## Тема 1. Модульная структура приложений и стандартные модули в Visual Basic

1.1. Структура приложений и программ в VB

[1.2. Стандартные модули и их использование](file:///F:\2015УчП-Раз-5-Инф-Шакин-П+Л\КНИГА-4\Тема-05-00-Огл-Пред-Введ%20ПРОВ-23_11_2014.docx#_5.1.1._Стандартные_)

[1.3. Создание проектов с несколькими формами](file:///F:\2015УчП-Раз-5-Инф-Шакин-П+Л\КНИГА-4\Тема-05-00-Огл-Пред-Введ%20ПРОВ-23_11_2014.docx#_5.1.2._Создание_проектов)

[1.4. Задачи для самостоятельного решения](file:///F:\2015УчП-Раз-5-Инф-Шакин-П+Л\КНИГА-4\Тема-05-00-Огл-Пред-Введ%20ПРОВ-23_11_2014.docx#_5.1.4.__Задачи)

[1.5. Лабораторная работа](file:///F:\2015УчП-Раз-5-Инф-Шакин-П+Л\КНИГА-4\Тема-05-00-Огл-Пред-Введ%20ПРОВ-23_11_2014.docx#_5.1.5.__Тестовые) «Модульная структура приложений и стандартные

модули в Visual Basic»

1.6. Контрольные вопросы

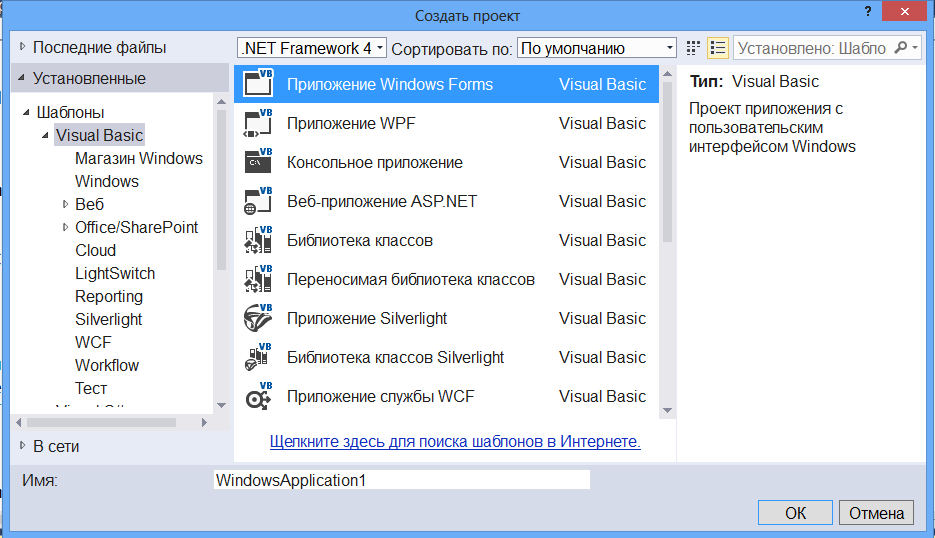
### 1.1. Структура приложений и программ в Visual Basic

Среда разработки VS .NET предоставляет мощные и удобные средства написания, корректировки, отладки и запуска приложений, использующих .NET-совместимые языки (В.3). Все программы независимо от языка **используют одни и те же базовые классы библиотеки FCL каркаса .NET Framework**.Приложение в процессе разработки называется ***проектом***. Проект объединяет все необходимое для создания приложения: файлы, папки, ссылки и прочие ресурсы. Среда VS .NET позволяет создавать проекты различных типов (шаблонов). Кратко охарактеризуем три основных шаблона проектов:

* ***Приложение Windows Forms*** – основной вид проектов языка VB, использует элементы интерфейса Windows, включая формы, кнопки, флажки и т.п. Таким образом, интерфейс пользователя таких приложений строится в визуальном стиле на основе форм. Приставка **Visual** в названии языка VB во многом определяется возможностями этого типа проектов.
* ***Консольное приложение*** – выполняет вывод «на консоль», т.е. в окно командного процессора, это весьма популярный у разработчиков тип проектов, используемый для внутренних целей в ходе разработки, когда для понимания тех или иных возможностей языка VB достаточно весьма простого интерфейса – ввода и вывода данных на консоль.
* ***Библиотека классов*** – проект, позволяющий построить DLL – динамически подключаемую библиотеку классов, которая объединяет классы, предназначенные для использования в других приложениях. Дело в том, что всегда полезно отделять содержательную часть приложения от ее интерфейса. Классы, определяющие содержание приложения, могут собираться в DLL, которая затем будет подключаться к проекту, описывающему интерфейс приложения. Такой подход в наибольшей степени соответствует **повторному использованию**. Один раз созданные классы, собранные в DLL, могут использоваться в разных приложениях.

Как известно, для создания любого проекта в VS используется диалоговое окно **Создание проекта** (рис. 1.1-1). Для его открытия необходимо выполнить одно из следующих действий:

* выбрать ссылку Создать проект в окне Начальная страница;
* выполнить команду Создать проект из элемента главного меню Файл (File);
* нажать кнопку Создать проект, расположенную на стандартной панели инструментов.



*Рис. 1.1-1. Диалоговое окно* ***Создать проект***

Дадим краткие определения основных понятий, которые будут использоваться при рассмотрении примеров.

***Класс (*Class*)*** – это центральное понятие объектно-ориентированного программирования и языка VB. Разработчик проектов на VB может использовать стандартные классы из библиотеки **FCL** (Framework Class Library, библиотеки классов платформы .NET) и создавать собственные классы.

Причем класс может выполнять две различные роли.

* Во-первых***, Класс*** – это ***модуль*** – архитектурная единица построения проекта по модульному принципу. Справиться со сложностью большого проекта можно только путем деления его на модули, – сравнительно небольшие единицы, допускающие независимую разработку и последующее объединение в большую программную систему.
* Во-вторых***, Класс*** – это ***тип данных***. Тип данных – это семантическая единица, описывающая свойства и поведение множества объектов, являющихся экземплярами класса. Подробнее классы как типы и как объекты будем рассматривать в **Темах 3, 4**.
* Хороший стиль программирования требует, чтобы каждый класс сохранялся в отдельном файле, имя которого совпадало бы с именем класса. Это требование стиля, которое на практике может и не выдерживаться. В наших примерах будем стараться выдерживать этот стиль.
* ***Пространство имен* (Namespace)** – это виртуальная оболочка, содержащая множество классов, объединенных, как правило, общей тематикой или группой разработчиков. Все классы библиотеки .NET, а также все классы, которые создает разработчик приложений в среде VS, имеют одного общего предка – класс **Object** и организованы в единую иерархическую структуру. Внутри нее классы логически сгруппированы в так называемые пространства имен, которые служат для упорядочивания имен классов и предотвращения конфликтов имен (в разных пространствах имена могут совпадать). Собственные имена классов внутри пространства имен должны быть уникальны. В разных пространствах могут существовать классы с одинаковыми именами. Полное или уточненное имя класса состоит из уникального имени пространства имен и собственного имени класса. В пространстве имен могут находиться как классы, так и пространства имен. Пространства имен облегчают независимую разработку проекта большим коллективом разработчиков, каждая группа которого работает в своем пространстве имен. Центральным пространством имен библиотеки FCL является пространство **System,** содержащее другие пространства и классы, имеющие широкое употребление в различных проектах.
* ***Проект (Project)*** – это единица компиляции. Результатом компиляции проекта является **сборка**. Каждый проект содержит одно или несколько пространств имен. На начальном этапе создания проекта по заданному типу проекта (шаблону) автоматически строится каркас проекта, состоящий из классов, являющихся наследниками классов, входящих в состав библиотеки FCL. Так, если разработчик указывает, что он хочет построить проект типа ***Приложение Windows Forms***, то в состав каркаса проекта по умолчанию войдет класс **Form1** – наследник библиотечного класса **Form**. Разработчик проекта дополнит созданную форму элементами управления – объектами соответствующих классов, тем самым расширив возможности класса, построенного по умолчанию.
* Каждый проект содержит всю информацию, необходимую для построения сборки. В проект входят все файлы с классами, построенные автоматически в момент создания проекта, и файлы с классами, созданные разработчиком проекта. Помимо этого, проект содержит ссылки на пространства имен из библиотеки FCL, содержащие классы, используемые в ходе вычислений. Проект должен содержит ссылки на все подключаемые к проекту DLL, COM-объекты, другие проекты. В проект входят установки и ресурсы, требуемые для выполнения загрузочного модуля. Частью проекта является файл, содержащий описание сборки.
* В зависимости от выбранного типа проект может быть выполняемым(\*.exe) или невыполняемым(\*.dll). К выполняемым проектам относятся консольные и Windows Forms проекты. При построении каркаса выполняемого проекта в него включается класс, содержащий статическую процедуру с именем **Main()**.
* К невыполняемым проектам относятся библиотечные проекты типа DLL (библиотеки классов). В результате компиляции такого проекта в сборку войдет файл с расширением \*.dll. Такие проекты (сборки) непосредственно не могут быть выполнены на компьютере. Они присоединяются к выполняемым сборкам, откуда и вызываются методы классов, размещенных в невыполняемом проекте (DLL).
* ***Сборка (Assembly)*** – результат компиляции проекта. Сборка представляет собой коллекцию из одного или нескольких файлов, помеченных номером версии. Каждая сборка разворачивается на компьютере как единое целое. Разработчик проектов работает с проектами, **CLR** (Common Language Runtime – общеязыковая среда выполнения) работает со сборками. Сборка позволяет решать вопросы безопасности, так как содержит описание требуемых ей ресурсов и права доступа к элементам сборки. Каждая сборка содержит манифест, содержащий полное описание сборки, ее элементов, требуемые ресурсы, ссылки на другие сборки, исполняемые файлы. Благодаря этому описанию **CLR** не требуется никакой дополнительной информации для развертывания сборки, трансляции промежуточного кода и его выполнения. Манифест идентифицирует сборку, специфицирует файлы, требуемые для реализации сборки, специфицирует типы и ресурсы, составляющие сборку, задает зависимости, необходимые в период компиляции для связи с другими сборками, специфицирует множество разрешений, необходимых, чтобы сборка могла выполняться на компьютере.
* ***Решение.*** Каждый проект, создаваемый в VS, помещается в оболочку, называемую ***Решением***. Решение может содержать несколько проектов, как правило, связанных общей темой (например, три проекта в **Примере 1.1-1**, помещены в одно ***Решение***). Заметим, что когда создается новый проект, то он может быть помещен в уже существующее ***Решение*** или может быть создано новое ***Решение*,** содержащее проект. Решения позволяют придать структуру множеству проектов, что особенно полезно, когда проектов много.

Итак, программа VB построена из стандартных блоков. **Решение** состоит из одного или нескольких проектов. **Проект,** в свою очередь, может содержать одну или несколько сборок. Каждая **сборка** компилируется из одного или нескольких исходных файлов. **Исходный файл** включает в себя определения и реализацию классов, структур, модулей и интерфейсов на языке программирования, и, в конечном счете, содержит весь программный код.

Рассмотрим на примере основные шагисоздания основных типов проектов и решения [7, 8].

Необходимо отметить, что ***Консольное приложение*** в дальнейшем изложении данного пособия не будет встречаться. Однако необходимость знать, как строится проект этого типа, связана с тем, что некоторые лекторы излагают примеры разделов на базе проектов типа ***Консольное приложение.***

**Пример 1.1-1. Создать три проекта, которые содержат – *консольный проект*, *Windows-проект* и *библиотеку классов,* которую будем подключать к консольному и Windows проектам. Проект библиотеки классов ClassLibraryMath должен реализовать класс ClassLibraryMath, аналог класса System.Math, только ограничимся лишь одним методом- вычислением одной функции sin(x).**

Напомним, что функцию **sin(x)**, как и другие математические функции, можно вычислить, используя разложение в ряд Тейлора:

.

Построим необходимые рекуррентные соотношения. Запишем соотношения для членов ряда Тейлора a0, ak, ak+1:

.

Вычислив *ak+1/ak*, получим требуемое соотношение:

1. ***Создание DLL* – *проекта Библиотека классов.***

Запустим VS, со стартовой страницы перейдем к созданию проекта и в качестве типа проекта укажем тип **Библиотека классов**. В открывшемся окне создания DLL, все поля заполнены значениями по умолчанию. Как правило, их следует переопределить.

В поле **Имя** задается имя строящейся DLL – **ClassLibraryMath**.

В поле **Расположение** указывается путь к папке проекта.

Обратите внимание и на другие установки, сделанные в этом окне – в верхнем окошке из списка возможных каркасов выбран каркас   
Framework .NET. Задав требуемые установки и щелкнув по кнопке **OK**, получим автоматически построенную заготовку проекта **DLL**, открытую в среде разработки проектов **VS**.

В окне **Обозреватель решений**  указано **Решение** с именем **ClassLibraryMath**, содержащее проект DLL с именем **ClassLibraryMath**. В папке **Свойства** проект содержит файл с описанием сборки – ее имя и другие характеристики. В папке **Ссылки** проект содержит ссылки на основные пространства имен библиотеки FCL, которые могут понадобиться в процессе работы DLL. Если содержимое этой папки не отображается, то необходимо щелкнуть по значку ***Показать все файлы*** в верхней части **Обозревателя решений.**

Всякая DLL содержит один или несколько классов, и сейчас для одного класса, которому по умолчанию дано имя **Class1**, заготовка построена. Класс этот, показанный в окне кода, пока что пуст – не содержит никаких элементов.

Построенный автоматически класс вложен в пространство имен, которое по умолчанию получило имя, совпадающее с именем проекта – **ClassLibraryMath**.

Изменим имя **Class1** на имя **ClassLibraryMath**.Это продиктовано правилом стиля – **имя класса** и **имя файла**, хранящего класс, **должны совпадать**. Переименование имени файла делается непосредственно в окне **Обозреватель решений** щелчком правой кнопки мыши по выделенному имени **Class1** и выбором пункта ***Переименовать*** в всплывающем контекстном меню.

Теперь допишем в этот класс статический (**Shared**) метод (**Function**), позволяющий вычислить функцию **sin(x)**. Подробно статические и нестатические методы класса будут рассмотрены в п. 2.2.3. Имя метода сделаем таким же, как и в классе **System.Math,** т.е. **sin(x)**. В результате программный код класса будет таким, как на рис. 1.1-2.

|  |
| --- |
| **Option Explicit On**  **Option Strict On**  **Public Class ClassLibraryMath**  *'Константы класса*  **Const EPS As Double = 0.000000001**  **Const** **OPT** **As Double** **= 2 \* Math.PI**  *'Статические методы класса*  **Shared Function Sin(ByVal x As Double) As Double**  **Dim a, res As Double**  **Dim k As Integer**  *'Оптимизация - приведение к интервалу*  **x = x** **Mod** **OPT**  *'Инициализация*  **a = x**  **res = 0**  **k = 0**  *'Основные вычисления*  **Do While (Math.Abs(a) > EPS)**  **res = res + a**  **a = a \* (-x \* x / ((2 \* k + 2) \* (2 \* k + 3)))**  **k = k + 1**  **Loop**  **Return res**  **End Function**  **End Class** |

*Рис.1.1-2. Программный код класса со статическим методом  
вычисления функции* ***Sin(x)*** *разложением в ряд*

Поставленная цель достигнута, – создан **программный код**, содержащий класс, метод которого позволяет вычислять по заданному аргументу **x** значениефункции **sin(x)**. При его построении использованы два важных подхода для оптимизации. Во-первых, применено рекуррентное соотношение, позволяющее существенно ускорить время и точность вычисления функции. Во-вторых, аргумент **x** с помощью операции **Mod** (остаток от деления на приведен к сравнительно небольшому интервалу, что увеличивает скорость сходимости и гарантирует работоспособность метода для больших значений **x**. Если не делать этой оптимизации, то для больших по модулю значений метод может давать некорректные результаты.

Теперь вернемся к технической стороне дела. Построим **Решение**, содержащее проект, для чего в **Главном меню** среды выберем элемент **Построение** и выполним команду ***Построить решение.*** В результате успешной компиляции будет построен файл с расширением **\*.dll**. Поскольку построенная сборка не содержит выполняемого файла, то непосредственно запустить наш проект на выполнение не удастся.

Далее построим консольный проект, к которому присоединим нашу библиотеку DLL, и протестируем, насколько корректно работает созданный нами метод. Заодно разберемся с тем, как строится консольный проект и как к нему подсоединяется сборка, содержащая DLL.

1. ***Создание консольного проекта***

Вначале необходимо построить интерфейс, обеспечивающий конечному пользователю доступ к тем сервисам, которые предоставляет созданный класс **ClassLibraryMath,** т.е. к вычислению **Sin(x)**. Начнем с построения простейшего интерфейса, позволяющего пользователю с консоли вводить исходную информацию – в нашем случае аргумент **x**, а затем полученные результаты вывести на консоль – экран дисплея. Для организации подобного интерфейса и служит тип проекта – **Консольное приложение.**

Добавим в уже существующее решение новый проект.

Чтобы создать новый проект, находясь в среде разработки, вовсе не обязательно начинать со стартовой страницы. Достаточно выбрать элемент ***Создать*** Главного меню **Файл** и выполнить команду ***Проект***, открывающий окно **Создания проекта**. В этом окне, как описано ранее, зададим шаблон строящегося проекта, и дадим ему имя **–** **ConsoleToMath**, а в строке ***Решение*** надо выбрать команду ***Добавить в решение***.

Можно также для добавления нового проекта в существующее решение выполнить команду ***Добавить*** элемента Главного меню **Файл**, а затем выбрать ***Создать проект.***

В консольном проекте автоматически создается стандартный модуль **Module1,** содержащий единственный статический метод – процедуру **Main()**. Если этот проект запустить на выполнение, то начнет выполняться код этой процедуры, который пока отсутствует, и который нам предстоит создать.

Когда проект запускается на выполнение, то в него должна входить сборка, отмеченная как стартовая, содержащая класс с точкой входа – статическим методом **Main().** Дальнейший сценарий зависит от содержимой точки входа. Как правило, в процессе работы метода **Main()** создаются один или несколько объектов других классов, они и вызывают методы и/или обработчики событий, происходящих с созданными объектами. В этих методах и обработчиках событий могут в свою очередь создаваться новые объекты, вызываться новые методы и новые обработчики. Так, начиная с одной точки, разворачивается целый мир объектов приложения.

Добавим в процедуру **Main()** свой программный код (Рис. 1.1-3**)**.

|  |
| --- |
| **Option Explicit On**  **Option Strict On**  **Module Module1**  **Sub Main()**  *'Входные данные*  **Dim x As Double = 0**  **Const INVITE As String = "Введите вещественное число x" + \_**  **"- аргумент функции Sin(x)"**  **Const CONTINU As String = "Продолжим? (Yes/No)"**  **Dim answer As String = "Yes"**  **Do** *'Организация ввода данных*  **Console.WriteLine(INVITE)**  **Dim temp As String = Console.ReadLine()**  **x = Convert.ToDouble(temp)**  *'Вычисления и вывод результата*  **Dim res As Double = 0**  **res = Math.Sin(x)**  **Console.WriteLine("Math.Sin(x) = " + res.ToString())**  **res = ClassLibraryMath.ClassLibraryMath.Sin(x)**  **Console.WriteLine("Sin(x) из библ.ClassLibraryMath=" + \_**  **res.ToString())**  *'диалог с пользователем*  **Console.WriteLine(CONTINU)**  **answer = Console.ReadLine()**  **Loop While (answer <> "No")**  **End Sub**  **End Module** |

*Рис. 1.1-3. Программный код консольного проекта* ***ConsoleToMath***

1. ***Связывание с DLL***

Первым делом свяжем два построенных проекта, для чего в консольный проект добавим ссылку на проект с DLL **ClassLibraryMath**. Для этого в окне **Обозреватель решений** подведем указатель мыши к имени консольного проекта, щелкнем правой кнопкой и из появившегося контекстного меню выполним команду ***Добавить ссылку***. Если необходимый проект с DLL, как в нашем случае включен в текущее **Решение**, то в открывшемся окне добавления ссылок надо выбрать вкладку ***Проект***, и он появится в открывшемся окне. Если ссылку нужно установить на проект, не включенный в текущее **Решение**, то в окне добавления ссылок нужно задать путь к проекту **ClassLibraryMath**. Для этого необходимо щелкнуть на вкладке ***Обзор*** и задать путь к проекту

|  |
| --- |
| **…ClassLibraryMath\bin\Debug\ClassLibraryMath.dll** |

где и находится нужный файл. После этого добавления ссылка на DLL появится в папке ***Ссылки*** консольного проекта. Теперь проекты связаны и из консольного проекта доступны сервисы, предоставляемые соответствующей библиотекой DLL, т.е. можно вычислить **sin(x),** вызвав этот записанный в библиотеке метод.

1. ***Организация консольного интерфейса***

Поясним программный код созданной процедуры **Main().**

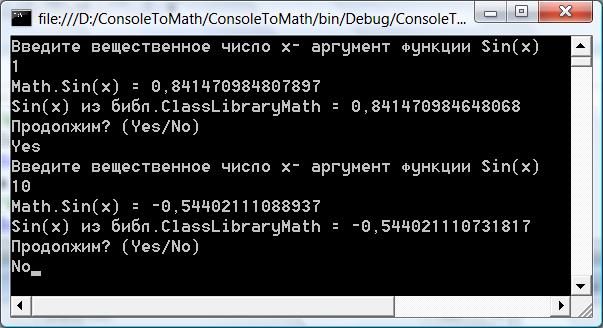
Для исходных данных задается лишь одна переменная **x** типа **Double***.*

Вводу значения **x** предшествует, как и положено для хорошего стиля программирования, приглашение к вводу. Для ввода и вывода значений используются статические методы **ReadLine()** и **WriteLine()** класса **Console**, входящего в библиотеку **FCL**. Ввод исходного значения аргумента **x** происходит в строковую переменную **temp.** Для преобразования введенных данных к нужному типу (строки к типу **Double**) используются статические методы класса **Convert**, сервисы которого позволяют проводить различные преобразования между типами данных.

Значение функции **Sin(x)** вычисляется двумя разными методами – методом стандартного класса **Math** и методом класса **ClassLibraryMath**.

В коде применяется стандартный прием зацикливания тела метода   
**Main()**, позволяющий пользователю самому решать, когда прервать выполнение метода.

На рис. 1.1-4 показаны результаты работы консольного проекта.

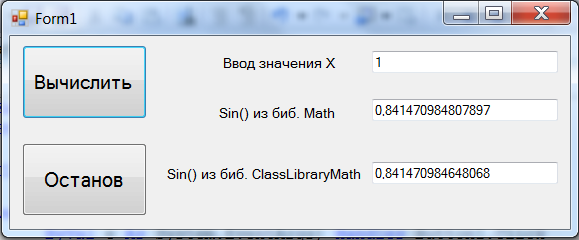


*Рис. 1.1-4. Результаты работы консольного проекта*

Анализируя эти результаты можно видеть, что два метода на всех исследуемых аргументах дают одинаковые результаты, совпадающие с точностью до 9 цифр после запятой. Точность метода в классе **ClassLibraryMath** обеспечивается константой **EPS** этого класса. Достигнутая точность достаточна для большинства практических задач.

1. ***Создание Windows проекта***

Создадим новый проект, аналогично тому, как был добавлен консольный проект. В качестве типа проекта выберем **Приложение Windows Forms**, дадим проекту имя **WindowsToMath**.



*Рис. 1.1-5. Пользовательский интерфейс формы класса Form1   
с результатами работы проекта*

|  |
| --- |
| **Option Explicit On**  **Option Strict On**  **Public Class Form1**  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button1.Click**  **Dim x As Double**  **Dim temp As String**  **temp = TextBox1.Text : x = Val(temp)**  *'Вычисления и вывод результата*  **Dim res As Double = 0**  **res = Math.Sin(x)**  **TextBox2.Text = Convert.ToString(res)**  **res = ClassLibraryMath.ClassLibraryMath.Sin(x)**  **TextBox3.Text = Convert.ToString(res)**  **End Sub**  **Private Sub Button2\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button2.Click**  **End**  **End Sub**  **End Class** |

*Рис. 1.1-6. Программный код класса Form1*

Создадим простой пользовательский интерфейс (рис. 1.1-5) и программный код класса **Form1** (рис. 1.1-6), и аналогично тому, как описано для консольного проекта, свяжем библиотеку **DLL** **ClassLibraryMath** с построенным проектом **WindowsToMath**.

Если в одном решении содержится несколько проектов (как в нашем примере), то при нажатии на кнопку ***Начать отладку*** будет выполняться тот проект, чье имя в **Обозревателе решений** выделено полужирным шрифтом. Чтобы выполнить другой проект решения, необходимо щелкнуть правой кнопкой мыши по имени этого проекта в Обозревателе решений и в открывшемся контекстном меню выбрать пункт ***Назначить запускаемым проектом***.

При создании **Консольного** проекта автоматически был создан класс, содержащий метод **Main()** с пустым кодом. В **Windows** – проекте автоматически был создан класс с именем **Form1**. Этот класс является наследником класса **Form** из библиотеки FCL и наследует все свойства и поведение (методы и события) родительского класса. Класс **Form** поддерживает организацию **интерфейса пользователя** **в визуальном стиле**. Форма является контейнером для размещения визуальных элементов управления – кнопок (**Button**), текстовых полей (**TextBox**), списков (**ListBox**) и более экзотичных элементов – таблиц (**DataGridView**), деревьев (**TreeView**) и многих других элементов. Некоторые элементы управления были изложены в [1].

Классы в VB синтаксически не являются неделимыми, а могут состоять из нескольких частей, каждая из которых начинается с ключевого слова **Partial *(частичный)***. Таковым является и построенный автоматически класс **Form1**. Возможность разбиения описания одного класса на части облегчает работу над большим классом. Каждая часть класса хранится в отдельном файле со своим именем. Одна часть класса **Form1** хранится в файле с именем **Form1.Designer.vb**. Эта часть класса заполняется автоматически инструментарием, называемым ***Дизайнером формы***. Когда мы занимаемся визуальным проектированием формы [1] и размещаем на ней различные элементы управления, меняем их свойства, придаем форме нужный вид, задаем обработчиков событий для элементов управления, то ***Дизайнер формы*** транслирует наши действия в действия над объектами соответствующих классов, создает соответствующий код и вставляет его в нужное место класса **Form1**. Предполагается, что разработчик проекта не вмешивается в работу **Дизайнера** и не корректирует часть класса **Form1**, созданную **Дизайнером**. Тем не менее, понимать код, созданный **Дизайнером** необходимо, а иногда полезно и корректировать его. Другая часть класса **Form1**, хранящаяся в файле **Form1.vb**, предназначена для разработчика, – именно в ней располагаются автоматически создаваемые обработчики событий, происходящих с элементами управления, код которых создается самим разработчиком. Такая технология программирования, основанная на работе с формами, называется **визуальной**, **событийно управляемой технологией.**

Обратите внимание, что каждое приложение VB должно содержать точку входа. Этой точкой входа является процедура с именем **Main()**. Платформа .NET Framework вызывает процедуру **Main()** при загрузке приложения. Процедура **Main()** содержит код, который будет выполняться в первую очередь.

Файл, запускаемый самостоятельно с расширением \***.exe**, должен также содержать процедуру **Main()**, в то время как библиотека с расширением \*.dll не может быть запущена самостоятельно и не требует процедуры **Main()**. Рассмотрим требования к процедуре **Main()**:

|  |
| --- |
| **Module Module1**  **Sub Main()**  **…**  **End Sub**  **End Module** |

Рис. 1.1-7. Автоматически генерированный программный код   
Консольного приложения

1. Консольные приложения выполняются самостоятельно, в них необходимо указать, одну процедуру **Main()**, которая создается автоматически, явно присутствует в программном коде проекта и в дальнейшем заполняется программным кодом разработчика (рис. 1.1.7).

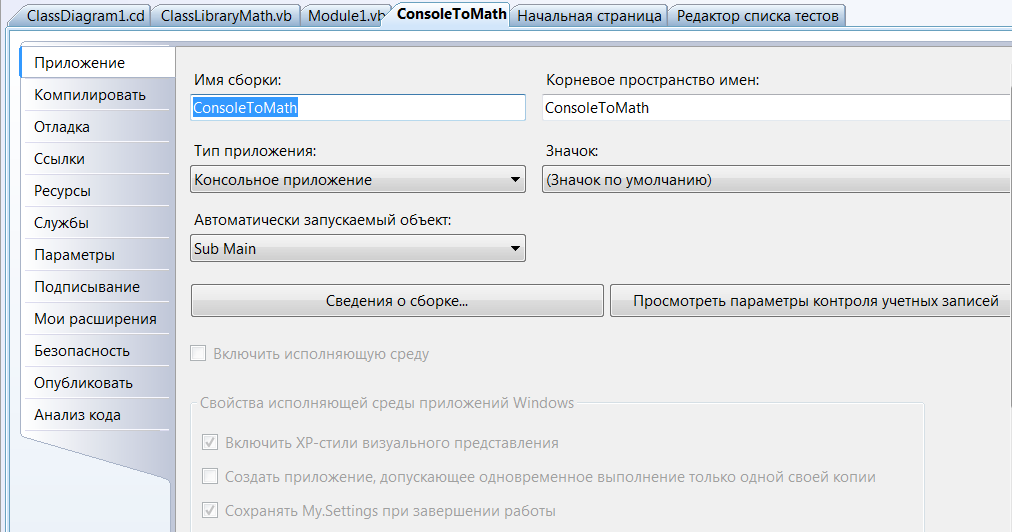


Рис. 1.1-8. Информация о стартовой процедуре в окне My Project

Итак, в VB стартовым объектом всегда является процедура Main(). Имя этой процедуры **Sub Main()** должно быть указано на вкладке Приложение диалогового окна **My Project** (Мой Проект) (рис. 1.1-8). Диалоговое окно **My Project** появляется на экране, если дважды щелкнуть по его имени в окне Обозреватель решений.

1. Приложения Windows Forms выполняются самостоятельно. Для этого VB создает процедуру **Main()** автоматически.

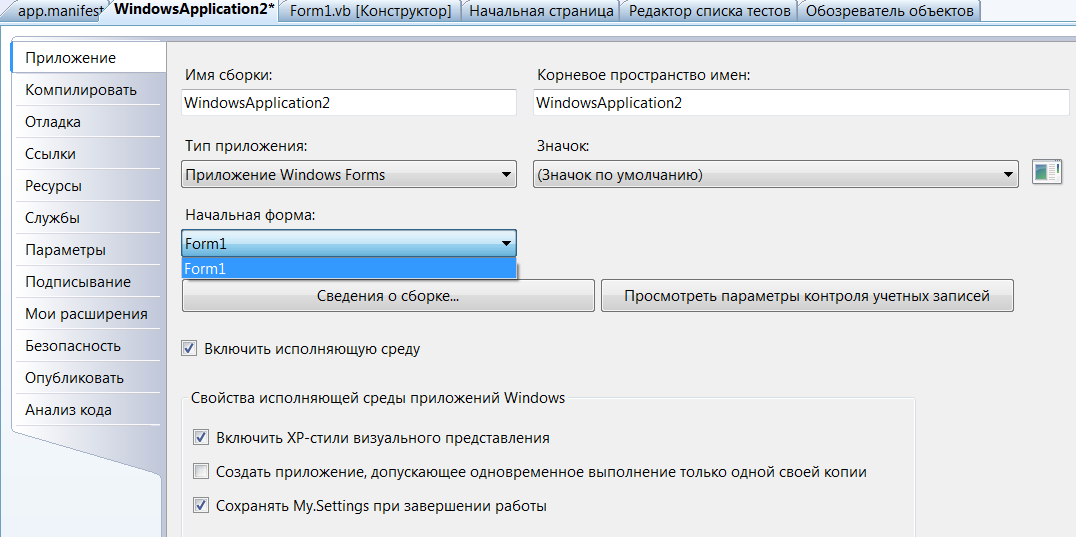


Рис. 1.1-9. Информация о стартовом объекте

Дело в том, что в VB поддерживается концепция стартового объекта, автоматически создаваемого при запуске приложения. По умолчанию стартовым объектом всегда является класс, производный от класса **Form**. Имя этого объекта **Form1** указывается на вкладке Приложение диалогового окна **My Project** (рис. 1.1-9). Диалоговое окно **My Project** появляется на экране, если дважды щелкнуть по его имени в окне Обозреватель решений.

В отличие от Консольного проекта, Windows Forms проект начнет выполнятся только после того, как со стартовым объектом произойдет одно из допустимых событий.

1. Библиотеки классов не требуют процедуры **Main().**

Итак, присоздании **Приложения Windows Forms** и **Консольных приложений** программный код проекта на языке VBможет состоять из ***программного кода, генерируемого средой разработки, и программного кода, написанного (дописанного) программистом.*** Причем базовая структура программного кода проекта создается автоматически в тот момент, когда пользователь выбирает в окне **Создание проекта**соответствующий шаблон. При выборе**Приложения Windows Forms*,*** базовая структура содержит ***модуль формы* (Class Form1)**, а при выборе **Консольных приложений,** базовая структура содержит ***стандартный модуль* (Module1)** и ***метод*** **Main()** внутри модуля.

По мере усложнения приложений к ним могут добавляться ***модули классов***, ***стандартные модули***, а если программа достаточно длинная, в ней может быть ***несколько форм*** и большое число ***процедур событий***, которые используют одни и те же переменные и процедуры. Часть программного кода можно выделить в отдельный программный код, который будет являться общим для всех. Такой программный код является частью ***стандартного модуля (модуля пользователя)***. Более подробно рассмотрим ***стандартные модули*** в следующем параграфе.

Таким образом, программный код VBхранится в ***программных модулях*** в виде процедур (методов). Эти программные модули, являющиеся, по сути, ***контейнерами*** для данных и процедур, могут быть трех видов:

* модуль формы, например, **Public Class Form1**;
* стандартный модуль, например, **Module Module1**;

модуль класса, например, **Public Class Class1**.

**Приложения Windows Forms** могут состоять из нескольких форм, нескольких стандартных модулей и нескольких классов***:***

|  |
| --- |
| **Option Explicit On**  **Option Strict On**  *'импортирование компонент других библиотек классов*  **Imports System.Math**  **Imports System.IO**  **…**  **Module Module\_1**  **…** *'стандартный модуль 1*  **End Module**  **…**  **Module Module\_n**  **…** *'стандартный модуль n*  **End Module**  **Public Class Class1**  **…** *'модуль класса 1*  **End Class**  **…**  **Public Class Class\_k**  **…** *'модуль класса k*  **End Class**  **Public Class Form1**  **…** *'модуль формы 1*  **End Class**  **…**  **Public Class Form\_m**  **…** *'модуль формы m*  **End Class** |

Как нам уже известно, при создании проекта или файла и открытии редактора кода, пользователю предоставляется уже готовая часть кода, расположенная в правильном порядке. Любой код должен создаваться в следующей последовательности:

1. Операторы **Option**.
2. Операторы **Imports**.
3. Операторы **Namespace** и элементы уровня пространства имен.

Размещение операторов в иной последовательности может вызвать ошибки компиляции.

Операторы **Option** и **Imports** были рассмотрены в [1].

Операторы **пространства имен** помогают организовывать и классифицировать элементы программирования для облегчения их группировки и доступа к ним. [Оператор **Namespace**](ms-help://MS.VSCC.v90/MS.MSDNQTR.v90.ru/dv_vbalr/html/a31fbd95-9ace-4c3d-bbb1-51222a2272b2.htm) используется для классификации операторов в определенном пространстве имен, объявляет имя пространства имен и вызывает исходный код ниже объявления для компиляции в пределах этого пространства имен:

|  |
| --- |
| **Namespace {***name | name.name***}**  [*componenttypes*]  **End Namespace** |

Где *name*: обязательное уникальное имя пространства имен,

которое должно быть допустимым идентификатором

Visual Basic;

*сomponenttypes*: дополнительный элемент пространства имен.   
 В пространство имен могут входить классы, модули,

интерфейсы, делегаты, другие пространства имен и

прочие элементы;

**End Namespace**: завершает блок **Namespace**.

Пространства имен используются в качестве организационной системы. Они предоставляют возможность классификации и представления элементов программирования, которые предоставляются другим программам и приложениям. Обратите внимание, что пространство имен не является типом как класс или структура, поэтому нельзя объявлять элемент программирования с типом данных пространства имен. Причем использовать инструкцию **Namespace** можно только на уровне файла или пространства имен. Это означает, что контекст объявления внутри пространства имен должен быть исходным файлом или другим пространством имен и не может быть классом, структурой, модулем, интерфейсом или процедурой.

Обратите внимание, что пространства имен обрабатываются так же, как если бы они имели уровень доступа **Public.** К пространству имен можно обращаться из кода в том же проекте, из других проектов, которые ссылаются на проект, и из любой сборки, построенной из проекта.

Любая библиотека .NET представлена в виде файла сборки, которая может содержать внутри себя любое количество пространств имён, которые в свою очередь содержат любое количество типов (классы, структуры и так далее), а типы в свою очередь могут содержать любое количество членов (поля, методы, свойства и так далее).

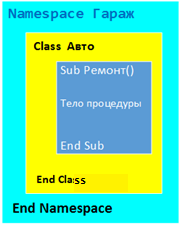
В результате получается:



Главное запомнить, что всегда нужно соблюдать правильную последовательность обращения к данным внутри сборки:

|  |
| --- |
| *ПространствоИмен.Тип.Член* |

Например:



Для того чтобы обратиться к методу **Ремонт** сначала нужно указать: пространство имён **Гараж**, затем класс **Авто** и лишь только потом указываем имя метода.

|  |
| --- |
| **Гараж.Авто.Ремонт** |

### 1.2. Стандартные модули и их использование

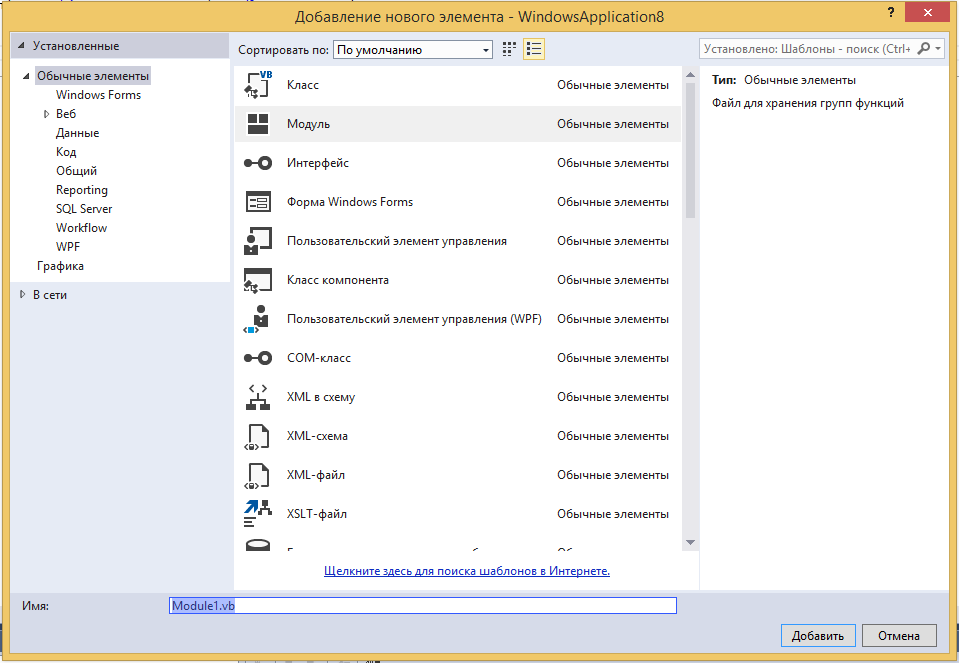
Итак, программный код Visual Basic хранится в программных модулях, которые могут быть трех видов и файлы которых имеют расширение **\*.vb**: **модуль формы, стандартный модуль и модуль класса**. Простейшее приложение, состоящее из одной формы, содержит, как правило, только модуль формы (например, **Form1.vb**). При добавлении в проект, например, второй формы, ее программный код будет храниться в модуле формы **Form2.vb**. Модули формы содержат объявления констант, переменных, типов данных, процедур и функций, а также событийных процедур, которые применяются на уровне модуля формы (т.е. используются в этой форме), хотя в них также можно ссылаться на другие формы и объекты данного проекта. Повторяющиеся функции и процедуры, выполняемые в нескольких формах, можно выделить в отдельный программный код, который будет являться общим для всех. Такой программный код называется ***стандартным модулем (модулем пользователя)***, и служит для хранения функций, процедур, констант и переменных, которые являются глобальными для всего приложения (проекта).

***Стандартный модуль*** – это специальный файл, имеющий расширение **\*.vb** и содержащий объявления переменных и описание про­цедур, которые могут быть использованы в любом месте программного кода проекта.

Как и модули формы, стандартные модули отображаются в окне   
**Обозреватель решений**, и могут быть сохранены на диске с помощью команды ***Со­хранить Module****Name****.vb как*** элемента главного меню **Файл**.Стандартный модуль ***Module****Name****.vb***является элементом проекта. В отличие от модуля форм, стандартные модули содержат толь­ко программный код и не содержат интерфейса пользователя. Модули класса создаются при использовании объектно-ориентированного программирования и будут рассмотрены позже. И хотя стандартные модули имеют некоторые сходства с классами (и те, и другие являются ***контейнерами для данных и процедур)***, они отличаются от классов тем, что не являются объектно-ориентированными, стандартные модули не определяют структуры и характеристики объектов и не могут наследоваться.

Чтобы создать в программе новый стандартный модуль, необходимо щелкнуть на кнопке ***Добавление нового элемента*** стандартной панели инструментов или на команде ***Добавить новый элемент…*** элемента Главного меню **Проект**. Также можно использовать команду ***Добавить модуль…*** элемента главного меню **Проект*.*** В результате выполнения одной из этих команд откроется диалоговое окно **Добавление нового элемента**, в котором следует выб­рать шаблон ***Модуль*** (***Module)*** и указать имя модуля (рис.1.2-1).

После нажатия на кнопку ***Добавить*** в окне **Редактор кода** по­явится новый пустой модуль. Первый создаваемый модуль проекта по умолчанию имеет имя **Module1.vb**, но это имя можно изменить, сделав щелчок пра­вой кнопкой мыши на модуле в окне **Обозреватель решений** и введя новое имя моду­ля, или с помощью команды ***Сохранить Module1.vb как*** элемента меню **Файл.**



*Рис. 1.2-1. Диалоговое окно* ***Добавление нового элемента***

Рассмотрим пример создания в проекте пустого модуля.

**Пример 1.2-1. Создать пустой стандартный модуль.**

1. Запустить VS и создать новый проект с именем Пустой модуль.

Будет создан новый проект, и в окне Конструктор появится пустая

форма.

1. Выполнить команду ***Добавить новый элемент…*** элемента главного меню **Проект**.

Появится диалоговое окно Добавление нового элемента.

1. Выбрать шаблон ***Модуль***.

В текстовом поле Имя появится имя по умолчанию **Module1.vb**.

(рис. 1.2-1)

1. Щелкнуть на кнопке ***Добавить***.

VB добавит в проект модуль **Module1**. Этот модуль появится в окне   
 Редактор кода, как показано на рис. 1.2-2.



*Рис. 1.2-2. Окно* ***Редактора кода*** *с добавленным в проект* ***Стандартным модулем Примера 1.2-1***

1. Чтобы увидеть окно **Обозреватель решений**, дважды щелкните мышью на строке заголовка окна **Обозреватель решений**, Окно **Обозреватель решений** будет выглядеть так, как на рис. 1.2-3.

******

Рис. 1.2-3. Окно Обозреватель решения проекта Пустой модуль ***Примера 1.2-1***

Окно Обозреватель решений в списке компонент проекта содержит модуль, ко­торый был добавлен в программу. Имя **Module1** указывает на имя фай­ла модуля по умолчанию, которое впоследствии можно изменить.

Выберите **Module1.vb** в окне **Обозреватель решений**.

1. Чтобы увидеть окно **Свойства**, дважды следует щелкнуть на строке его заголовка. Как показано на рис. 1.2-4, окно **Свойства** отображает свойства **Module1.vb**.

Так как модуль содержит только код, он имеет всего несколько свойств. Используя наиболее важное его свойство – Имя файла – можно указать другое имя файла модуля, сделав его более описатель­ным в будущем.



*Рис. 1.2-4. Окно* ***Свойства*** *стандартного модуля* ***Module1.vb***

1. Изменить свойство **File Name** (**Имя файла**)на *Математические*

*Функции.vb* или другое имя файла и нажать на **<**Enter**>.**

Имя файла модуля в окнах **Свойства**, **Обозреватель решений** и в окне

**Редактор кода** обновится.

1. Вернуть окна **Свойства** и **Обозреватель решений** в их нормальные прикреп­ленные положения, дважды щелкнув мышью на их заголовках.
2. Чтобы удалить модуль из проекта, следует щелкнуть на модуле в окне **Обозреватель решений**, а затем щелкнуть на команде ***Исключить* *из проекта*** элемента главного меню **Проект.**
3. Команда ***Исключить из проекта*** не удаляет модуль с жесткого диска, а удаляет связь между указанным модулем и текущим проектом. Чтобы отме­нить эффект этой команды, щелкнуть на команде ***Добавить существующий элемент*** элемента Главного меню **Файл**, выбрать файл, который требуется добавить в проект, и затем щелкнуть на кнопке ***Добавить***.
4. Объявление глобальной, или открытой переменной в модуле достаточно просто – ввести ключевое слово **Public**, за которым следует имя переменной и объявление ее типа. После того как переменная будет объявлена, читать ее содержимое, изменять его или отображать его на экране можно в любой процедуре программы.

Например, следующий оператор объявляет открытую переменную с именем **Global** типа **Integer**:

|  |
| --- |
| **Public Global As Integer** |

### 1.3. Создание проектов с несколькими формами

Проекты, которые создавались ранее, состояли из одной формы. Однако большинство реальных задач требуют, чтобы проекты имели более сложную структуру. Причем при использовании в проекте нескольких форм, взаимодействие между ними просто реализовать с помощью стандартного модуля, так как объекты, объявленные в стандартном модуле как **Public,** доступны всем формам проекта.

Рассмотрим пример, как создать вторую форму и организовать взаимодействие между этими формами.

**Пример 1.3-1. Создать проект с двумя формами.**

1. Создать новый проект с именем Проект-1.3-1.

Автоматически будет создана форма **Form1**.

1. Чтобы добавить в проект еще одну форму, необходимо поступить так же, как при добавлении модуля, т.е. выполнить команду ***Добавить новый элемент…*** из Главного меню **Проект**, и в появившемся диалоговом окне **Добавление нового элемента** выбрать элемент ***Форма Windows Forms***.

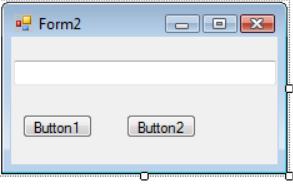
После этого в окне **Обозреватель решений** появится информация о том, что в проект добавлена форма **Form2**.

1. Загрузить проект.

На экране отобразится первая форма **Form1**, а вторая нет.

Обратите внимание, что в качестве стартовой формы можно выбрать любую форму в окне **Свойства Проекта-1.3-1**, щелкнув на элементе Главного меню **Проект** и выполнив команду ***Свойства: Проект-1.3-1***, как было ранее показано на рис. 1.1-9.

1. Разместить на формах элементы управления (кнопки и одно текстовое поле), как показано на рис. 1.3-1 и рис. 1.3-2.

|  |  |
| --- | --- |
|  |  |

*Рис. 1.3-1.Интерфейс пользователя Рис. 1.3-2. Интерфейс пользователя*

**Form1 Form2**

1. Добавить в проект стандартный модуль.
2. Добавить программный код модуля и двух форм (рис. 1.3-3 – 1.3-5).

|  |
| --- |
| **Public Class Form1**  *'Процедура для кнопки окрашивания форм и очистки текст. полей*  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button1.Click**  **Me.BackColor = Color.Blue**  **F2.BackColor = Color.Yellow**  **TextBox1.Text = ""**  **F2.TextBox1.Text = ""**  **End Sub**  *'Процедура для кнопки заполнения текстовых полей обеих форм*  **Private Sub Button2\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button2.Click**  **TextBox1.Text = " Форма Form1 приветствует саму себя"**  **F2.TextBox1.Text = "Привет Форме2 от Формы1"**  **F2.Focus()**  **End Sub**  *'Событийная процедура загрузки формы1*  **Private Sub Form1\_Load(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles MyBase.Load**  **F1 = Me**  **End Sub**  *'Процедура для кнопки создания и показа экземпляра формы2*  **Private Sub Button3\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) ­\_**  **Handles Button3.Click**  **F2 = New Form2** *'создается новый экземпляр формы2*  **F2.Show()**  **End Sub**  **End Class** |

Рис. 1.3-3. Программный код формы **Form1**

|  |
| --- |
| **Public Class Form2**  *'Процедура для кнопки окрашивания форм и очистки текст. полей*  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button1.Click**  **Me.BackColor = Color.Green**  **F1.BackColor = Color.Tomato**  **TextBox1.Text = ""**  **F1.TextBox1.Text = ""**  **End Sub**  *'Процедура для кнопки заполнения текстовых полей обеих форм*  **Private Sub Button2\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button2.Click**  **TextBox1.Text = "Форма Form2 приветствует саму себя"**  **F1.TextBox1.Text = "Привет Форме1 от Формы2 "**  **F1.Focus()**  **End Sub**  *'Событийная процедура загрузки формы2*  **Private Sub Form2\_Load(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles MyBase.Load**  **TextBox1.Text = "Я родилась"**  **End Sub**  **End Class** |

Рис. 1.3-4. Программный код формы **Form2**

|  |
| --- |
| **Module Module1**  *'Объявление глобальных переменных,*  *'доступных для всех форм и процедур проекта*  **Public F1 As Form1**  **Public F2 As Form2**  **End Module** |

Рис. 1.3-5. Программный код модуля **Module1**

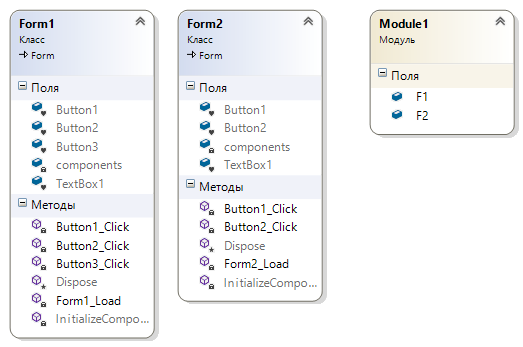


Рис. 1.3-6. Схема классов **Form1, Form2** и модуля **Module1**

***Примера 1.3-1***

Проанализируем вышеприведенные программные коды.

Прежде всего, несколько слов о том, видны ли модульные переменные **F1** и **F2** из других модулей проекта. Если они объявлены с помощью **Dim** или **Private**, то не будут видны, а если с помощью **Public**, как на рис. 1.3-5 - то будут видны.

Модули используются в качестве ***общедоступного конт*е*йнера*** для переменных, констант, процедур, функций и других элементов VB. В нашем случае модуль используется для объявления двух общедоступных переменных: **F1** и **F2**. Переменная **F1** объявляется как объект класса **Form1**. Это значит, что после загрузки формы **Form1** мы хотим к ней обра­щаться по имени **F1**. С этой же целью в процедуру **Form1\_Load()** вставлен оператор **F1=Me**, что позволяет ссылаться переменной **F1** на этот «экземпляр» формы.

После запуска на выполнение проекта форма **Form1** появляется на экране. Чтобы появилась и форма **Form2**, необходимо нажать кнопку **Button3**. Оператор **F2=Form2** или **F2=New** **Form2** порождает форму **Form2** как объект класса **Form**. А чтобы форма стала видна, должен выполниться оператор **F2.Show()**. Причем, если использовать оператор **F2=New Form2**, то при каждом нажатии на кнопку **Button3** будет каждый раз порождаться новая форма **Form2** (их на экране станет столько, сколько раз была нажата эта кнопка). Оператор **F2=Form2** (без слова **New**) порождает форму **Form2** только один раз.

Теперь на экране обе формы, и можно работать с каждой из них, нажимая кнопки и вводя текст в текстовые поля.

Посмотрим на содержимое процедур **Button1\_Click(…)** в окнах кода двух форм. Первая строка обеих процедур красит своего хозяина, вторая - соседа. Таким образом, в каждом окне кода можно, как обычно, писать операторы, приказывающие что-то сделать в форме-хозяине данного окна, а можно с таким же успехом писать операторы, приказывающие что-то сделать в любой другой форме проекта. Чтобы VB понимал, о какой форме идет речь, перед именем свойства, принадлежащего чужой форме, нужно писать имя этой формы (в нашем случае **F1** или **F2**) с точкой. Если же форма своя, то вместо имени хозяина можно писать слово **Me** или вообще ничего не писать. Так, в процедуре **Button1\_Click(…)** первой формы вместо

|  |
| --- |
| **Me.BackColor = Color.Blue**  *' можно написать*  **F1.BackColor = Color.Blue**  *' или просто*  **BackColor = Color.Blue** |

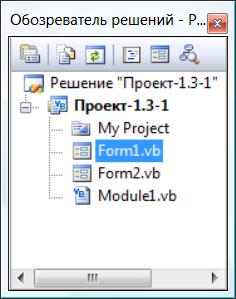
Теперь посмотрите на содержимое процедур **Button2\_Click(…)** в обоих окнах кода. Очевидно, что первая строка обеих процедур выводит текст в текстовое поле **TextBox1**своего хозяина, вторая - в текстовое поле **TextBox1** соседа. Третья строка обеих процедур при помощи метода **Focus()**, принадлежащего соседу, переводит фокус на соседа. Таким образом, можно обращаться не только к свойствам чужой формы, но и к ее элементам управления, и к методам. К ним тоже применимо соглашение об имени хозяина с точкой.

Рассмотрим структуру проекта **Проект-1.3-1.** Ее можно увидеть в окне **Обозреватель решений (Solution Explorer)** (рис. 1.3-7).

Строки окна **Обозреватель решений** отражают тот факт, что проект   
**Проект-1.3-1** состоит из трех модулей: формы **Form1**, формы **Form2** и стандартного модуля **Module1**, причем каждый из трех упомянутых модулей сохраняется в своем файле.

Щелкая в окне **Обозреватель решений** по нужному модулю, а затем (для форм) по одной из двух кнопок слева на панели окна **Перейти к коду и Открыть в Конструкторе,** мы можем удобно пере­ключаться между различными окнами модулей.

Щелкая правой клавишей мыши в окне, **Обозреватель решений**(рис.1.3-7) по нужному модулю, а за­тем в открывшемся контекстном меню выбирая нужный пункт, вы можете этот модуль переименовывать, копировать, вырезать, удалять из проекта, удалять с диска, просматривать и изменять его свойства. Обратите внимание, что, переименовав форму в окне свойств формы, вы не сможете запустить проект, если он с этой формы стартует. Вам понадобится зайти в окно ***Свойства* Проект-1.3-1**элемента главного меню ***Проект***.



*Рис. 1.3-7. Окно* ***Обозреватель решений*** *проекта* ***Проект-1.3-1***

### 1.4. Задачи для самостоятельного решения

1. В **n** разных школах г. Москвы (номера школ известны) проведен анонимный опрос **m** школьников 6 классов (числа **n** и **m** известны). Школьникам задавался вопрос, курят ли они. В опросе участвовали как мальчики, так и девочки. Результаты опроса поступали неупорядоченными по школам. Определить школы с максимальным процентом курящих отдельно по мальчикам и девочкам. Предусмотреть проверку правильности ввода информации.
2. Известны фамилии **n** сотрудников больницы (**n** - заданное число). Сотрудники работают по индивидуальному графику (1-й день - утро, 2-й день - вечер, 3-й день - выходной). В первое число месяца известно, в какую смену работает каждый сотрудник или имеет выходной. Составить алгоритм и программу, которые позволяют на любое заданное число месяца напечатать пронумерованный список сотрудников, имеющих выходной. Считать, что в месяце 30 дней.
3. Известны названия **m** предприятий города, производящие по **n** наименований различных изделий, для которых известны количества их выпуска в год. Составить алгоритм и программу, которые по наименованию изделия определяют название предприятия, производящего максимальное количество этого изделия, а также печатают список предприятий, на которых это изделие производится, располагая названия этих предприятий в порядке убывания объема выпуска данного изделия.
4. Список **n** рабочих цеха (**n** – заданное число) содержит следующие сведения о каждом: фамилия, числовой номер бригады, зарплата. Список не имеет определенной упорядоченности. Составить алгоритм и программу, которые выводят на экран списки рабочих по бригадам, располагая фамилии в порядке убывания зарплаты, и, кроме того, определяют и печатают значения средних зарплат по бригадам.
5. Пароходство владеет **n** пассажирскими теплоходами (**n** и названия теплоходов заданы), каждый из которых имеет каюты люкс на 1 и 2 палубах, первого класса на 2 и 3 палубах, второго класса на 3 и 4 палубах и третьего класса на 3 и 4 палубах. На каждом из теплоходов каюта может быть занята, забронирована или свободна. Составить алгоритм и программу для вывода по указанному пользователем теплоходу отдельно информации о количестве свободных кают и отдельно информации о количестве забронированных кают. Информацию выдать по каждой из палуб в порядке возрастания их номеров с указанием класса кают.
6. В течение месяца (30 дней) абоненты международной телефонной сети производили переговоры с **m** странами. Каждая страна имеет свой код и тариф за 1 мин. Составить алгоритм и программу, которые печатают счета абонентам, в которых указывается телефон абонента, код страны, количество минут, стоимость каждого разговора и итог за месяц. В общем списке абоненты представлены телефонными номерами, а для каждого разговора известна его продолжительность.
7. В чемпионате по футболу приняло участие **n** команд (названия команд известны). Составить алгоритм и программу, которые определяют 4 команды, показавшие лучшие результаты, если известен счет каждого матча. При равных условиях лучшей считается команда, у которой разность между забитыми и пропущенными мячами лучше.
8. В течение года (12 месяцев) отслеживалось изменение курсов **n** валют. Известны названия валют и их курс по отношению к рублю на последнее число каждого месяца. Составить алгоритм и программу, которые за каждый квартал определяют валюты, у которых наблюдался наибольший и наименьший рост относительно среднего квартального курса данной валюты.
9. Имеются сведения о доходах пяти магазинов по дням в течение одного месяца (в месяце 30 дней). Составить алгоритм и программу, которые для каждого магазина определяют числа месяца и соответствующие им дни недели с минимальным доходом, учитывая, что по субботам и воскресеньям магазины закрыты, а первое число месяца - вторник.
10. В отделе кадров предприятия имеется список сведений о **n** сотрудниках. О каждом сотруднике известно: фамилия (в списке фамилии расположены по алфавиту), номер отдела, должность, оклад и стаж работы. Составить алгоритм и программу, которые выводят списки фамилий сотрудников по отделам с указанием занимаемой должности, оклада и стажа.
11. Каждый из пяти цехов завода составляет ежемесячную заявку на приобретение по **M** разных комплектующих деталей. В заявке указаны: наименование детали, требуемое количество и завод-изготовитель (поставщик). Составить алгоритм и программу, которые выводят сводные списки заказов деталей для рассылки по поставщикам, в которых указаны наименования деталей с указанием общего количества требуемых деталей.
12. В бюро по трудоустройству имеются сведения о **N** вакансиях на предприятиях города. О каждом вакантном месте известно: название предприятия, должность, оклад, требуемое образование (высшее или среднее), предельный возраст. Составить алгоритм и программу, которые по данным об образовании и возрасте печатают список вакансий, располагая их в порядке убывания окладов.
13. В префектуре имеются сведения о потребностях в улучшении жилищных условий **n** семей. О каждой семье известно, на квартиру из скольких комнат она претендует. Составить алгоритм и программу, которые печатают список семей, получивших квартиры в новом доме, и список семей, оставшихся в очереди, если известно, что в доме 100 квартир, из которых 30 однокомнатных, 40 – двухкомнатных и 30 – трехкомнатных.
14. Имеются сведения об урожаях зерна в **n** регионах России за **m** лет. Составить алгоритм и программу, которые определяют самый урожайный и самый неурожайный годы и для каждого года печатают список регионов в порядке убывания полученного в них урожая.
15. Список участниц конкурса красоты содержит **n** фамилий. Каждый из **m** опрошенных называет трех претенденток, и определяет каждой из них одно из первых 3-х мест. Вывести на экран фамилию участницы, набравшей наибольшее число первых мест. Если таких участниц несколько, то вывести фамилии всех участниц.
16. В типографии заказано **n** визитных карточек. Составить алгоритм и программу, которые определяют стоимость заказа, если первые **n** визиток стоят 5 руб. за штуку, а стоимость каждой последующей карточки уменьшается соответственно на 1 коп.
17. Для каждого из **n** классов школы известно, сколько в них мальчиков и сколько девочек. Составить алгоритм и программу, которые выводят номера классов по убыванию процентного состава мальчиков и определяют процент девочек и мальчиков по школе в целом.
18. В доме проводится остекление окон. Количество квартир в доме и количество комнат в каждой квартире (1, 2 или 3) известно. Составить схему алгоритма и программу, которые определяют, сколько м2 стекла надо закупить и общую стоимость стекла, если в трехкомнатной квартире площадь окон составляет 10 м2, в двухкомнатной – 7 м2, в однокомнатной – 5 м2, а стоимость 1 м2 стекла 200 руб.
19. Составить алгоритм и программу, которые выводят список фамилий школьников по убыванию их среднего балла за год, а также определяют процент отличников (все отметки 5) и процент хорошистов (отметки 5 и 4), если для каждого из 25 школьников известны итоговые оценки по 15 предметам.
20. Объем файла, подлежащего архивации, составляет **S** мегабайт. При архивации объем файла уменьшается: программой ZIP – на **30%**, а программой ARJ – на **25%**, после чего архив разбивается на дискеты по 1,44 мегабайт. Составить алгоритм и программу, определяющую необходимое количество дискет для архивации файла заданного объема в зависимости от типа выбранной программы архивации (тип программы архивации вводится).

### 1.5. Лабораторная работа по теме

### *«Модульная структура приложений и стандартные модули в Visual Basic»*

Цель данной лабораторной работы состоит в изучении средств, приемов и получении практических навыков разработки, написания и отладки проектов, использующих несколько форм, закреплении материала по обработке строковых данных и текстовых файлов, и практическом использовании стандартных модулей при решении различных задач.

#### **Общее задание**

1. Изучите основные понятия и определения модульной структуры приложений VB, использующих несколько форм, а так же способы работы со стандартными модулями (**Тема 1**).
2. Изучите общее задание в п. 1.5-2 и выберите вариант задания из табл. 1.5-1 в соответствии с указанием преподавателя, а также по указанию преподавателя выберите варианты реализации проекта: со стандартным модулем и двумя формами и вводом с клавиатуры; со стандартным модулем и двумя формами и вводом из файла.
3. Проведите формализацию задачи, для чего:

* определите перечень исходных данных и предусмотрите для них проверку правильности ввода;
* представьте смысловые данные, которые будут использоваться в ходе решения задачи в виде переменных и массивов, присвоив им соответствующие имена;
* определите размерности и типы используемых данных;
* представьте выходные данные в виде массивов;
* приведите геометрическую иллюстрацию решения задачи;
* при необходимости приведите расчетные формулы, которые будут использоваться для преобразования исходных данных в результаты.

1. Разработайте интерфейс пользователя, то есть:

* разработайте форму, предназначенную для ввода исходных данных, как с клавиатуры, так и из текстового файла, и для отображения входных данных на форме;
* разработайте выходную форму, предназначенную для вывода результатов решения задачи, как на форму, так и для записи в файл.

1. Составьте схемы алгоритмов решения функциональных.
2. Создайте программный код проекта, реализующий поставленную задачу.
3. Подготовьте тестовые наборы исходных данных, соответствующие смысловой направленности решаемой задачи как для ввода с клавиатуры, так и из текстового файла. Формат текстового файла должен быть задан преподавателем (см. примеры выполнения задания п. 1.5.4), а сам текстовый файл с расширением \*.txt должен быть создан студентом средствами MS Windows, например, в Блокноте.
4. Выполните проект и получите решение.
5. Обоснуйте правильность полученных результатов.

#### 

#### **Варианты индивидуального задания**

Разработайте вариант программного проекта задач из табл. 1.5-1.

Таблица 1.5-1

|  |  |
| --- | --- |
| 1)  2)  3)  4)  5)  6)  7)  8)  9)  10)  11)  12)  13)  14)  15)  16)  17)  18)  19)  20)  21)  22)  23)  24)  25)  26)  27)  28)  29)  30) | В разных городах провели обследование одинакового процента жителей на предмет употребления или неупотребления ими наркотиков. Общее число обследованных составило n человек (n задано). Результаты обследования жителей поступают по городам неупорядоченными. Составить алгоритм и программу, которые определяют города с максимальной тенденцией к употреблению наркотиков, также составляют список городов по убыванию тенденции к употреблению наркотиков.  В ГИБДД имеется информация о n владельцах автомобилей. О каждом автовладельце известно: фамилия, номер его машины, год ее выпуска, цвет, год прохождения технического осмотра. Необходимо составить алгоритм и программу, для каждой марки определяют фамилию владельца самой старой машины и количество автомобилей, не проходивших техосмотр последние два года.  Каждое из n предприятий города (n - заданное число) выпускает 5 одинаковых наименований продукции (наименования предприятий и наименования продукции заданы). Для каждого из предприятий известны объем выпуска и стоимость единицы продукции каждого вида. Необходимо для каждого предприятия города определить процентную долю стоимости производимой им продукции каждого вида.  Имеются сведения об n студентах (n - заданное число): фамилия, номер группы и 4 экзаменационные отметки. Составить алгоритм и программу, которые для каждой группы определяют количество двоечников и количество студентов, сдавших сессию на “4” и “5” и составляют список студентов группы с максимальным числом двоечников.  В профкоме завода имеются неупорядоченные сведения об n рабочих: номер цеха, фамилия, год рождения, нуждается ли он в жилплощади (0 - нет, 1 - да). Составить алгоритм и программу, которые для каждого цеха находят общее количество необходимых квартир и определяют самого старого рабочего, нуждающегося в жилплощади для этого цеха.  Имеется список n команд, принимающих участие в чемпионате мира по футболу. Перед турниром проведен опрос k человек, каждый из которых назвал по 3 команды, которые с его точки зрения, займут 1-е, 2-е и 3-е место. Составить алгоритм и программу, которые по результату турнира определяют приз каждого из тех опрошенных, которые точно угадали последовательность победителей, и составляют их список, если общий призовой фонд - S.  Ремонтное предприятие выполняет 6 видов работ. Для каждого вида известно название и стоимость. К предприятию обратились n заказчиков с предложениями выполнить определенное количество работ каждого вида. Составить алгоритм и программу, которые для каждого заказчика определяют сумму договора с ремонтным предприятием, а также организуют формирование списка заказчиков в порядке убывания сумм договоров.  В гостинице проживает n постояльцев (n - заданное число). О каждом постояльце известны три характеристики: номер проживания, фамилия, заказанное им блюдо (или отсутствие заказа). Необходимо составить сводные (по наименованиям заказанных блюд) заявки на кухню с указанием необходимого количества каждого блюда и составить список фамилий постояльцев, заказавших самое популярное блюдо.  Даны итоги выборов в Думу города: по k округам известно число голосовавших за представителей m партий. Считая, что от каждого округа в Думу проходит по одному представителю, и, предполагая, что все кандидаты набрали разное число голосов, составить алгоритм и программу, которые для каждой партии определяют число прошедших в Думу своих представителей, и название округа, в котором партия-победитель набрала наибольшее число голосов  В автохозяйстве имеются n автомашин (n - заданное число). Для каждого автомобиля заданы три характеристики: номер автомашины, марка автомашины, тип неисправности (или ее отсутствие). Составить алгоритм и программу, которые составляют сводные заявки на ремонт (по типу неисправностей) с указанием нуждающихся в этом типе ремонта количества автомашин и составляют общий список машин (с указанием марки и номера) для самого «популярного» ремонта.  Для каждого из 10-ти кинотеатров города имеются сведения о количестве мест и количестве билетов, проданных на каждый из пяти сеансов. Считая заданной среднюю стоимость билета, составить алгоритм и программу, которые формируют список кинотеатров, у которых убытки превышают выручку от проданных билетов и оценивают общие потери всех 10-ти кинотеатров.  Имеется список n сотрудников фирмы (фамилия и пол), работающей без выходных дней. Ежедневно в течение июня регистрировали их выход на работу. Составить алгоритм и программу определения числа месяца, когда был зарегистрирован максимальный процент сотрудников, не вышедших на работу и сформировать списки фамилий сотрудников (отдельно мужчин и женщин), работавших в этот день.  Имеется список 60-ти зданий города, подлежащих реконструкции. Сведения о каждом здании содержат название микрорайона, улицу, номер дома и год постройки. Составить алгоритм и программу определения количества самых старых зданий, подлежащих реконструкции в каждом микрорайоне, и найти микрорайон с наибольшим количеством таких домов. Для найденного микрорайона сформировать список, содержащий полные сведения об этих домах.  В бюро по ремонту квартир поступили сведения о состоянии n кооперативных квартир (n - задано). Сведения о каждой квартире содержат: номер квартиры, фамилию владельца и тип необходимого ремонта (или отсутствие необходимости в ремонте). Составить алгоритм и программу, позволяющие составить заявки на ремонт каждого типа по убыванию количества нуждающихся в нем квартир, и отдельный список на самый «непопулярный» ремонт с указанием номеров квартир и фамилий владельцев.  Составить алгоритм и программу, которые составляют общий перечень необходимых марок машин для n фермерских хозяйств с указанием их количеств. Причем каждое фермерское хозяйство представило свой перечень, состоящий из m машин разных наименований (марок) и потребность в каждой из них (n и m - заданы).  Произвели опрос n студентов литературного института (n - заданное число), каждый из которых назвал две фамилии своих любимых писателей. Составить алгоритм и программу, которые определяют, сколько и какие писатели были названы, и сколько раз был назван каждый из них. Сформировать 2 списка писателей: по алфавиту и по убыванию количества упоминаний студентами.  Произведен опрос 1000 жителей Москвы. Каждый из них назвал фамилию самого любимого телеведущего. Составить алгоритм и программу формирования двух списков фамилий первых десяти наиболее популярных ведущих, расположив фамилии в первом списке в порядке уменьшения количества поданных за них голосов, а во втором – по алфавиту. Учесть, что общее число фамилий может быть меньше 10-ти.  Имеются сведения об урожаях зерна в n регионах России за m лет. Составить алгоритм и программу, которые определяют самый урожайный и самый неурожайный год и для каждого из найденных лет печатают список регионов в порядке убывания полученного в них урожая.  Известны названия m предприятий города, производящие поn наименований различных изделий, для которых известны количества их выпуска в год. Составить алгоритм и программу, которые по наименованию изделия определяют название предприятия, производящего максимальное количество этого изделия, а также печатают список предприятий, на которых это изделие производится, располагая названия этих предприятий в порядке убывания объема выпуска данного изделия.  Список n рабочих цеха (n – заданное число) содержит следующие сведения о каждом: фамилия, числовой номер бригады, зарплата. Список не имеет определенной упорядоченности. Составить алгоритм и программу, которые для каждой бригады определяют фамилии самого низко- и самого высокооплачиваемого рабочего, а также находят значение средней зарплаты по этой бригаде и количество тех рабочих, чья зарплата больше средней зарплаты.  В бюро по трудоустройству имеются сведения о N вакансиях на предприятиях города. О каждом вакантном месте известно: название предприятия, должность, оклад, требуемое образование (высшее или среднее), предельный возраст. Составить алгоритм и программу, которые по данным об образовании и возрасте формируют список вакансий, располагая их в порядке убывания окладов.  Составить алгоритм и программу, которые определяют процент отличников (все отметки 5) и процент хорошистов (отметки 5 и 4), формируют список фамилий отличников исписок фамилий хорошистов, а также общий список школьников по убыванию их среднего балла за год, если для каждого из 25 школьников известны итоговые оценки по 15 предметам.  Известны фамилии и месячные заработки 10 рабочих бригады в течение года. Составить алгоритм и программу, которые формируют список фамилий рабочих, имевших хотя бы раз минимальный в этом году для члена бригады месячный заработок, а также список фамилий тех рабочих, среднегодовой заработок которых выше общего среднего заработка по бригаде в целом.  На продажу выставлено N квартир. О каждой квартире известно: адрес, категория дома (кирпичный, монолитный, панельный), общая площадь, количество комнат, цена 1 м2. Составить алгоритм и программу, которые для обратившегося клиента формируют списки подходящих квартир (все сведения) с разбивкой по категориям в порядке уменьшения их стоимости. Квартира считается подходящей, если ее общая площадь не менее требуемой, а ее цена не превышает требуемую.  Рекламное агентство по продаже моющих средств, провело опрос N случайных прохожих, каждый из них назвал название самого качественного и самого некачественного моющего средства. Составить алгоритм и программу, которые определяют, список каких моющих средств оказался длиннее, а также печатают список качественных моющих средств по убыванию популярности, а некачественных – по алфавиту.  Известны наименования и количество 10-ти видов товаров, на каждой из 15-ти баз города, а также потребность города в товаре этого вида. Составить алгоритм и программу, которые для каждого товара определяют номер базы с его максимальным запасом и формируют список товаров, которые необходимо пополнить, чтобы удовлетворить потребности города.  В театральную кассу поступило n заявок от школ города. Каждая заявка содержит название спектакля и необходимое количество билетов. Составить алгоритм и программу, которые на основании этих данных определяют популярность каждого спектакля и печатают их названия в порядке убывания количества заказанных на них билетов.  Составить алгоритм и программу, которые определяют суммарный спрос на имеющиеся в магазине товары каждого наименования и печатают список наименований в порядке убывания спроса. Известно, что в магазине имеются товары различных наименований, а в течение дня каждый из m покупателей (m – заданное число) сообщил о своем намерении приобрести определенное количество товара одного из наименований.  Каждая из m библиотек района (m - задано) составила заявку на приобретение книг. Заявка содержит перечень книг, состоящий из 10-ти наименований. Каждая библиотека в каждой строке заявки указывает название книги, фамилию автора, а также количество экземпляров, причем названия книг для разных библиотек могут повторяться. Составить алгоритм и программу, позволяющие определить суммарный запрос на каждую заказанную книгу, и напечатать общий список книг в порядке убывания спроса.  Имеются сведения об n рейсах Аэрофлота: номер рейса, пункт назначения и количество перевезенных пассажиров. Составить алгоритм и программу, которые определяют количество пунктов назначения и формируют список номеров рейсов для самого популярного пункта назначения в порядке уменьшения числа пассажиров, перевезенных этими рейсами. |

#### **Содержание отчета**

1. Тема и название работы.
2. Задание на разработку проекта и вариант задания.
3. Формализация задания и геометрическая иллюстрация решения

задачи.

1. Разработка приложения:
2. Графический интерфейс пользователя;
3. Таблица свойств объектов;
4. Схемы алгоритмов функциональных задач;
5. Программный код проекта с использованием процедур

функциональных задач и стандартных модулей.

1. Результаты выполнения приложения.
2. Обоснование правильности полученных результатов.

#### **Примеры выполнения задания**

1. Тема и название лабораторной работы.

Стандартные модули и модульная структура приложений в VB.

Формирование списка фамилий студентов, которым по итогам сессии

необходимо повысить успеваемость до фактического среднего уровня.

1. Задание на разработку проекта и вариант задания:

* Разработайте программный проект для решения следующей задачи: Для каждого из **n** студентов группы известны оценки по пяти экзаменам. Составить алгоритм и программу формирования списка фамилий студентов, которым по итогам сессии необходимо повысить успеваемость до фактического среднего уровня (т.е. включить в список тех студентов, у которых средний балл за сессию ниже общего среднего балла группы).
* Исходные списки студентов и результатов сессии могут находиться как в текстовом файле, так, и введены с клавиатуры.
* Пользовательский интерфейс разрабатываемого приложения для решения поставленной задачи должен содержать как минимум две формы (для ввода и отображения на форме исходных данных, и для вывода результатов на другую форму).
* Реализация программного кода должна использовать как минимум две формы и стандартный модуль.

1. Формализация задания:

Для решения всех вариантов задач:

* определите входные данные:
* **n** – количество студентов в одной группе (**Integer**);
* **m** – количество экзаменов в одной группе (**Integer**);
* **fam(n-1**) – одномерный массив фамилий студентов одной группы (**String**);
* **oc(n-1,m-1)** – двумерный массив оценок студентов одной группы (**Integer**);
* определите выходные данные:
* средние баллы за сессию каждого из **n** студентов одной группы – одномерный массив **so(n-1)** (**Double**);
* общий средний балл группы **sr** (**Double**);
* фамилии и средние баллы студентов, для которых средняя оценка **so(i)** меньше средней по группе **sr** - одномерный массив **f(k-1)** (**String**) и одномерный массив **sb(k-1)**  
  (**Double**), а также число таких студентов **k**;
* вычислите:
* **sr** - средний балл за сессию студентов одной группы; для получения среднего балла следует накопить сумму баллов, полученных n студентами за m экзаменов, а затем разделить эту сумму на количество оценок **m\*n** (**Double**);
* **so(n-1)** - средний балл за сессию каждого из n студентов одной группы; для получения каждого **i**-го элемента следует накопить сумму баллов, полученных **i**-м студентом за m экзаменов, а затем разделить эту сумму на **m** (**Double**);
* отсортируйте:

баллы (массив **sb(k-1)**) и соответственно фамилии студентов (массив

**f(k-1)**) в порядке убывания;

Геометрическая иллюстрация решения задачи приведена на рис. 1.5-1.

**Входные данные:**

**Вводимые Рассчитываемые**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Массив**  **фамилий**  **fam()**  0   |  | | --- | | Шакин | | Сосновиков | | Кравченко | | Загвоздкина | | Минязов | |  | |  |   n-1 | **Массив оценок**  **oc(,)**  0 m-1   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 2 | 3 | 3 | 4 | 2 | | 3 | 3 | 3 | 3 | 3 | |  |  |  |  |  | | 4 | 4 | 4 | 4 | 4 | | 4 | 5 | 4 | 5 | 4 | | 5 | 5 | 5 | 5 | 5 | |  |  |  |  |  |   n-1 | **Средние оценки каждого студента**  **so()**  0   |  | | --- | |  | |  | |  | |  | |  | |  | |  |   n-1 | **Средний балл группы**   |  | | --- | |  |   **Sr** |

##### **Выходные данные**:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Массив фамилий**  **студентов с низким**  **средним баллом**  **f()**  0   |  | | --- | |  | |  | |  | |  |   k-1 | **Средние баллы,**  **которые ниже общего  среднего**  **sb()**  0   |  | | --- | |  | |  | |  | |  |   k-1 |

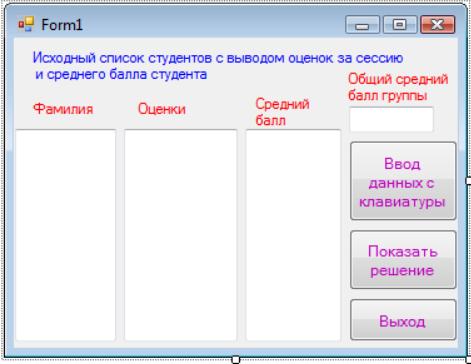
*Рис. 1.5-1. Геометрическая интерпретация   
(входные и выходные массивы) решения задачи*

***1 вариант реализации проекта:***

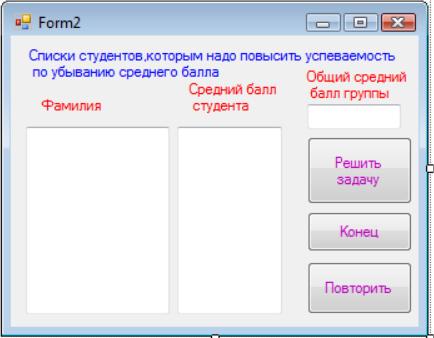
**В программном коде проекта Пример 1.5.4-1 используется стандартный модуль и две формы; исходные данные вводятся с клавиатуры и отображаются на 1-ой форме; результаты отображаются на 2-ой форме.**

1. Разработка проекта:
2. **Разработайте графический интерфейс пользователя:**

Формы интерфейса пользователя приведены на рис. 1.5-2 и 1.5-3.



*Рис. 1.5-2. Интерфейс пользователя* ***Form1****при вводе данных с клавиатуры проекта* ***Пример 1.5.4-1***



*Рис. 1.5-3. Интерфейс пользователя* ***Form2****при выводе данных проекта* ***Пример 1.5.4-1***

1. **Установите свойства объектов:**

Свойства объектов управления разработанных форм установите самостоятельно и отобразите в таблице.

1. **Разработайте схемы алгоритмов:**

В соответствии с заданием **1**-го варианта реализации проекта, решение задачи разбейте на следующие функциональные части:

* ввод количества студентов в группе – **n**;
* ввод количества экзаменов в группе – **m**;
* ввод исходных данных с клавиатуры – **vvod(n, m, fam, oc)**;
* отображение массивов данных (вещественного и строкового) и матрицы с оценками в текстовом поле на форме – процедуры   
  **vivodMas(), vivodMasStr()** и **vivodMatr();**
* нахождение **sr** - общего среднего балла за сессию студентов одной группы и **so**(**i**) - среднего балла каждого студента – **sred(n, m, oc, so, sr)**;
* решение задачи: подсчет количества **k** и формирование списка фамилий студентов **f()**, для которых средняя оценка меньше общей средней в группе, а также формирование массива **sb()** из их средних оценок – **FormirSpisok(n, sr, so, fam, f, sb, k)**;
* сортировка массива **sb()** в порядке убывания с дновременной перестановкой фамилий студентов – **sort(f ,sb).**

Исходные данные **n** - количество студентов, **m** – количество экзаменов, **fam()** - массив фамилий студентов, **oc(,)** – массив их оценок, и вычисленные из них значение общего среднего балла группы **sr** и массива средних оценок каждого студента **so()** должны быть доступны обеим формам проекта, поэтому эти данные объявляются как **Public** в стандартном модуле **Module1**. Там же в стандартном модуле **Module1** описываются процедуры, которые могут вызываться из обеих форм проекта, а именно: процедуры вывода массивов и матрицы, а также процедуры **sred()** подсчета средних оценок каждого студента и среднего балла всей группы. Результирующие списки студентов, которым надо повысить успеваемость (т.е. количество таких студентов **k**, массивы их фамилий **f()** и средних баллов **sb()**) нужны только для второй формы, поэтому объявляются на ней в соответствующей событийной процедуре.

Так как решить задачу можно только после ввода исходных данных, на первой форме кнопку для перехода на вторую форму **Button3** («Показать решение») сначала делаем недоступной (свойство Enable равно *False*). После ввода данных и расчета средних оценок, в конце событийной процедуры кнопки **Button1**, изменяем для кнопки **Button3** свойство Enable на *True*. Для отображения на форме исходных данных в процедурах **vodMas()**, **vivodMasStr()** и **vivodMatr()** формируется строка **z**, содержащая элементы соответствующих массивов. Эта строка должна возвращаться из процедур и ее значение выводится в нужные текстовые поля. Аналогичным образом происходит и отображение результатов на 2-ой форме.

Схемы алгоритмов отдельных функциональных задач разработайте самостоятельно.

1. **Разработайте программный код:**

Программный код проекта **Пример 1.5.4-1**, реализованный с использованием процедур, стандартного модуля и двух форм, представлен на рис. 1.5-4.

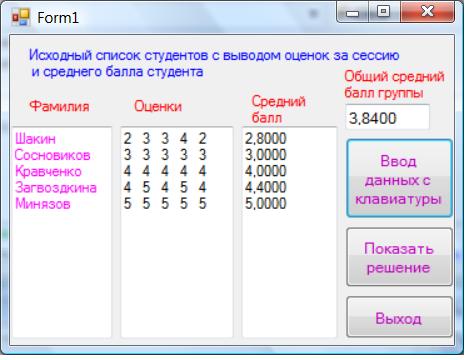
|  |  |
| --- | --- |
| **Option Strict On**  **Option Explicit On**  **Module Module1**  **Public n As Integer** *' Количество студентов*  **Public m As Integer** *' Количество экзаменов*  **Public oc(,) As Integer***' Оценки, полученные в сессию*  **Public fam() As String** *' Фамилии студентов*  **Public sr As Double** *' Общий средний балл группы за сессию*  **Public so() As Double** *' Средние баллы каждого студента* | |
| *'Процедура нахождения среднего балла каждого студента*  *'и общего среднего балла группы*  **Public Sub sred(ByVal n As Integer, ByVal m As Integer, \_**  **ByRef oc(,) As Integer, \_**  **ByRef so() As Double, ByRef sr As Double)**  **Dim i, j As Integer**  **Dim s As Double**  **sr = 0**  **For i = 0 To n - 1**  **s = 0**  **For j = 0 To m - 1**  **s = s + oc(i, j)**  **Next j**  **so(i) = s / m : sr = sr + s**  **Next i**  **sr = sr / (m \* n)**  **End Sub**  *'Процедура вывода строкового массива данных*  **Public Sub VivodMasStr(ByVal a() As String, ByRef z As String)**  **Dim i As Integer**  **z = ""**  **For i = 0 To UBound(a)**  **z = z + a(i) + vbNewLine**  **Next**  **End Sub**  *'Процедура вывода вещественного массива данных*  **Public Sub VivodMas(ByVal a() As Double, ByRef z As String)**  **Dim i As Integer**  **z = ""**  **For i = 0 To UBound(a)**  **z = z + Format(a(i), "0.0000") + vbNewLine**  **Next**  **End Sub**  *'Процедура вывода двумерного массива*  *'(собираем матрицу в одну строку)*  **Sub vivodMatr(ByRef x(,) As Integer, ByVal n As Integer, \_**  **ByVal m As Integer, ByRef z As String)**  **Dim i, j As Integer**  **z = ""**  **For i = 0 To n - 1**  **For j = 0 To m - 1**  **z = z + CStr(x(i, j)) & Space(3)**  **Next j**  **z = z + vbNewLine**  **Next i**  **End Sub**  **End Module**  *'На этой форме вводятся и выводятся исходные данные о студентах*  **Option Strict On**  **Public Class Form1**  *'Процедура ввода данных с клавиатуры*  **Public Sub vvod(ByVal n As Integer, ByVal m As Integer, \_**  **ByRef fam() As String, ByRef oc(,) As Integer)**  **Dim i, j As Integer**  **For i = 0 To n - 1**  **fam(i) = InputBox("Фамилия")**  **For j = 0 To m - 1**  **oc(i, j) = CInt(InputBox("Оценка"))**  **Next j**  **Next i**  **End Sub**  *'кнопка для ввода с клавиатуры*  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button1.Click**  **Dim s As String = ""**  **n = CInt(InputBox("Введите количество студентов"))**  **m = CInt(InputBox("Введите количество экзаменов"))**  **ReDim fam(n - 1), oc(n - 1, m - 1)**  *'вызов процедуры ввода данных с клавиатуры*  **vvod(n, m, fam, oc)**  **ReDim so(n - 1)**  *'вызов процедуры вычисления среднего балла каждого*  *'студента и общего среднего балла группы*  **sred(n, m, oc, so, sr)**  **VivodMasStr(fam, s)**  **TextBox1.Text = s**  **vivodMatr(oc, n, m, s)**  **TextBox2.Text = s**  **VivodMas(so, s)**  **TextBox3.Text = s**  **TextBox4.Text = Format(sr, "0.0000")**  *'Кнопку3 делаем доступной после ввода данных и расчета*  *'средних баллов*  **Button3.Enabled = True**  **End Sub**  *'Процедура для продолжения работы (перехода на 2-ю форму)*  **Private Sub Button3\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button3.Click**  **Form2.Show()**  **End Sub**    **Private Sub Button2\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button2.Click**  **End**  **End Sub**  **End Class**    **Public Class Form2**  *' Формирование массива f() фамилий студентов,*  *' имеющих средний балл ниже общего среднего*  *' и массива sb() средних баллов этих студентов*  **Public Sub FormirSpisok(ByVal n As Integer, \_**  **ByVal sr As Double, \_**  **ByVal so() As Double, \_**  **ByVal fam() As String, \_**  **ByRef f() As String, \_**  **ByRef sb() As Double, \_**  **ByRef k As Integer)**  **Dim i As Integer**  **k = 0**  **For i = 0 To n - 1**  **If so(i) < sr Then**  **f(k) = fam(i)**  **sb(k) = so(i)**  **k = k + 1**  **End If**  **Next**  **ReDim Preserve f(k - 1), sb(k - 1)**  **End Sub**  *'сортировка по убыванию среднего балла студента*  **Sub sort(ByRef s() As String, ByRef kol() As Double)**  **Dim i, j, k As Integer, temp As String, t As Double**  **k = s.Length - 1**  **For i = 0 To k - 1**  **For j = i + 1 To k**  **If kol(i) < kol(j) Then**  **t = kol(i) : kol(i) = kol(j) : kol(j) = t**  **temp = s(i) : s(i) = s(j) : s(j) = temp**  **End If**  **Next**  **Next**  **End Sub**  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs)Handles Button1.Click**  **Dim z As String = ""** *'строка для вывода массивов*  **Dim k As Integer '***кол-во студ, к-рым надо*  *'повысить успеваемость*  **Dim f(n - 1) As String** *'массив их фамилий*  **Dim sb(n - 1) As Double** *'массив их средних баллов*  **FormirSpisok(n, sr, so, fam, f, sb, k)**  **sort(f, sb)**  **VivodMasStr(f, z)**  **TextBox1.Text = z**  **VivodMas(sb, z)**  **TextBox2.Text = z**  **TextBox3.Text = Format(sr, "0.0000")**  **End Sub**  *' Кнопка для повторения*  **Private Sub Button4\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) Handles Button4.Click**  **Me.Close()**  **Form1.TextBox1.Text = ""**  **Form1.TextBox2.Text = ""**  **Form1.TextBox3.Text = ""**  **Form1.TextBox4.Text = ""**  **Form1.Button1.Enabled = True**  **Form1.Button2.Enabled = True**  **Form1.Button3.Enabled = False**  **Form1.Show()**  **End Sub**  **End Class** |

*Рис. 1.5-4. Программный код проекта* ***Пример 1.5.4-1*** *при вводе данных с клавиатуры*

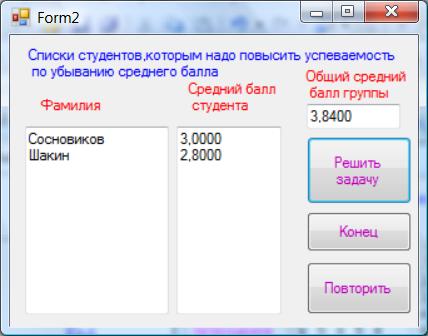
1. Выполните приложение и получите результаты:

Результаты работы проекта **(**вариант 1**)** представлены на рис. 1.5-5

и рис. 1.5-6.



*Рис. 1.5-5. Результаты работы проекта* ***Пример 1.5.4-1*** *при вводе данных с клавиатуры на* ***Form1***



*Рис. 1.5-6. Результаты работы проекта* ***Пример 1.5.4-1****при их выводе на* ***Form2***

1. Докажите правильность работы программы**:**

В результате работы проекта с клавиатуры введены исходные данные (массив фамилий студентов и матрица полученных в сессию оценок), которые использованы для подсчета средней оценки каждого студента и среднего балла всей группы и выведены на **Form1**. На **Form2** представлен сформированный и отсортированный по убыванию среднего балла список фамилий тех студентов, которым надо повысить успеваемость до фактического среднего уровня (у которых средняя оценка ниже общего среднего балла всей группы).

***2 вариант реализации проекта:***

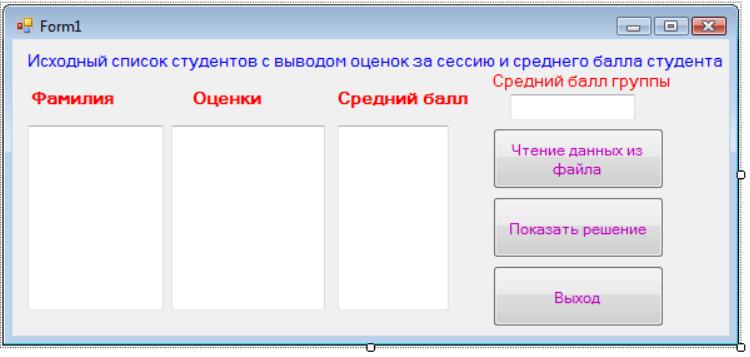
**В программном коде проекта Пример 1.5.4-2 используется стандартный модуль и две формы; исходные данные вводятся из текстового файла чтением построчно и отображаются на первой форме; результаты вычислений отображаются на второй форме и построчно записываются в другой текстовый файл.**

**Формат исходного файла: каждая строка файла состоит из фамилии и 5 оценок, разделенных между собой одним или несколькими пробелами, например,**

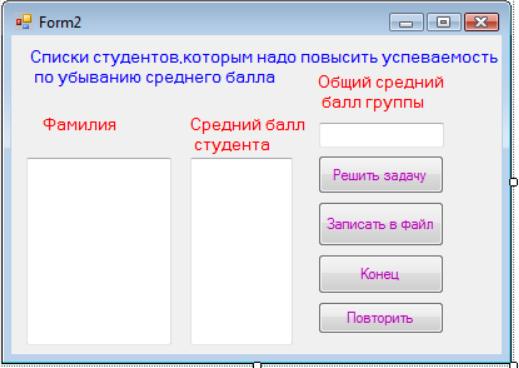
**Фамилия 5 4 5 3 4.**

1. Разработка проекта:
   1. **Разработайте графический интерфейс пользователя:**

Разработанные формы интерфейса приведены на рис. 1.5-7 и   
 рис. 1.5-8.

****

*Рис. 1.5-7. Интерфейс пользователя* ***Form1****при вводе данных из файла проекта* ***Пример 1.5.4-2***



*Рис. 1.5-8. Интерфейс пользователя* ***Form2****при выводе данных проекта* ***Пример 1.5.4-2***

* 1. **Установите свойства объектов:**

Свойства объектов управления разработанных форм установите самостоятельно и отобразите в таблице.

* 1. **Разработайте схемы алгоритмов:**

В соответствии с заданием ***2-го*** варианта реализации проекта, решение задачи состоит из функциональных задач:

* + построчное чтение исходных данных из текстового файла и определение числа студентов в группе – **readfile("путь1", n)**;
  + определение количества экзаменов и создание массивов фамилий студентов **fam()** и оценок **oc()** из читаемого файла –   
    **vvod("путь1", n, m, fam, oc)**;
  + отображение массивов данных (вещественного и строкового) и матрицы с оценками в текстовом поле на форме – процедуры **vivodMas(), vivodMasStr()** и **vivodMatr()**;
  + нахождение **sr** - общего среднего балла за сессию студентов одной группы и **so(i)**- среднего балла каждого студента –   
    **sred(n, m, oc, so, sr)**;
  + подсчет количества **k** и формирование списка фамилий студентов **f()**, для которых средняя оценка меньше общей средней в группе, а также формирование массива **sb()** из их средних оценок **FormirSpisok(n, sr, so, fam, f, sb, k)**;
  + сортировка массива **sb()** в порядке убывания с одновременной перестановкой фамилий студентов – **sort(f ,sb)**;
  + построчная запись результирующих массивов **f()** и **sb()** в текстовый файл – **writefile("путь2",f, sb)**.

Исходные данные **n** – количество студентов, **m** - количество экзаменов, **fam()** - массив фамилий студентов, **oc(,)** – массив их оценок, и вычисленные из них значение общего среднего балла группы sr и массива средних оценок каждого студента **so()** должны быть доступны обеим формам проекта, поэтому эти данные объявляются как **Public** в стандартном модуле **Module1**. Там же в стандартном модуле **Module1** описываются процедуры, которые могут вызываться из обеих форм проекта, а именно: процедуры вывода массивов и матрицы. Результирующие списки студентов, которым надо повысить успеваемость (т.е. количество таких студентов **k**, массивы их фамилий **f()** и средних баллов **sb()**) нужны только для второй формы, но для двух кнопок (событийных процедур), поэтому объявляются на второй форме, как глобальные для нее (перед всеми остальными процедурами).

Так как решить задачу можно только после ввода исходных данных, на первой форме кнопку для перехода на вторую форму **Button3** («Показать решение») сначала делаем недоступной (**свойство** Enableравно*False*). После ввода данных и расчета средних оценок, в конце событийной процедуры для кнопки **Button1**, изменяем для кнопки **Button3** свойство Enable на *True*. Для отображения на форме массивов в процедурах **vivodMas(), vivodMasStr()** и **vivodMatr()** формируется строка **z**, содержащая элементы соответствующих массивов. Эта строка должна возвращаться из процедур и ее значение выводится в нужные текстовые поля.

Схемы алгоритмов отдельных функциональных задач разработайте самостоятельно.

* 1. **Напишите программный код с использованием модуля и**

**процедур:**

Программный кодпроекта **Пример 1.5.4-2**, реализованный с использованием процедур, стандартного модуля и двух форм, представлен на рис. 1.5-9. При работе с текстовым файлом (чтении и записи данных) используется блок перехвата и обработки исключений **Try**, который подробно описывается в **Теме 4**, а также потоковый ввод-вывод, описанный в [1].

Обратите внимание, что программный код стандартного модуля, а также коды процедур **FormirSpisok()** и **Sort()**, которые описываются в **Form2**, представленные на рис. 1.5-4, абсолютно одинаковы для всех трех реализаций задания и их не будем повторять, а лишь обозначим заголовками.

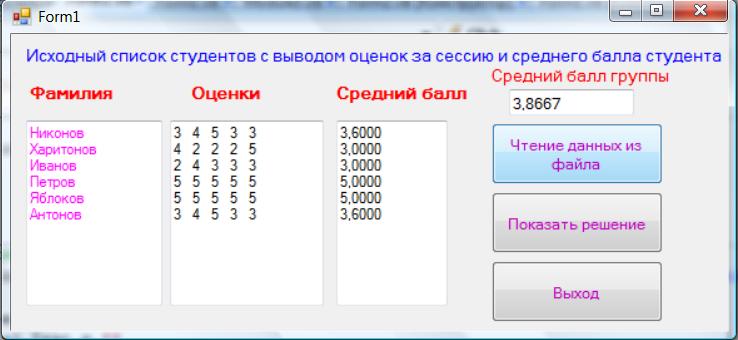
|  |
| --- |
| **Option Strict On**  **Option Explicit On**  **Module Module1**  *'…Содержание Module1 одинаково для всех релизаций (см. рис.1.5-4)*  **End Module**  *'на 1-й форме вводятся и выводятся исходные данные о студентах*  **Option Strict On**  **Imports System.IO**  **Public Class Form1**  *'Процедура построчного чтения сведений о студентах из тек. файла,*  *'для подсчета числа студентов*  **Sub readfile(ByVal filename As String, ByRef n As Integer)**  *'создание потока для чтения из текстового файла*  **Dim f = New StreamReader(filename)**  **Dim line As String**  **n = 0** *'количество строк в файле*  **Try**  **Do While Not (f.EndOfStream)***'пока не достигнут конец потока*  **line = f.ReadLine()** *'чтение строки из файла в line*  **n = n + 1**  **Loop**  **f.Close()** *'закрытие потока*  **Catch ex As Exception**  **MsgBox("Невозможно прочитать данные из файла")**  **End Try**  **End Sub**  *'Процедура построчного считывания сведений о студентах из*  *'файла и создание массива фамилий студентов и массива оценок*  **Sub vvod(ByVal filename As String,ByVal n As Integer, \_**  **ByRef m As Integer, ByRef fam() As String, \_**  **ByRef oc(,) As Integer)**  **Dim i, j As Integer**  **Dim line As String**  **Dim buf() As String**  *'создание потока для чтения из текстового файла*  **Dim f As New StreamReader(filename)**  **Try** *'номер читаемой строки в файле*  **i = 0** *'пока не достигнут конец потока*  **Do While Not (f.EndOfStream)**  **line = f.ReadLine()** *'чтение строки из файла в line*  *'Удаление из строки возможных лишних пробелов*  *'(цикл замены двух пробелов одним)*  **Do While line.IndexOf(Space(2)) >= 0**  **line = line.Replace(Space(2), Space(1))**  **Loop**  **If line = "" Then**  **n = n - 1**  **MsgBox("В файле есть пустая строка")**  **Exit Sub**  **Else**  *'Ф-ция Split разбивает строку line на массив*  *'подстрок buf()по символам пробела*  **buf = Split(line, Space(1))**  *'***=================================================================**  *' При обработке первой строки файла(при i = 0)определяем*  *' кол-во экзаменов m и выделяем память под массив оценок oc(,).*  *' В каждой строке файла(массиве**buf)записана фамилия студента*  *' и m оценок, поэтому кол-во экзаменов m равно длине массива buf*  *'минус единица (фамилия студента)*  **If i = 0 Then**  *'кол-во элементов в массиве buf() равно количеству экзаменов m*  *'минус единица (фамилия студента)*  **m = buf.Length - 1**  **ReDim oc(n - 1, m - 1)**  **End If '==================================================================**  *'Первый элемент этого массива buf()-фамилия,следующие-оценки*  **fam(i) = buf(0)**  **For j = 0 To m - 1**  **oc(i, j) = CInt(buf(j + 1))**  **Next**  **End If**  **i = i + 1**  **Loop**  **f.Close()**  **Catch ex As Exception**  **MsgBox("Невозможно прочитать данные из файла")**  **End Try**  **n = i**  **End Sub**  *'кнопка для ввода из файла*  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button1.Click**  **Dim s As String = ""**  *'Вызов проц-ры построчного чтения из текстового файла*  *'для определения количества студентов n*  **readfile("myfile.txt", n)**  **ReDim fam(n - 1)**  *'вызов процедуры ввода группы из файла(создание массивов*  *'фамилий студентов и массива оценок)*  **vvod("myfile.txt", n, m, fam, oc)**  **ReDim so(n - 1)**  *'вызов процедуры вычисления среднего балла каждого*  *'студента и общего среднего балла группы*  **sred(n, m, oc, so, sr)**  *'вывод данных на форму*  **VivodMasStr(fam, s) : TextBox1.Text = s**  **vivodMatr(oc, n, m, s)**  **TextBox2.Text = s**  **VivodMas(so, s)**  **TextBox3.Text = s**  **TextBox4.Text = Format(sr, "0.0000")**  *'Кнопку3 делаем доступной после ввода данных*  *'и расчета средних баллов*  **Button3.Enabled = True**  **End Sub**  *'кнопка для перехода на 2-ю форму*  **Private Sub Button3\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button3.Click**  **Form2.Show()**  **End Sub**  *'Процедура для окончания работы программы*  **Private Sub Button2\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button2.Click**  **End**  **End Sub**  **End Class**  *'***=================================================================**  *'Программный код 2 формы*  **Option Strict On**  **Imports System.IO**  **Public Class Form2**  *'Объекты k,f(n-1)и sb(n-1)объявляем глобальными для 2-й формы,*  *'т.к. они должны быть доступны сразу двум событийным процедурам*  *'(для кнопок "Решить задачу" и "Записать в файл")*  **Dim k As Integer** *'кол-во студ, к-рым надо повысить успеваемость*  **Dim f(n - 1) As String** *'массив их фамилий*  **Dim sb(n - 1) As Double** *'массив их средних баллов*  *' Формирование массива f() фамилий студентов,*  *' имеющих средний балл ниже общего среднего*  *' и массива sb() средних баллов этих студентов*  **Public Sub FormirSpisok(ByVal n As Integer, \_**  **ByVal sr As Double, ByVal so()As Double, \_**  **ByVal fam() As String, ByRef f()As String, \_**  **ByRef sb() As Double, ByRef k As Integer)**  **. . .** *' (см. рис.1.5-4)*  **End Sub**  *'сортировка по убыванию среднего балла студента*  **Sub sort(ByRef s() As String, ByRef kol() As Double)**  **. . .** *' (см. рис.1.5-4)*  **End Sub**  *'Процедура построчной записи в текстовый файл отсортированного*  *' массива студентов,имеющих средний балл ниже общего среднего*  **Sub writefile(ByVal filename As String, ­**  **ByVal f() As String, \_**  **ByVal sb() As Double)**  *'создание нового текст. файла и потока wf для записи в файл*  **Dim wf As StreamWriter = File.CreateText(filename)**  **Dim s As String** *'строка для записи в файл*  **Dim i, k As Integer**  **k = f.Length - 1**  **For i = 0 To k**  **s = fam(i) + " " + CStr(sb(i))**  *'запись сведений об одном студенте в текстовый файл*  **wf.WriteLine(s)**  **Next**  **wf.Flush() : wf.Close()** *'очистка и закрытие потока*  **End Sub**  *'кнопка "Решить задачу"*  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button1.Click**  **Dim z As String = ""** *'строка для вывода массивов*  **FormirSpisok(n, sr, so, fam, f, sb, k)**  **sort(f, sb)**  **VivodMasStr(f, z)**  **TextBox1.Text = z**  **VivodMas(sb, z)**  **TextBox2.Text = z**  **TextBox3.Text = Format(sr, "0.0000")**  **End Sub**  *'кнопка "Записать в файл"*  **Private Sub Button2\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs)\_**  **Handles Button2.Click**  **writefile("file2.txt", f, sb)**  **End Sub**  *'Процедура для окончания работы программы*  **Private Sub Button3\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button3.Click**  **End**  **End Sub**  *'кнопка "Повторить"*  **Private Sub Button4\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button4.Click**  **Me.Close()**  **Form1.TextBox1.Text = "" : Form1.TextBox2.Text = ""**  **Form1.TextBox3.Text = "" : Form1.TextBox4.Text = ""**  **Form1.Button1.Enabled = True**  **Form1.Button2.Enabled = True**  **Form1.Button3.Enabled = False**  **Form1.Show()**  **End Sub**  **End Class** |

*Рис. 1.5-9. Сокращенный программный код проекта* ***Пример 1.5.4-2****при построчном чтении данных из файла*

*и построчной записи результатов в файл*

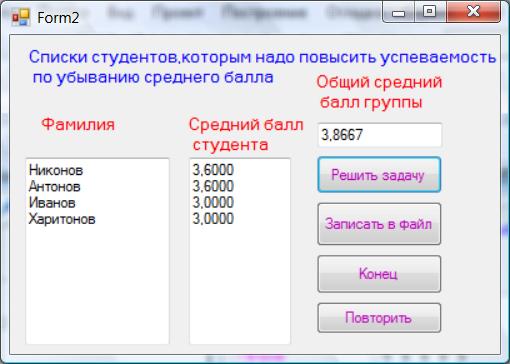
1. Выполните приложение и получите результаты:

Результаты работы проекта **Пример 1.5.4-2 (вариант 2)** представлены на рис. 1.5-10 и рис. 1.5-11.



*Рис. 1.5-10. Результаты работы проекта* ***Пример 1.5.4-2*** *при построчном чтении данных из файла*

*и отображение их на* ***Form1***



*Рис. 1.5-11. Результаты работы проекта* ***Пример 1.5.4-2*** *при выводе результатов на* ***Form2***

1. Докажите правильность работы программ.

В результате работы проекта из файла построчно прочитаны исходные данные и корректно преобразованы в массивы, которые использованы для подсчета средней оценки каждого студента и среднего балла всей группы и выведены на 1 форму. На второй форме представлены полученные и отсортированные по убыванию среднего балла массив фамилий и массив среднего балла тех студентов, которым надо повысить успеваемость до фактического среднего уровня (у которых средняя оценка ниже общего среднего балла всей группы). Полученные результаты построчно записаны в другой текстовый файл.

***3 вариант реализации проекта:***

**В программном коде проекта Пример 1.5.4-3 используется стандартный модуль и две формы; исходные данные вводятся из текстового файла чтением целиком в одну строку и отображаются на первой форме; результаты вычислений отображаются на второй форме и формируются в одну строку, которая записывается в другой текстовый файл.**

**Формат исходного файла: допускается разделение данных в строке несколькими пробелами, т.е. может иметь формат вида:  
 фамилия 5 4 5 3 4.**

1. Разработка проекта:
2. **Разработайте графический интерфейса пользователя:**

Разработанные формы пользовательского интерфейса полностью совпадают с формами проекта ***Пример 1.5.4-2*** и приведены на рис. 1.5-7 и рис. 1.5-8.

1. **Установите свойства объектов:**

Свойства объектов управления разработанных форм студенты устанавливают самостоятельно и отображают в таблице.

1. **Разработайте схемы алгоритмов:**

В соответствии с заданием **3-го** варианта реализации проекта,

решение задачи можно разбить на следующие функциональные

задачи:

* ввод исходных данных из текстового файла одной строкой – **vvodFile("путь1", s);**
* получение из этой строки массива подстрок mas() с помощью функции **Split()** и определение количества студентов **n**;
* определение количества экзаменов и создание массива фамилий студентов **fam()** и массива оценок **oc()** из полученного массива **mas()** подстрок файла – **vvod(mas, n, m, fam, oc)**;
* отображение массивов данных (вещественного и строкового) и матрицы с оценками в текстовом поле на форме – процедуры   
  **vivodMas()**, **vivodMasStr()** и **vivodMatr();**
* нахождение **sr** - общего среднего балла за сессию студентов одной группы и **so(i**) - среднего балла каждого студента – **sred(n, m, oc, so, sr);**
* подсчет количества **k** и формирование списка фамилий студентов f**()**, для которых средняя оценка меньше общей средней в группе, а также формирование массива **sb()** из их средних оценок – **FormirSpisok(n, sr, so, fam, f, sb, k)**;
* сортировка массива **sb()** в порядке убывания с одновременной перестановкой фамилий студентов – **sort(f ,sb);**
* формирование строки **S** из результирующих массивов **f()** и **sb()** – функция **res(f, sb, k);**
* запись строки-результата **S** в файл – **vivod("путь2", S)**.

Исходные данные **n** - количество студентов, **m** - количество экзаменов,**fam()** - массив фамилий студентов, **oc(,)** – массив их оценок, и вычисленные из них значение общего среднего балла группы **sr** и массива средних оценок каждого студента **so()** должны быть доступны обеим формам проекта, поэтому эти данные объявляются как **Public** в стандартном модуле **Module1**. Там же в стандартном модуле **Module1** описываются процедуры, которые могут вызываться из обеих форм проекта, а именно: процедуры вывода массивов и матрицы, и процедура подсчета баллов. Результирующие списки студентов, которым надо повысить успеваемость (т.е. количество таких студентов **k**, массивы их фамилий **f()** и средних баллов **sb())** нужны только для второй формы, но для двух кнопок (событийных процедур), поэтому объявляются на второй форме, как глобальные для нее (перед всеми остальными процедурами).

Так как решить задачу можно только после ввода исходных данных, на первой форме кнопку для перехода на вторую форму   
**Button3** (Показать решение) сначала делаем недоступной (свойство Enable = *False*). После ввода данных и расчета средних оценок, в конце событийной процедуры для кнопки **Button1**, изменяем для кнопки **Button3** свойство Enable на *True*. Для отображения на форме массивов в процедурах **vivodMas()**, **vivodMasStr()** и **vivodMatr()** формируется строка **z**, содержащая элементы соответствующих массивов. Эта строка должна возвращаться из процедур и ее значение выводится в нужные текстовые поля.

Общую схему алгоритма решения задачи и схемы алгоритмов отдельных функциональных задач разработайте самостоятельно.

1. **Написание программного кода с использованием модуля и**

**процедур.**

Программный код проекта, представлен на рис. 1.5-12.

При работе с текстовым файлом (чтении и записи данных) используется блок перехвата и обработки исключений **Try**, который подробно описывается в **Теме 4**, а также ввод-вывод целиком одной строкой с помощью методов объекта **My.Computer.FileSystem**, которые описаны в [1].

Обратите внимание, что программный код стандартного модуля, а также коды процедур **FormirSpisok()** и **Sort()**, которые описываются в **Form2**, представленные на рис. 1.5-4, абсолютно одинаковы для всех трех реализаций задания и их не будем повторять, а лишь обозначим заголовками.

|  |
| --- |
| **Option Strict On**  **Option Explicit On**  **Module Module1**  **. . .*****'*** *(см. рис.1.5-4)*  **End Module**  *'на 1-й форме вводятся и выводятся исходные данные о студентах*  **Option Strict On**  **Public Class Form1**  *'Процедура чтения сведений о студентах из текстового файла в строку*  **Sub vvodFile(ByVal filename As String, ByRef S As String)**  *'Открываем существующий файл и загружаем его текстовое*  *'содержимое целиком в одну строку S.*  **Try**  **S = My.Computer.FileSystem.ReadAllText(filename)**  **Catch ex As Exception**  **MsgBox("ошибка при чтении файла")**  **End Try**  **End Sub**  *'Процедура ввода данных из файла(создание массива фамилий*  *'студентов и массива оценок из массива строк файла)*  **Public Sub vvod(ByVal mas() As String, ByVal n As Integer, \_**  **ByRef m As Integer, ByRef fam() As String, \_**  **ByRef oc(,) As Integer)**  **Dim i, j As Integer**  **Dim line, buf() As String**  **For i = 0 To n - 1**  **line = mas(i)**  *'Удаление из строки возможных лишних пробелов*  *'(цикл замены двух пробелов одним)*  **Do While line.IndexOf(Space(2)) >= 0**  **line = line.Replace(Space(2), Space(1))**  **Loop**  **If mas(i) = "" Then**  **n = n - 1**  **MsgBox("В файле есть пустая строка")**  **Exit For**  **Else**  *'Ф-ция Split разбивает строку line на массив*  *'подстрок buf()по символам пробела*  **buf = Split(line, Space(1))**  **==============================================================**  *' При обработке первой строки файла(при i = 0)определяем*  *' кол-во экзаменов m и выделяем память под массив оценок oc(,)*  *' В каждой строке файла(массиве**buf)записана фамилия студента*  *' и m оценок, поэтому кол-во экзаменов m равно длине массива buf*  *'минус единица (фамилия студента)*  **If i = 0 Then**  *'длина массива buf() - кол-во экз-нов минус один элемент (фамилия)*  **m = buf.Length - 1**  **ReDim oc(n - 1, m - 1)**  **End If ==============================================================**  *'Первый элемент этого массива-фамилия,следующие-оценки*  **fam(i) = buf(0)**  **For j = 0 To m - 1**  **oc(i, j) = CInt(buf(j + 1))**  **Next**  **End If**  **Next**  **End Sub**  *'кнопка для ввода исходных данных из файла и*  *'создания массивов исходных данных*  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button1.Click**  **Dim s As String = ""**  *'Вызов проц-ры чтения сведений о студентах из текстового*  *'файла целиком в одну строку s*  **vvodFile("myfile.txt", s)**  **Dim mas() As String**  *'Ф-ция Split разбивает строку s на массив подстрок mas()*  *'по символам конца строки vbNewLine*  **mas = Split(s, vbNewLine)**  *'кол-во эл-тов получ. массива(кол-во студентов)*  **n = mas.Length** *'По полученному количеству студентов n*  *'переопределяем размер массивов*  **ReDim fam(n - 1), so(n - 1)**  *'вызов процедуры ввода группы из файла(создание массивов*  *'фамилий студентов и массива оценок)*  **vvod(mas, n, m, fam, oc)**  *'вызов процедуры вычисления среднего балла каждого студ.*  *'и общего среднего балла группы*  **sred(n, m, oc, so, sr)**  *'Вывод полученных массивов на форму*  **VivodMasStr(fam, s)**  **TextBox1.Text = s**  **vivodMatr(oc, n, m, s)**  **TextBox2.Text = s**  **VivodMas(so, s)**  **TextBox3.Text = s**  **TextBox4.Text = Format(sr, "0.0000")**  *'Кнопку3 делаем доступной после ввода данных*  *' и расчета средних баллов*  **Button3.Enabled = True**  **End Sub**  *'кнопка для перехода на 2-ю форму*  **Private Sub Button3\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button3.Click**  **Form2.Show()**  **End Sub**  **Private Sub Button2\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button2.Click**  **End**  **End Sub**  **End Class**  *'Программный код 2 формы*  **Option Strict On**  **Public Class Form2**  *'Объекты k,f(n-1)и sb(n-1)объявляем глобальными для 2-й формы,*  *'т.к. они должны быть доступны сразу двум событийным процедурам*  *'(для кнопок "Решить задачу" и "Записать в файл")*  *'кол-во студ, к-рым надо повысить успеваемость*  **Dim k As Integer**  **Dim f(n - 1) As String** *'массив их фамилий*  **Dim sb(n - 1) As Double** *'массив их средних баллов*  *' Формирование массива f() фамилий студентов,*  *' имеющих средний балл ниже общего среднего*  *' и массива sb() средних баллов этих студентов*  **Public Sub FormirSpisok(ByVal n As Integer, \_**  **ByVal sr As Double, \_**  **ByVal so() As Double, \_**  **ByVal fam() As String,\_**  **ByRef f() As String, \_**  **ByRef sb() As Double, \_**  **ByRef k As Integer)**  **. . .** *'**(см. рис.1.5-4)*  **End Sub**  *'сортировка по убыванию среднего балла студента*  **Sub sort(ByRef s() As String, ByRef kol() As Double)**  **. . .*****'*** *(см. рис.1.5-4)*  **End Sub**  *'Функция формирования одной строки из отсортированного массива*  *'студентов,имеющих средний балл ниже общего среднего*  **Function res(ByVal f() As String, ByVal sb() As Double, \_**  **ByVal k As Integer) As String**  *'Создание объекта result типа StringBuilder(результ. строки)*  *'для очень быстрой и эффективной обработки строковых данных*  **Dim result As New System.Text.StringBuilder**  **For i As Integer = 0 To k - 1**  **result.Append(f(i))**  *'добавление фамилии в объект result*  **result.Append(Space(1))** *'добавление пробела*  *'добавление среднего балла*  **result.Append(sb(i).ToString)**  *'добавление признака конца строки*  **result.Append(vbNewLine)**  **Next i**  **Return result.ToString**  **End Function**  *' Процедура записи результата в текстовый файл*  **Sub vivodFile(ByVal filename As String, ByRef S As String)**  *'Запись строки-результата S в текстовый файл;значение False*  *'третьего параметра задает,что файл будет перезаписан.*  **Try**  **My.Computer.FileSystem.WriteAllText(filename, S, False)**  **Catch ex As Exception**  **MsgBox("ошибка при записи в файл")**  **End Try**  **End Sub**  *'кнопка "Решить задачу"*  **Private Sub Button1\_Click\_1(ByVal sender As System.Object,\_**  **ByVal e As System.EventArgs) \_**  **Handles Button1.Click**  **Dim z As String = ""** *'строка для вывода массивов*  **FormirSpisok(n, sr, so, fam, f, sb, k)**  **sort(f, sb)**  **VivodMasStr(f, z)**  **TextBox1.Text = z**  **VivodMas(sb, z)**  **TextBox2.Text = z**  **TextBox3.Text = Format(sr, "0.0000")**  **End Sub**  *'кнопка "Записать в файл"*  **Private Sub Button2\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button2.Click**  **Dim S As String = ""**  *'Вызов функции формирования одной строки из отсортированного*  *'массива студентов,имеющих средний балл ниже общего среднего*  **S = res(f, sb, k)**  *'Вызов процедуры записи результата в текстовый файл*  **vivodFile("file2.txt", S)**  **End Sub**  **Private Sub Button3\_Click\_1(ByVal sender As System.Object,\_**  **ByVal e As System.EventArgs) \_**  **Handles Button3.Click**  **End**  **End Sub**  *'кнопка "Повторить"*  **Private Sub Button4\_Click\_1(ByVal sender As System.Object,\_**  **ByVal e As System.EventArgs) \_**  **Handles Button4.Click**  **Me.Close()**  **Form1.TextBox1.Text = ""**  **Form1.TextBox2.Text = ""**  **Form1.TextBox3.Text = ""**  **Form1.TextBox4.Text = ""**  **Form1.Button1.Enabled = True**  **Form1.Button2.Enabled = True**  **Form1.Button3.Enabled = False**  **Form1.Show()**  **End Sub**  **End Class** |

*Рис. 1.5-12. Сокращенный программный проекта* ***Пример 1.5.4-****3   
при чтении данных из файла целиком в одну строку*

1. Выполните приложение и получите результаты:

Результаты работы проекта Пример 1.5.4-3 полностью совпадают с результатами проекта Пример 1.5.4-2 и представлены на рис. 1.5-10 и рис. 1.5-11.

1. Обоснуйте правильность работы программы.

В результате работы проекта из файла введены исходные данные чтением целиком в одну строку. Прочитанная строка корректно преобразована в массивы, которые использованы для подсчета средней оценки каждого студента и среднего балла всей группы и выведены на 1 форму. На второй форме представлены полученные и отсортированные по убыванию среднего балла массив фамилий и массив среднего балла тех студентов, которым надо повысить успеваемость до фактического среднего уровня (у которых средняя оценка ниже общего среднего балла всей группы). Полученные результаты сформированы в одну строку, которая целиком записана в другой текстовый файл.

### 1.6. Контрольные вопросы

1. Какова структура Windows-приложения?
2. Что такое модуль?
3. Что такое стандартный модуль?
4. Что необходимо для создания проекта с несколькими формами и организации взаимодействия между ними?
5. Какие шаблоны проектов используются в среде VS .NET?
6. В чем заключается главное отличие проекта библиотеки класса от консольного и Windows – проектов?
7. Что такое пространство имен? Назначение пространства имен.
8. Из чего состоит каркас проекта VS?
9. Что такое Сборка в VS? Состав и назначение Сборки.
10. Что такое Решение в VS?
11. Что такое проект в VS?
12. Что такое шаблон проекта в VS?
13. Какие основные проекты вы знаете?
14. Как создать Консольное приложение?
15. Как создать Приложение Windows Forms?
16. Как создать DLL библиотеку классов?
17. Что является точкой входа в различных проектах?
18. Что такое Свойство в VS?
19. Что такое глобальные переменные в VB?
20. Что такое локальные переменные в VB?