявление ошибки дает повод для выполнения отладочных действий; отсутствие же сообщений об ошибках не означает их отсутствия в программе. План тестирования включает при этом проверку правильности полученных результатов для каких-либо допустимых значений исходных данных.
6. Тестирование программы. Если программа выполняется успешно, желательно завершить ее испытания тестированием при задании исходных данных, принимающих предельные для программы значения, а также выходящие за допустимые пределы значения на входе.

4.3 Стратегии тестирования:

1. Тестирование программы как «черного ящика». В данной стратегии известно только о том, что делает программа, но её внутренняя структура остаётся в тайне. Задается набор входных данных, получаются результаты, которые сверяются с эталонными. При этом обнаружить все ошибки невозможно, так как возможны сложные структурные ошибки кода.
2. Тестирование программы как «белого ящика». Данный метод заключается в том, что перед составлением теста изучается логика программы, и её внутренняя структура. Тестирование будет считаться удачным, если программа проходит проверку по всем направлениям.

Необходимо уточнить, что тестирование программы как «черного ящика» не требует навыков программирования и выполняется в основном тестерами, в то время как «белый ящик» требует знания программирования и используется разработчиками программного обеспечения.

5 Ошибка и виды ошибок

Программная ошибка –это изъян в разработке программного продукта, из-за которого происходят расхождения между ожидаемыми и полученными данными. Ошибка программы может возникнуть на любом из этапов проектирования программного продукта.

Основные виды ошибок в программировании:

1. Логические ошибки – это ошибки, вызванные недостатками разработанного алгоритма, то есть для её решения необходимо переработать алгоритм.
2. Синтаксические ошибки – это ошибки в записи конструкций языка программирования (чисел, переменных, функций, выражений, операторов, меток, подпрограмм).

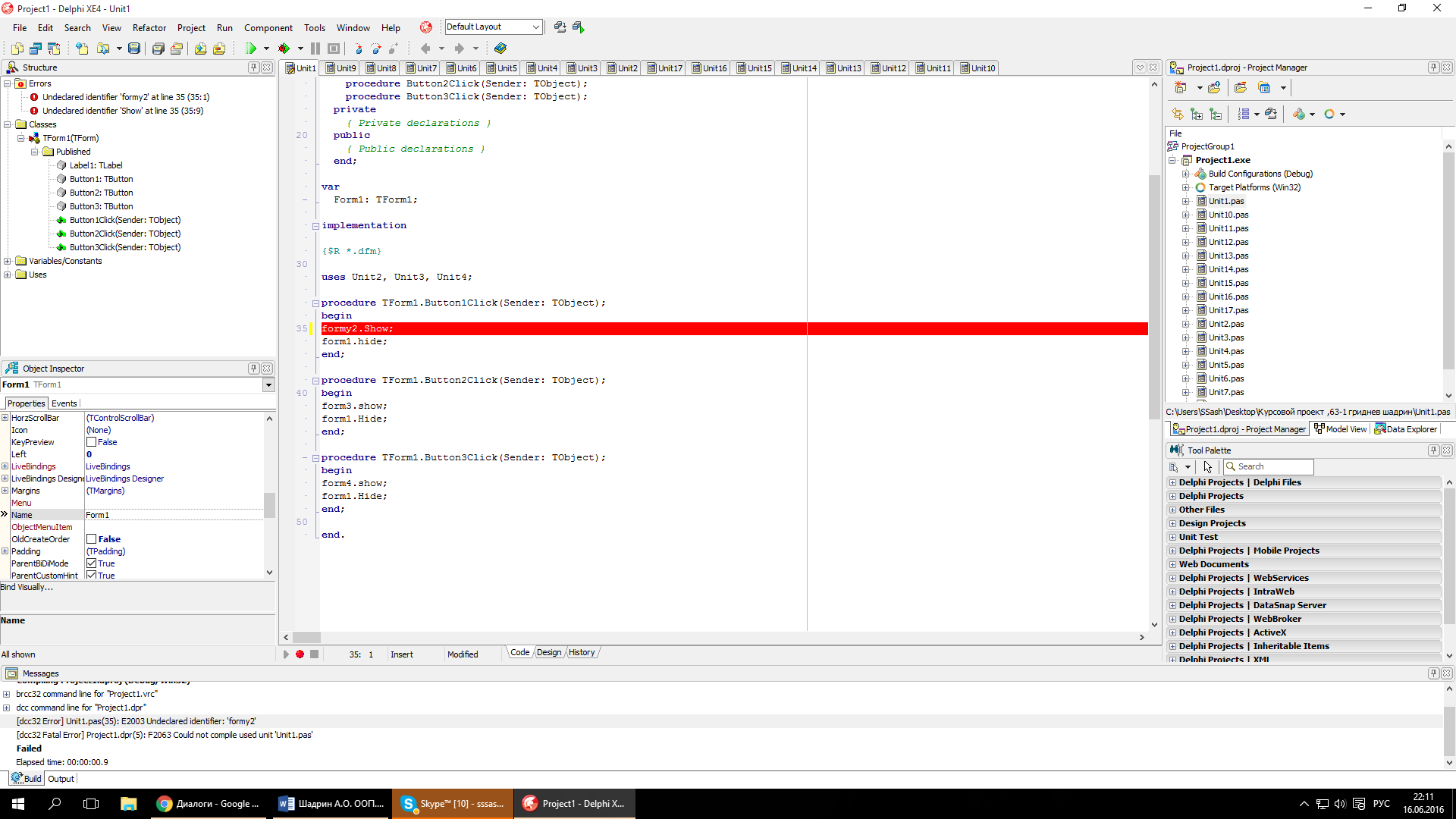


Рисунок 5 – Пример синтаксической ошибки в DelphiXE4

1. Семантические ошибки – это ошибки, связанные с неправильным содержанием действий и использованием недопустимых значений величин.

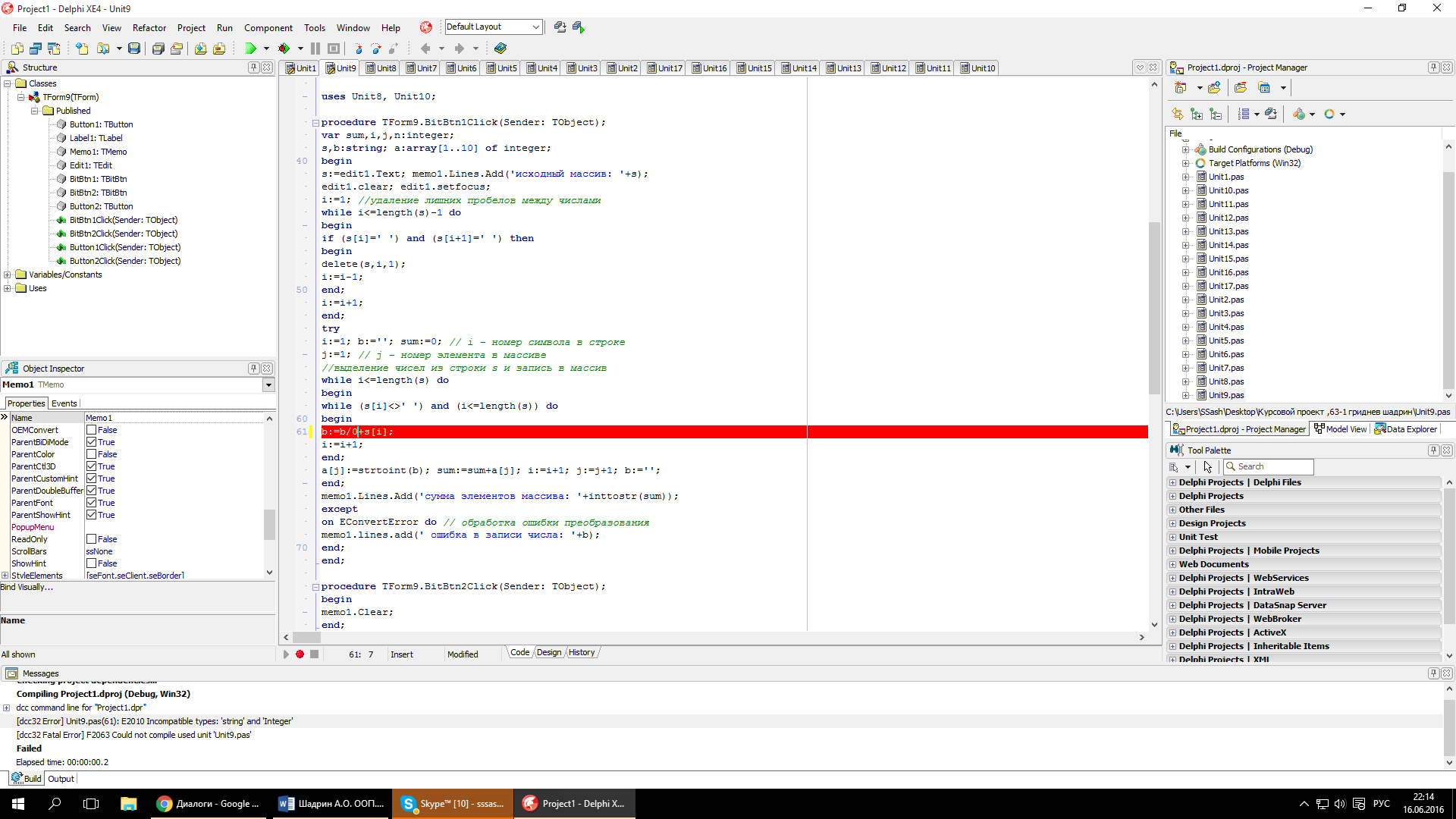


Рисунок 6 – Пример семантической ошибкив DelphiXE4

1. Ошибки ресурса – это ошибки, которые возникают, когда значение переменной превышает максимально допустимое значение, например, переполнение стека, превышение количества столбцов в табличном редакторе Grid и т.д.

6 Создание программы «Электронный учебник»

6.1 Этап проектирования

На первом этапе создания программы была спроектирована структура программы.

Данный электронный учебник будет содержать три основных раздела:

* одномерные массивы;
* многомерные массивы;
* динамические массивы.

Которые внутри будут поделены на разделы, содержащие:

* описание способов ввода данных в массив;
* примеры рабочих программ;
* задания для самостоятельного решения.

Также на этапе проектирования программы была составлена диаграмма IDEF0, предназначена для формализации процесса создания программы «Электронный учебник по ООП»:

1. Уровень A0-1

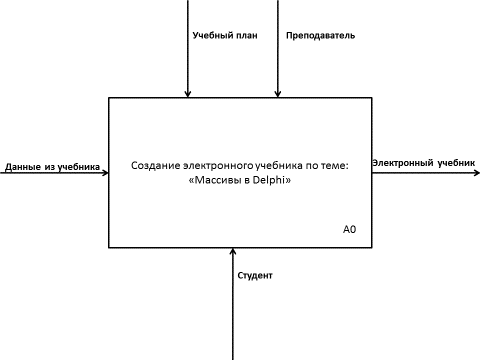


Рисунок 7 – Диаграмма потоков создания электронного учебника

1. Уровень A0

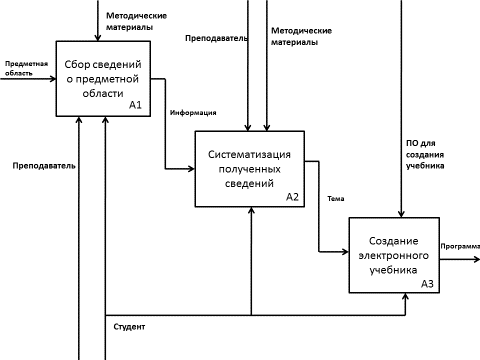


Рисунок 8 – Диаграмма процесса создания электронного учебника

1. Уровень A3

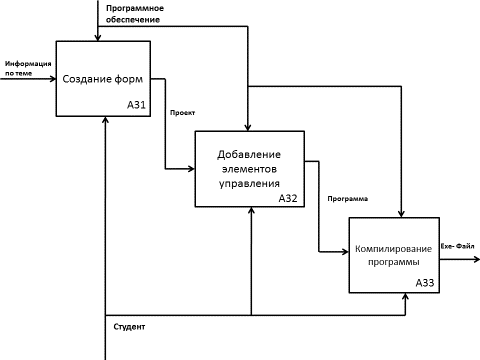


Рисунок 9 – Диаграмма процесса создания учебника в среде Delphi

6.2 Разработка программы

1. Разработка программы начинается с создания проекта в среде Delphi XE2.

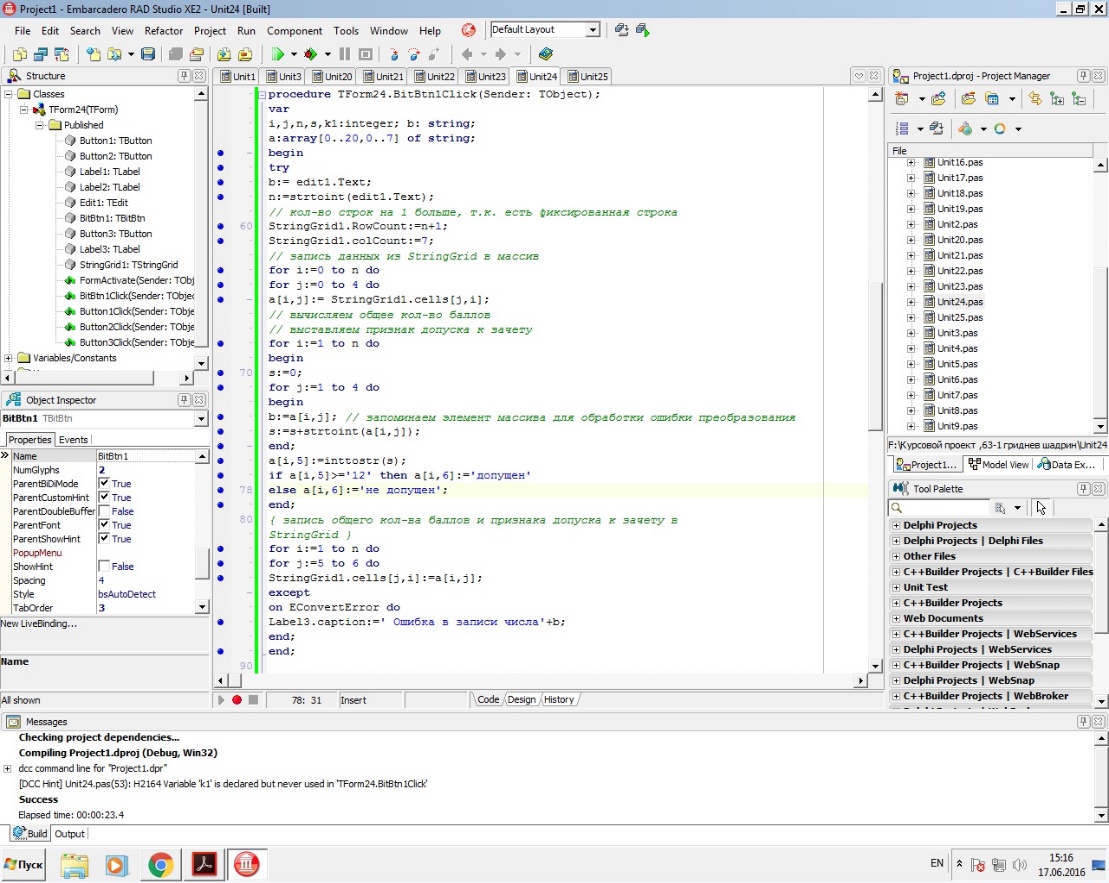


Рисунок 10 – Среда Delphi XE2

1. Далее создается нужное количество форм в проекте. По желанию можно их визуально настроить (Например, изменить заголовки форм).

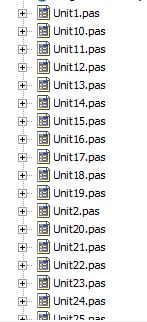


Рисунок 11 – Список форм проекта

1. Из панели Standart добавляем элементы Label и Button, путем перетаскивания их мышью на форму.

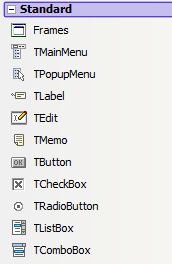


Рисунок 12 – Стандартные элементы

1. Изменяем их параметры Caption.

Таблица 1 – Таблица внешних спецификаций

|  |  |  |  |
| --- | --- | --- | --- |
| Выделенный объект | Вкладка окна Object Inspector | Имя свойства/ события | Действие |
| Form1 | Properties | Caption | Установка имени формы «Массивы в Delphi» |
| Label1 | Properties | Caption | Ввести «Электронный учебник по теме «Массивы в Delphi» |
| Button1 | Properties | Caption | Установка имени кнопки: «Одномерные массивы» |
| Button2 | Properties | Caption | Установка имени кнопки: «Двумерные массивы» |
| Button3 | Properties | Caption | Установка имени кнопки: «Динамические массивы» |

1. Для кнопок прописываются события Onclick. На открытие нужной формы и скрытие предыдущей. Таким образом мы реализовали переходы по нашим формам

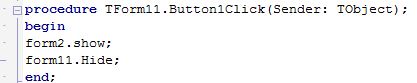


Рисунок 13 – Процедура перехода по формам

1. Теоретический материал, взятый из методических указаний, будем вставлять с помощью многострочного редактора Memo.



Рисунок 14 – Пример вывода текста

1. Для этого нужно дважды кликнуть по параметру Lines. Откроется меню ввода текста.

Таблица 2 – Таблица внешних спецификаций

|  |  |  |  |
| --- | --- | --- | --- |
| Выделенный объект | Вкладка окна Object Inspector | Имя свойства/ события | Действие |
| Form1 | Properties | Caption | Установка имени формы «Массивы в Delphi» |
| Label1 | Properties | Caption | Ввести «Ввод одномерного массива с помощью таблицы Stringgrid» |
| Button1 | Events | OnClick | Реализация перехода по формам «Form2.show  Form1.hide» |
| Button2 | Events | OnClick | Реализация перехода по формам «Form3.show  Form2.hide» |
| Memo1 | Properties | Lines | Ввод данных в многострочный редактор |
| Properties | Scrollbar | Установка параметра ssvertical |

1. Теперь нужно добавить полосу прокрутки для удобства. Для этого в параметре Scrollbars устанавливаем значение ssvertical.

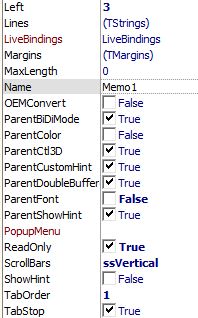


Рисунок 15 – Параметры элемента Memo

1. Добавляем кнопку перехода на пример.

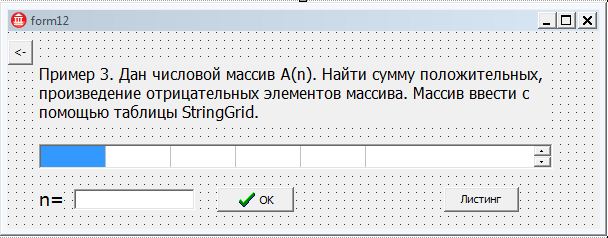


Рисунок 16- Форма с рабочим примером

Таблица 3 – Таблица внешних спецификаций

|  |  |  |  |
| --- | --- | --- | --- |
| Выделенный объект | Вкладка окна Object Inspector | Имя свойства/ события | Действие |
| Form1 | Properties | Caption | Установка имени формы «Массивы в Delphi» |
| Label1 | Properties | Caption | Ввести «Ввод одномерного массива с помощью таблицы Stringgrid» |
| Label2 | Properties | Caption | Ввести «N=» |
| Label3 | Properties | Visible | Установить параметр Hide |
| Button1 | Events | OnClick | Реализация перехода по формам «Form2.show  Form1.hide» |
| Button2 | Events | OnClick | Реализация процедуры выполнения программы |
| Button3 | Events | OnClick | Реализация перехода по формам «Form3.show Form1.hide» |
| Stringgrid1 | Properties | Lines | Ввод данных в многострочный редактор |

1. В форме с примером делаем ссылку на листинг нашей программы, которая отображается с помощью многострочного редактора Memo.

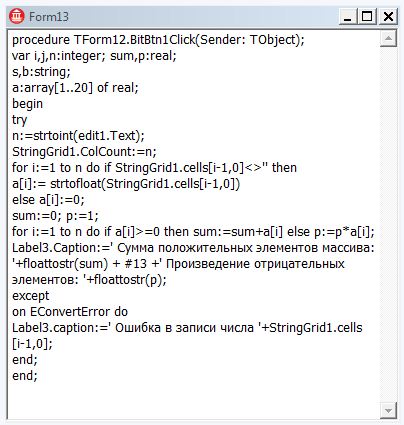


Рисунок 17 – Листинг программы из примера

Остальные пункты меню нужно будет создать подобным образом. Их структура не будет сильно отличатся, а только будет иметь разное наполнение.

1. Далее компилируем наш проект.

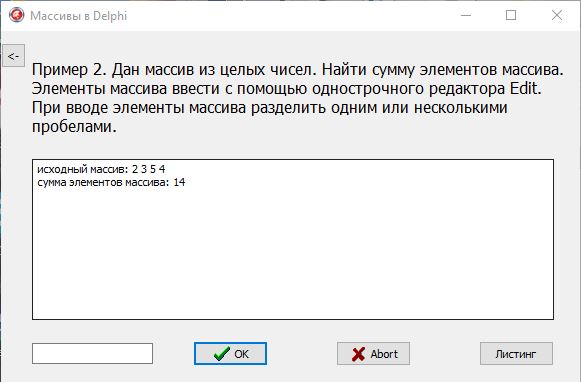


Рисунок 18 - Скомпилированная программа.

1. Последним шагом является тестирование и отладка программы, путем запуска программы из EXE-Файла. Ошибок в ходе тестирования не было найдено, а значит программа полностью готова к работе.

Заключение

Итак, тестирование и отладка является одним из главных этапов разработки программного обеспечения, так как программный продукт даже с мелкими ошибками будет вызывать как минимум дискомфорт со стороны пользователя. Нужно отметить, что во многих организациях на тестирование и отладку их программных продуктов уходит большой процент средств, выделенных на разработку программы, так как часто после релиза программного продукта находятся как мелкие, так и большие ошибки, которые нужно исправлять и выпускать обновления для программ.

Библиографический список

1. М.А. Плаксин, Тестирование и отладка программ для профессионалов будущих и настоящих. – М.: БИНОМ, 2013 г. – 168 с.
2. И.Ш. Хабибуллин, Программирование на языке высокого уровня С/C++. – СПб.: БХВ-Петербург, 2006г. –512 с.
3. Объектно-ориентированное программирование (ООП-1). [Электронный ресурс] / Режим доступа: http://bourabai.ru/alg/oop.htm
4. Технологии программирования. Объектно-ориентированное программирование. [Электронный ресурс] / Режим доступа: http://school-collection.lyceum62.ru/ecor/storage/588de0a9-a9bd-40e4-8497-2ab7f25a35ed/%5bINF10\_04\_09\_TI\_1C%5d.html#5
5. Введение в проектирование ИС. Предпосылки возникновения объектно-ориентированного подхода. [Электронный ресурс] / Режим доступа: http://listenbook.narod.ru/sucess\_full\_pis.html
6. Тестирование программного обеспечения. [Электронный ресурс] / Режим доступа: https://en.wikipedia.org/wiki/Software\_testing
7. Отладка программы. [Электронный ресурс] / Режим доступа: https://en.wikipedia.org/wiki/Debugging
8. Методы отладки программного обеспечения. [Электронный ресурс] / Режим доступа: http://tehprog.ru/index.php\_page=lecture0113.html