МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВПО «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ»

А.А. Казанский

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ НА VISUAL BASIC 2010 И VISUAL C# 2010 В СРЕДЕ РАЗРАБОТКИ MICROSOFT VISUAL STUDIO

Учебное пособие и практикум

Москва 2012

УДК 681.3(07)
ББК 32.973.26я73
К 14

Р е ц е н з е н т ы:

доктор технических наук *В.В. Серов*
(ФГБОУ ВПО «МГУТУ им. К.Г. Разумовского»);
кандидат технических наук, доцент *В.С. Варников*
(ФГБОУ ВПО «МГСУ»)

Казанский, А.А.

К 14 Объектно-ориентированное программирование на Visual Basic 2010 и Visual C# 2010 в среде разработки Microsoft Visual Studio : учебное пособие и практикум / А.А. Казанский ; М-во образования и науки Росс. Федерации, ФГБОУ ВПО «Моск. гос. строит. ун-т». – Москва : МГСУ, 2012. – 424 с.

ISBN 978-5-7264-0660-2

Предложена методика последовательного освоения современного объектно-ориентированного программирования, рассчитанная на применение методов на практике. Приведены теоретические сведения, необходимые для выполнения заданий. Дано подробное описание лабораторных работ, примеры выполнения работ, вопросы для самопроверки и варианты индивидуальных занятий по каждой теме. Рассмотрена объектная модель, дающая возможность описывать как элементы управления графического интерфейса (командные кнопки, текстовые поля, веб-браузеры), так и реальные объекты (мосты, дороги, студенческие группы и т.д.).

Для студентов, обучающихся по специальностям 203102, 230104, 220301, 080502 (дисциплина «Информатика») и по специальностям 231300, 151600 (дисциплина «Алгоритмические языки и программирование»).

УДК 681.3(07)
ББК 32.973.26я73

ISBN 978-5-7264-0660-2

Введение

Visual Studio является средой разработки, основанной на компонентах и технологиях для создания современных компьютерных приложений. Эта среда эффективно используется при разработке научных, промышленных и экономических проектов. Первая версия Visual Studio была выпущена компанией Microsoft в 1995 г., а уже в 2002 г. появилась и программная платформа .NET Framework.

Языки программирования Visual Basic 2010 и Visual C# 2010 используют общую интегрированную среду разработки (IDE), которая упрощает создание программных продуктов. Кроме того, в этих языках доступны функциональные возможности платформы .NET Framework, которая позволяет получить доступ к ключевым технологиям, упрощающим разработку веб-приложений ASP и XM, а также настольных и мобильных приложений. Эта среда непрерывно совершенствуется.

Основой платформы .NET Framework является общеязыковая среда CLR, которая управляет кодом во время выполнения и предоставляет основные службы, такие как управление памятью, управление потоками, удаленное взаимодействие, а также обеспечивает безопасность и надежность. Код, который обращается к среде выполнения, называют управляемым, а код, который не обращается к среде выполнения — неуправляемым. Вторая часть платформы .NET Framework — библиотека классов представляет объектно-ориентированную коллекцию типов, которые применяются для разработки приложений, начиная от обычных, запускаемых из командной строки или с графическим интерфейсом пользователя, и заканчивая приложениями, использующими последние технологические возможности ASP.NET, такими как Web Forms и веб-службы XML. Типы .NET Framework позволяют решать ключевые задачи программирования, включая работу со строками, сбор данных, подключения к базам данных и доступ к файлам. Кроме этого, библиотека классов содержит типы, поддерживающие многие специализированные сценарии разработки. Обычно платформа .NET Framework применяется для разработки следующих типов приложений и служб:

- приложения с графическим интерфейсом пользователя Windows (Windows Forms);
- приложения Windows Presentation Foundation (WPF);
- приложения ASP.NET;
- веб-службы;
- службы Windows;
- консольные приложения;
- сервисно ориентированные приложения с помощью Windows Communication Foundation (WCF);
- приложения, поддерживающие бизнес-процессы Windows Workflow Foundation (WF).

Компания Microsoft разработала новую идеологию программирования, получившую название .NET. Главная идея этой идеологии состоит в том, что программирование перестает быть средством работы на отдельном компьютере и становится инструментом для работы в компьютерных сетях и Интернете. При работе на сети особую важность приобретает вопрос безопасности выполнения программ. Данная идеология выводит программирование на новый уровень использования методов разработки программ и использования возможностей компьютера и Интернета.

Visual Basic 2010, так же как язык Visual C# 2010 и другие языки Visual Studio, позволяет очень эффективно решать научные, технические и экономические задачи с большим количеством числовых операций. С его появлением стало гораздо легче создавать код для решения сложных математических задач. Появляется возможность округления

чисел с заданной точностью, появились новые беззнаковые целые типы данных и новый числовой тип данных Decimal, который подходит для сложных финансовых расчетов. Созданы средства для преобразования двумерных координат из прямоугольной системы в полярную или наоборот. Теперь гораздо легче работать с трехмерными координатами как с единым объектом и гораздо проще осуществлять преобразования между прямоугольными, сферическими и цилиндрическими координатами.

Работу с комплексными переменными на Visual Basic 2010 нетрудно сделать понятной и естественной, если перегрузить стандартные математические операторы. Можно заметить, что создание подобных программ на C++ потребует гораздо больше усилий.

Visual Basic 2010 является очень продуктивным языком для разработки быстродействующих математических процедур на базе .NET Framework, реализующих громоздкие числовые задачи. Кроме этого, Visual Studio обладает самыми современными средствами для разработки графического интерфейса пользователя, средствами для программирования баз данных по технологии ADO.NET и средствами создания веб-сайтов и веб-страниц с помощью нового визуального инструмента Web Developer, использующего технологию ASP.NET.

В пособии будет рассмотрен инструмент разработки приложений для операционной системы Windows, Web и других сред, интегрированный в среду разработки Microsoft Visual Studio .NET 2010, который называется Visual Basic 2010 (его также называют Visual Basic .NET 2010).

Для простоты восприятия материала и более быстрого освоения среды разработки, первые две работы настоящего пособия хотя и используют Visual Basic 2010, но содержат только один класс (класс Form1), который создается автоматически. Однако, начиная с третьей работы, использована идеология объектно-ориентированного программирования, требующая написания собственных классов.

Часть I. ПРОГРАММИРОВАНИЕ НА VISUAL BASIC 2010

Т е м а 1. Освоение интегрированной среды разработки Microsoft Visual Studio. Работа с переменными и арифметическими выражениями на языке Visual Basic 2010 и .NET Framework

Изучение этой темы позволит:

- запускать Microsoft Visual Studio Express Edition;
- работать в интегрированной среде разработки Visual Studio;
- настраивать значения свойств элементов;
- перемещать, изменять размеры и скрывать окна инструментов;
- использовать переменные для хранения данных программы;
- использовать типы данных;
- использовать для ввода данных функцию InputBox;
- выводить данные на экран при помощи функции MsgBox;
- программировать математические формулы.

Платформа Visual Studio.NET Framework включает в себя языки программирования Visual Basic 2010 и Visual C# 2010, а также языки C++, F# и Jscript. Клиентские приложения используют окна, меню, кнопки и другие элементы графического интерфейса пользователя (GUI). Платформа .NET Framework позволяет настраивать визуальные атрибуты форм и заменяет элементы ActiveX на элементы Windows Forms, давая возможность отображать их в Интернете как веб-страницы.

Еще одним очень важным преимуществом платформы Visual Studio .NET является возможность создания, удаления и редактирования данных в отсоединенной среде при помощи технологии ADO.NET. Поскольку соединение с базой данных требует использования больших системных ресурсов сервера, то это ограничивает количество активных соединений с базой данных. Это создает много трудностей, особенно при работе интернет-приложений. Преимущество технологии ADO.NET состоит в том, что она позволяет клиентскому приложению установить соединение с сервером базы данных, получить оттуда необходимые данные и работать с ними в приложении, отключившись от базы данных. После выполнения обработки данных приложение имеет возможность повторно подключиться к источнику данных для его обновления.

Visual Basic Express Edition дает простой и надежный метод для создания программ для **Microsoft Windows**. Используя **Visual Basic 2010**, даже не имея достаточного опыта в программировании, можно получить полный набор средств для разработки необходимого технического приложения.

Хотя в данной работе используется язык программирования Visual Basic 2010, среда разработки, которую необходимо применить для написания программ, называется интегрированной средой разработки Microsoft Visual Studio, или коротко **IDE** Visual Studio. Эта среда программирования является в настоящее время наиболее перспективной и мощной, содержит все необходимые инструменты и средства для создания программ для Windows и Web. С Visual Basic 2010 есть возможность работать и без Visual Studio, однако при этом придется создавать файлы из командной строки, не имея возможности пользоваться средствами, которыми обладает указанная среда.

Что представляет собой Visual Basic? Слово Visual определяет метод, позволяющий создавать то, что пользователь видит на экране — так называемый **графический**

интерфейс пользователя (graphical user interface). Это очень важно, поскольку обычно бывает трудно разобраться в большом объеме поступающей информации. Слово Basic означает язык программирования, прошедший длинный путь развития и в настоящее время являющийся **объектно-ориентированным** языком программирования, основанным на событиях.

События (**events**) могут создаваться пользователем либо возникать при взаимодействии объектов в процессе выполнения приложения. Типичными событиями являются: щелчок мыши (Click), нажатие кнопки (Button_Click), загрузка формы (Load), создание на форме изображений (Paint), изменение значения заданной переменной, наступление заданного времени и т.п.

В этой работе необходимо научиться запускать **Visual Studio** и понять принципы использования IDE для открытия и выполнения программ на Visual Basic 2010. Необходимо также научиться изменять программные установки, называемые **свойствами**, а также научиться перемещать, изменять размеры, закреплять и скрывать все необходимые инструменты, которые используются в разрабатываемом проекте.

После запуска Visual Studio на экране появляется стартовая страница (Start Page) — рис. 1.1.1.

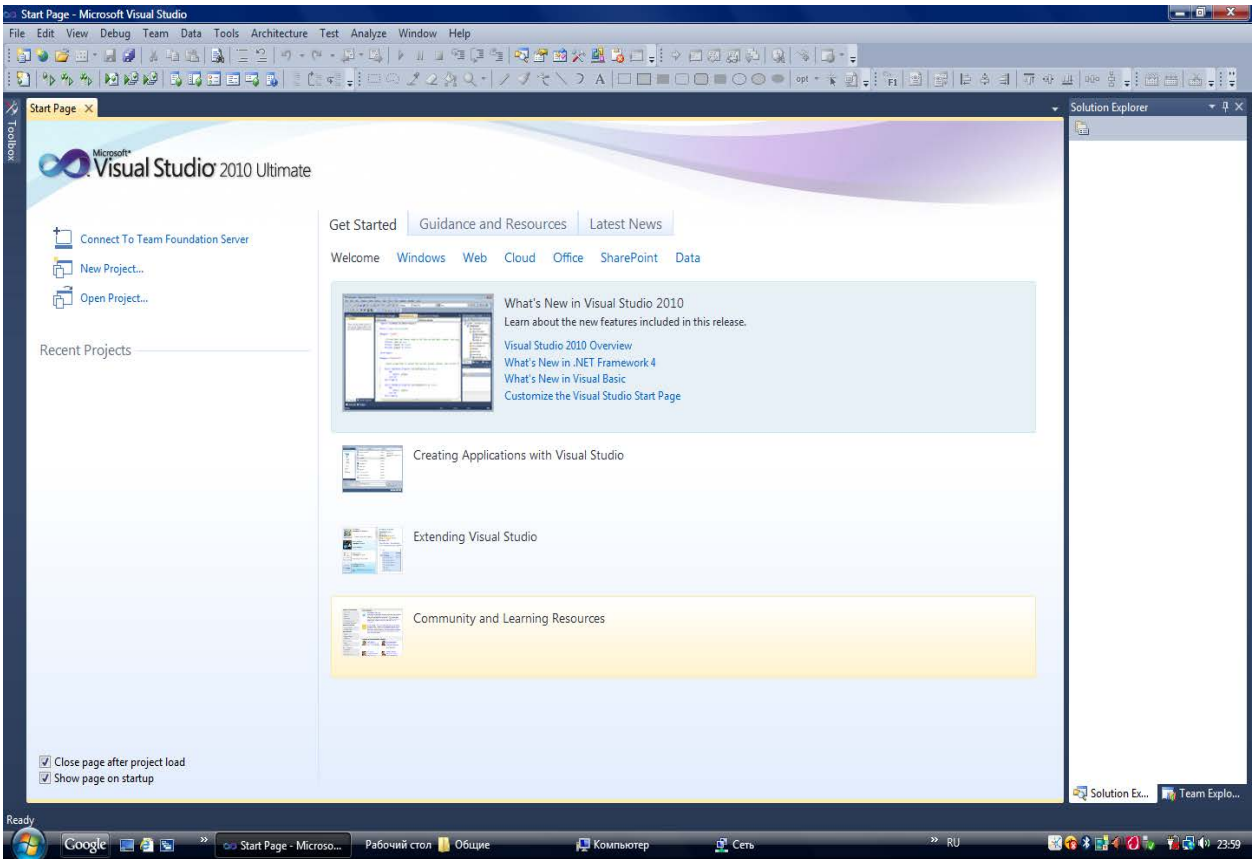


Рис. 1.1.1

Чтобы войти в среду разработки IDE, надо щелкнуть на ссылке **New Project** (Создать проект) и в раскрывшемся окне выбрать **Visual Basic** -> **Windows Forms Application** (Приложение Windows Forms). В нижней строке этого окна в строке Name (Имя) надо напечатать имя проекта (по умолчанию проекту присваивается имя **WindowsApplication1**) и щелкнуть на кнопке **ОК**. Если же необходимо работать с уже созданным ранее проектом,

то надо щелкнуть на ссылке **Open Project** и выбрать в появившемся окне имя нужного проекта, раскрыть папку и загрузить файл решения.

В Visual Studio программа разработки называется **проектом**, или **решением**, поскольку она содержат много отдельных компонентов, а не только один файл. Программы Visual Basic 2010 включают файл проекта (.vbprog) и файл решения (.sln). Файл проекта содержит сведения об одном или нескольких проектах.

Создание любого проекта обычно начинается с разработки дизайна формы (или нескольких форм), который позволяет пользователю взаимодействовать с компьютером в процессе выполнения проекта. Формы Windows представляют собой важную часть .NET Framework, которая используется для создания интерфейса пользовательского приложения. Форма по существу является окном, через которое пользователь видит на экране, что происходит в процессе работы приложения. Форма позволяет вводить данные, видеть происходящие события и получаемые результаты. Понимание принципа работы форм и умение их создавать крайне важно при разработке приложения.

Форма представляет собой рисунок, на котором создается интерфейс программы. На этом рисунке можно печатать текст, рисовать фигуры и размещать **элементы управления** (**Controls**), с которыми пользователь может взаимодействовать.

После входа в среду разработки (IDE) на экране появляется так называемый дизайнер форм (Designer); все создаваемые в проекте файлы показаны в **Solution Explorer** (Обозревателе решений). Эта среда может выглядеть по-разному: окна, из которых она состоит, могут помещаться в разные места экрана, изменять свои размеры и их можно просто убрать с экрана. Наиболее удобное их расположение показано на рис. 1.1.2. Дизайнер форм можно также вызвать на экран выбрав View -> Designer, или нажатием клавиш Shift+F7.

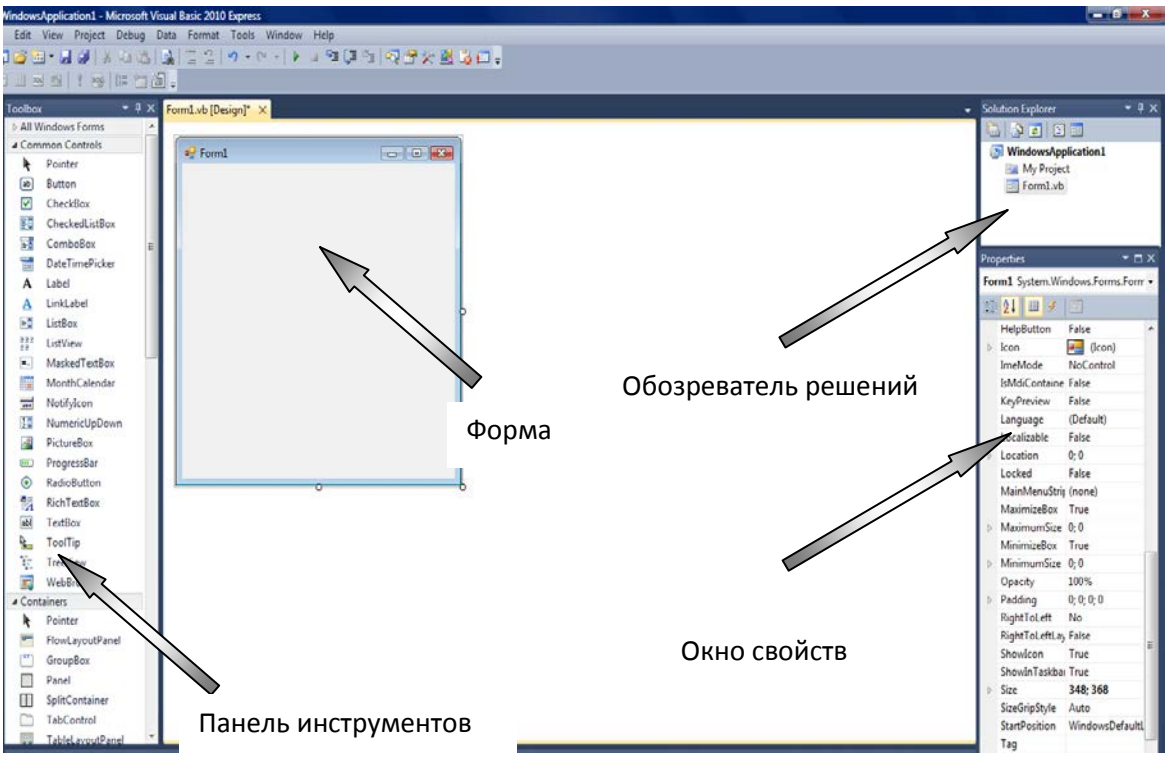


Рис. 1.1.2

Если на экране нет панели инструментов (**Toolbox**), то для получения доступа к ней надо в строке меню выбрать View -> Toolbox или одновременно нажать на клавиатуре

Ctrl+Alt+X. Если нет окна свойств (**Properties window**), то надо выбрать View -> Properties Window или нажать клавишу F4.

Создание графического интерфейса начинается с добавления элементов управления на форму. Предварительно надо увеличить размеры формы, растянув ее вправо и вниз. Затем необходимо перетащить при помощи мыши требуемые элементы в то место формы, которое предназначается для каждого элемента. В данной работе используются три вида элементов: Label (для отображения на форме текста), TextBox (для отображения данных) и командные кнопки Button (для программных кодов). Используя окно свойств (Properties window), можно установить многие свойства элементов, такие как цвет, размер, имя и позиции элементов на форме. Если элемент не требуется, то его можно пометить и удалить, нажав клавишу Delete. Каждый элемент управления имеет различный набор свойств в зависимости от действия, которое с ним связано.

Некоторые свойства являются универсальными, например, название (Name). Даже форма, которая является элементом управления, имеет это специальное свойство. Можно изменять имена, которые даются элементам по умолчанию, но при этом в имени не должны использоваться пробелы.

Существует несколько способов использования в приложении программных кодов. Один из этих способов связывает требуемый код с командной кнопкой Button. Для этого надо дважды щелкнуть в дизайнере на командной кнопке. После этого откроется новое окно, называемое **Редактором кода** (Code editor). Там уже будет некоторый код (написанный самой системой), и курсор установится между двумя линиями, задающими начало и конец процедуры Button1_Click. Между этими линиями кода и нужно напечатать требуемый код.

```
Private Sub Button1_Click (ByVal sender As System.Object, ByVal e As _
                    System.EventArgs) Handles Button1.Click
  < Код пользователя>
End Sub
```

Таким способом создается процедура **Sub**, которая будет выполняться после запуска приложения и щелчка мыши на кнопке Button1. Подобные процедуры называются **обработчиками событий**. Для данного случая событием будет щелчок мыши на кнопке Button (событие Button1_Click).

Запуск приложения можно выполнить, либо нажав функциональную клавишу **F5**, либо выбрав в пункте меню Debug команду Start Debugging (Отладка/Начать отладку), либо щелкнуть на кнопке  панели инструментов среды Visual Studio.NET.

Создание программного кода должно начинаться с определения типов всех используемых переменных. Типы данных делятся на структурные и ссылочные. Переменные структурного типа хранят данные непосредственно, а переменные ссылочного типа хранят в памяти не сами данные, а ссылки на них.

В Visual Basic 2010 типы данных являются классами и имеют собственные поля, свойства, методы и события. В табл. 1 представлены так называемые **примитивные типы** данных (типы-примитивы).

Таблица 1

Тип данных	Описание	Пример
Integer	Целое число, размер которого 4 байта и значение между –2, 147, 483, 648 и 2, 147, 483, 647	43
	Вещественное число (32-битное с плавающей	879.89

Single	точкой), которое может содержать значение между –3.4028235E38 и 3.4028235E38	
	Вещественное число с плавающей точкой, 64-	3.1415926535
Double	разрядное, содержит значение от –1 до 1.79769313486231E308	
Byte	Целое число, 1 байт, может содержать значение от 0 до 255	127
Decimal	Числа с десятичными знаками со значением в диапазоне $\pm 1.0 \times 10^{-28}$ и $\pm 7.9 \times 10^{28}$	10966.2592
String	Последовательность (или «строка») символов	«Hello!»
Char	Один символ	'h'
Boolean	Логическое значение, которое может быть либо True, либо False	True
Object	32-разрядные любого типа, может храниться в переменной типа Object	

В русифицированных версиях Visual Studio 2010 целая часть числа отделяется от дробной не точкой, а запятой.

Переменные играют важную роль в программировании. Имя переменной может быть буквой или комбинацией букв и цифр. При создании компьютерных программ переменные могут использоваться для хранения чисел (например, высота здания) или слов, таких как имя пользователя. Попросту говоря, можно использовать переменные для представления любой необходимой информации.

Имеются три шага при работе с переменной:

1. **Объявление переменной**. Необходимо объявить имя переменной и ее тип.
2. **Присвоение значения**.
3. **Использование переменной**.

Для объявления переменной используется ключевое слово **Dim**, например: **Dim a As Integer**.

Эта строка кода указывает программе, что будет использована переменная с именем a, и она будет иметь целый тип. Если необходима переменная со значением 232.5, например, то применяется тип данных **Single** или **Double**. И если нужно сохранить текст, то применяется тип данных, называемый **String**.

Типы переменных можно не объявлять, для этого в начале программного кода проекта надо установить для опции компилятора Option Explicit значение Off (отключить ее), что делается при помощи оператора

Option Explicit Off.

На уровне проекта по умолчанию для этой опции установлено значение On, и поэтому оператор Option Explicit On нужно задавать только тогда, когда известно, что эта опция была ранее отключена. Однако отключать эту опцию не следует, поскольку объявление типов переменных обеспечивает более надежную работу программы.

Повысить надежность программы может и опция Option Strict, которая по умолчанию отключена, т.е. установлена в Off. Для ее включения необходимо в начале программы записать оператор

Option Strict On.

После этого компилятор будет проверять, чтобы все значения, присваиваемые переменным, соответствовали типам, объявленным для этих переменных.

Например, пусть опция Option Strict отключена и имеется программа

```
Dim a As Integer
a = InputBox("Введите значение a")
```

В этом случае программа будет нормально работать, а необходимое преобразование будет выполнено автоматически по умолчанию, если вводимое значение не превысит значения, допустимого для типа Integer (т.е. будет меньше 2, 147, 483, 648).

Если же будет включена опция Option Strict, тогда компилятор выдаст сообщение об ошибке, где будет сказано, что оператор Option Strict On не позволяет выполнить неявное преобразование символьного значения (поскольку InputBox всегда выдает символьное значение) в целочисленное значение, которое имеет переменная a.

Для преобразования величин имеется много способов, некоторые из них взяты из предыдущих версий языка Visual Basic; так, например, можно использовать функцию Val()

```
a = Val(InputBox("Введите значение a")).
```

Для преобразования числа в символьную переменную можно использовать функцию Str() — например, когда надо преобразовать числовую переменную в символьную, для последующей ее записи в текстовое окно, значения которой имеют символьный тип Textbox1.Text = Str(A).

В то же время для преобразования примитивных типов данных несомненно более предпочтительными являются методы, представленные ниже:

Таблица 2

CBool( ) CDbl( ) CShort( )	CByte( ) CDec( ) CSng( )	CChar( ) CInt( ) CLng( )	CDate( ) CObj( ) CStr( )

Каждый из этих методов имеет в качестве аргумента любой другой примитивный тип и преобразует его в тип, указанный в имени данного метода. Так, метод CStr( ) преобразует примитивный тип в символьный, метод CSng( ) — в вещественный, метод CInt( ) —в целочисленный тип. Например, переменная вещественного типа a может быть преобразована в целый тип (Integer) одним из следующих способов:

```
Dim a As Single = 45.678
Dim d As Integer
    d = CInt(a),     или     d = Convert.ToInt32(a).
```

Опции компилятора Option Explicit, Option Strict, Option Infer можно устанавливать не только в программе. Установить их можно и в главном окне среды разработки, для этого надо выбрать команду меню Project и в ней свойства (Properties) проекта. Откроется окно общих настроек проекта, в котором надо выбрать вкладку Compile («Компилировать») и в раскрывшемся окне установить параметры компиляции (Compile Options).

Это окно можно также открыть, если в Обозревателе решений (Solution Explorer) дважды щелкнуть на My Project.

Далее в приведенной ниже таблице предлагает частичный список математических методов (функций) в классе **System.Math**. Аргумент n в таблице представляет собой либо число, либо переменную, либо выражение.

При использовании любой из этих функций необходимо в форме в Редакторе кода поместить объявление

Imports System.Math

Таблица 3

Метод	Цель
Abs(n)	Возвращает абсолютное значение n.
Atan(n)	Возвращает арктангенс в радианах n
Cos(n)	Возвращает косинус угла n. Угол выражается в радианах.
Exp(n)	Возвращает экспоненту.
Sin(n)	Возвращает синус угла n. Угол выражается в радианах.
Sqrt(n)	Возвращает квадратный корень из n.
Tan(n)	Возвращает тангенс угла n. Угол выражается в радианах.
Log(n)	Возвращает натуральный логарифм.
Pow(n, m)	Возвращает n^m
RND	Возвращает случайное число на отрезке [0, 1]

Если не включить предыдущее **Imports** заявление, то вызвать любой метод — например, Abs(n) — можно, если ввести **Math.Abs(n)** вместо просто **Abs(n).**

Visual Basic имеет следующие арифметические операторы:

Таблица 4

Оператор	Описание
+	Сложение
-	Вычитание
*	Умножение
/	Деление
\	Целочисленное деление
MOD	Остаток от деления
^	Возведение в степень
&	Сцепление строк (конкатенация)

Одна из особенностей Visual Basic 2010 состоит в том, что можно использовать **укороченные операторы** для математических и строковых операций (как в языке C++). Например, можно написать формулу X = X + 10 с помощью короткого синтаксиса X += 10. В табл. 5 показаны эти операции:

Таблица 5

Операция	Обычный синтаксис	Короткий синтаксис
Сложение	X = X + 10	X+=10
Вычитание	X =X – 10	X -= 10
Умножение	X = X * 10	X * = 10
Деление	X = X / 10	X / = 10
Целочисленное деление	X = X \ 10	X \ = 10
Возведение в степень	X = X ^ 10	X ^ = 10
Конкатенация	X = X & "СТРОКА"	X & = "СТРОКА"

Visual Basic 2010 позволяет использовать любое количество арифметических операторов в формуле. Например, имеется следующее математическое выражение:

$$t = \frac{a+b}{c+d}.$$

В этой формуле 4 переменные и 3 оператора. Порядок выполнения этих операторов очень важен и при написании программы это должно быть учтено. В данном примере и числитель, и знаменатель должны быть заключены в круглые скобки. Если, допустим, нет скобок в знаменателе, то числитель будет разделен на c, а потом к результату будет прибавлено d, что не позволит получить правильное значение t. Поэтому на Visual Basic 2010 должно быть записано так:

t = (a+b)/(c+d).

Скобки необходимы и в следующем выражении:

$$t1 = \frac{a+b}{c \cdot d}.$$

В Visual Basic 2010 оно будет будут записано так: T1 = (a+b)/(c*d).

Следует также отметить, что корень степени n (n > 2) могут быть представлен в форме

$$\sqrt[n]{x} = x^{\frac{1}{n}},$$

а затем Visual Basic 2010 он программируется как x^(1/n).

В табл. 6 перечислены операторы в порядке их выполнения в соответствии с приоритетом.

Таблица 6

Оператор	**Приоритет выполнения**
( )	Значения в круглых скобках всегда вычисляются раньше всего
^	Возведение в степень имеет второй приоритет
*/	Умножение и деление имеют третий приоритет
\	Целочисленное деление имеет четвертый приоритет
MOD	Остаток от деления — пятый
+-	Сложение и вычитание — последний

Выражением является сегментом кода, который выполняет арифметические действия и затем возвращает значение. Например, простое вычитание: 12 – 5.

Оно возвращает значение 7 и состоит из двух частей: операндов (12 и 5), которые являются значениями оператора вычитания (–).

Иногда бывает удобно при описании типа переменной сразу задать и ее начальное значение, которое также может быть выражением, например

Dim a AS Integer = 12 – 5.

Здесь объявляется новая переменная типа **Integer** с именем a и одновременно ей присваивается значение 12 – 5.

Типы переменных, которые используются в выражении, могут оказывать влияние на его результа. Например, если разделить 3 на 2, то результат будет равен 1.5. но это значение округляется до 2, поскольку возвращаемое значение является целым числом типа Integer.

Рассмотрим следующий пример

```
Dim A AS Double = Textbox1.Text
Dim B As Double = Textbox2.Text
  MsgBox (A + B): MsgBox (A - B):MsgBox(A * B): MsgBox(A / B)
```

Первые две строки представляют собой объявление переменных A и B; при этом A и B будут иметь числовой тип Double и получат значения, которые читаются из элементов управления TextBox1 и TextBox2.

В третьей строке отображаются на экране в диалоговых окнах при помощи функции MsgBox результаты четырех операций: сложения, вычитания, умножения и деления.

Для того чтобы операторы этих четырех строк были выполнены, надо поместить на форму кнопку Button1, дважды щелкнуть на ней и в редакторе кода напечатать их в процедуре данной кнопки. Затем нажать F5 для запуска приложения. После этого ввести числовые значения A и B в соответствующие текстовые окна и щелкнуть на кнопке Button1. Результаты будут последовательно появляться в диалоговых окнах.

ЛАБОРАТОРНАЯ РАБОТА № 1

Целью работы является ознакомление с типами данных и правилами построения арифметических выражений, а также приобретение практических навыков работы в интегрированной среде разработки Visual Studio.

Вопросы, которые должны быть изучены:

1. Создание простого приложения на Visual Basic.
2. Создание нового проекта.
3. Добавление элементов управления на форму.
4. Проектирование интерфейса.
5. Объявление переменных.
6. Присвоение значений переменным.
7. Использование переменных.
8. Порядок выполнения операций в выражении.
9. Использование значений, возвращенных выражениями.
10. Арифметические операторы.
11. Преобразование числовых переменных в текст.
12. Преобразование числовых типов данных.
13. Построение арифметических выражений.

Задание

1. Выбрать вариант индивидуального задания из табл. 7.
2. Написать код программы для математического выражения.
3. Запустить Visual Studio 2010. Выбрать в меню опцию **File** и затем **New Project**.
4. В окне **New Project** выбрать **Online** Templates и затем **Windows Forms** Application.
5. В поле **Name** ввести **Arithmetic** и нажать кнопку **ОК**. Открывается новый проект.
6. Из панели элементов **Toolbox** перетащите на форму требуемые элементы управления.
7. Двойным щелчком на элементе **Button1** открыть код «Редактор кода».
8. В процедуру события **Button1_Click** ввести требуемый код. Новая форма отображает в IDE все файлы, необходимые для проекта. Если это первый созданный проект

Windows Forms Application, то он называется WindowsApplication1 (его можно переименовать — допустим, Arithmetic).

9. В меню Debug Visual Basic IDE нажать кнопку Start Debugging. Эта команда запускает программу. Для запуска программы можно также нажать клавишу F5 или щелкнуть на пиктограмме в виде треугольника, находящейся в верней части экрана.

10. Получить результаты на компьютере.

11. Проверить правильность полученных результатов.

12. Написать отчет.

Индивидуальные задания

Таблица 7

№	x	y	Формула
1	2.34	5.2	$\frac{\cos x^5+sin^3(x+y)}{\sqrt{x-\cos y}} \cdot \mathrm{tg}\sqrt[4]{x \cdot y}$
2	–2	–5.61	$x \cdot y - \mathrm{ctg}(x^5+12) \cdot \frac{(x-y \cdot \sin y)^3}{2.7-\lg(x+4)}$
3	1.23	–5.56	$\sqrt[3]{x \cdot \cos\lvert y\rvert} + \sqrt[2]{\lvert \sin(x) + cos^5(x \cdot \lg(x))\rvert}$
4	1.2	4.6	$\frac{\sqrt[7]{(y+x)^3 - 3.6x^3}}{e^{5 \cdot 1.6}}$
5	21	2.1	$\frac{7.8^y \cdot (x^2+y)^2 + 1}{\sqrt{\lg(20+12.6 \cdot x) - e^{3.7}}}$
6	6.78	3.45	$\frac{\sqrt[3]{\sin\sqrt[5]{x+y^4}}}{tg\ x+\cos(x \cdot y)} + \sqrt{x-y}$
7	7.81	–3.44	$\frac{\cos x \cdot 3.14 \cdot 6.45^3}{5.87 \cdot tg^{x+y}3.5} + \frac{x+y^{3.23}}{32-x}$
8	1.45	–5.23	$\frac{e^{x+2 \cdot y} - \sin x^{2 \cdot y}}{1+x \cdot cos\ y} - \frac{1}{1+ctgx^3}$
9	5.1	–12.4	$\frac{\sqrt[3]{x+5.6}}{e^{x \cdot y} - 21.4 \cdot tgx} - \frac{3.55}{\sqrt[4]{x \cdot \lvert y\rvert}}$
10	0.78	0.93	$\frac{\lvert \sin y + cos^3 y\rvert}{e^{-x}+x^2} \cdot \mathrm{ctg}x^y + \ln x^3$

11	5.7	3.9	$\frac{tg\,x^3-(x+y)}{\sqrt{cosx+cosy}}\cdot\sqrt[5]{x-y+1}$
12	1.81	3.65	$\frac{lnx+cos^4(x+y)}{\sqrt{5+cosy}}\cdot\frac{2.3+x^2}{x\cdot y}$
13	21.7	8.1	$\frac{\sqrt{x+7}\cdot cosx^7+\sqrt[3]{2\cdot y})}{\sqrt{x-cosy}}\cdot tg\sqrt[3]{x+23\cdot y}$
14	0.25	1.76	$\sqrt{4\cdot x+\sqrt[3]{5\cdot y-\sqrt[4]{cosx^3}}}+\frac{cosy}{1-ctgx}$
15	5.7	127.4	$\frac{\sqrt{y\cdot 23.8}+\sqrt[5]{y}+sinx}{\sqrt{cosx}}\cdot e^{-x}\cdot x^2$
16	6.7	0.45	$\sqrt{\sqrt[5]{x+siny+\sqrt{cos^4 x}}}$
17	2.1	0.27	$\frac{cosx}{4\cdot x}\cdot\frac{2\cdot x^2}{ctgx+cos^4(x)}+\lvert\cos(x)+0.12\cdot y\rvert$
18	–12.7	2.3	$(x+siny)^{2+x}+\ln x^2+\frac{\lvert x\rvert+4}{45\cdot x\cdot y}$
19	–4.77	3.92	$\frac{x^3-x\cdot y}{0.23\cdot sin\lvert x\rvert}+\text{ctg}\frac{x-y}{2\cdot x}+ctg^3(x+y^4)$
20	2.6	4.13	$\ln\frac{3}{x+sinx^2}+\frac{e^y}{4+\frac{sinx}{1+cosy}}$
21	12	5.8	$\frac{y\cdot x^2}{4.2+cosy}+tgx^3-\cos^4(x+5\cdot y^3)$
22	3.5	6.8	$\frac{\sqrt[3]{cos^3 x+sinx^4}\cdot\sqrt{lny+12.6}}{ctgx^5}$
23	–4,8	–6.12	$\frac{\sqrt[4]{\lvert x-2\cdot y\rvert}}{1+e^{x+2}}-\sqrt{\lvert sinx-siny\rvert}$

24	5.1	12.4	$189 \cdot \sin x^4 + 1.3 \cdot 10^{15} + \frac{arctgx^4}{\sqrt[3]{x+12 \cdot y}}$
25	23	12.9	$\frac{\cos(x+y)^x + 12}{14.23 + \frac{sinx \cdot cosx}{\sqrt{x+17} + y + \sqrt[3]{cosx + 8.7}}}$
26	9	0.55	$\frac{x^{y+7}}{\cos x - \frac{\sin y}{5+e^{2y}}} + x^2 - \frac{\pi}{2} + \frac{\cos 2y}{\tan x}$

Содержание отчета

1. Название отчета.
2. Титульный лист.
3. Индивидуальное задание.
4. Запись математической формулы на языке Visual Basic 2010.
5. Форма проекта с элементами управления.
6. Результаты выполнения приложения.
7. Доказательство правильности работы программы.

Пример выполнения индивидуального задания

Задание на разработку проекта. Создание приложение для вычисления следующего математического выражения.

$$Z = \frac{\sqrt{y+5} + |x^2 + cos^3(x)|}{1 + ctg(y^5)},$$

x = 12,6 и y = 7.34,

где x, y — исходные данные тип Single и z — результат выражения.

Решение

1) создать новый проект (New Project);
2) создать графический интерфейса пользователя;
3) установить свойства элементов управления;
4) написать программный код для вычисления математического выражения;
5) запустить программу Visual Basic 2010 и получить результаты;
6) провести анализ результатов.

Когда создается новый проект, то он существует только в оперативной памяти. При закрытии интегрированной среды разработки Visual Basic 2010, появится запрос на сохранение или удаление этого проекта (save or discard). Если проект необходимо сохранить, то надо выбрать **save**, при этом имеется возможность дать ему другое имя.

Обычно на экране компьютера видна одна форма, однако проект может состоять и из нескольких форм, и при этом они все находятся в одном окне. В то же время можно закрывать окно с одной формой и переходить в окно с другой, и таких переходов можно делать сколько угодно.

Блок-схема алгоритма для этой задачи представлена на рис. 1.1.3.

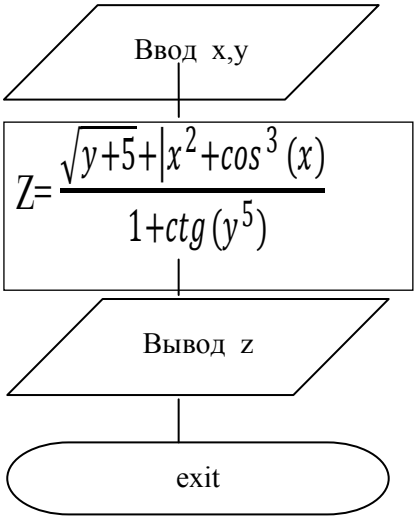


Рис. 1.1.3

Интерфейс пользователя, созданный в Дизайнере для данной задачи с названиями всех использованных элементов управления, показан на рис. 1.1.4. Кнопки представляют собой прямоугольники, которые видны на форме. Можно установить их размеры, цвет и внешний вид. Для этого надо установить их свойства (**Properties**). Наиболее очевидным является свойство Text, которое определяет текст на экране и отображает его в установленном шрифте или начертании шрифта, определяемом свойством **Font**. Свойство **BackColor** определяет цвет элемента.

В этой работе установлены свойства, как показано в табл. 8. Параметры показаны в кавычках (кавычки набивать не надо).

Таблица 8

Объект	**Свойство**	**Параметр**
Button1	Text	Start
Button2	Text	"Quit"
Label1	Text	"Formula"
Label2	Text	"x ="
Label3	Text	"y ="
Label4	Text	"Result"
TextBox1	Multiline	False
TextBox2	Multiline	False
TextBox3	Multiline	True
PictureBox1	Image	Image File
Form1	Font Text	Microsoft Sans Serif; 14,25 pt Creating Arithmetic Expressions

Для создания **Image File** надо в редакторе Word напечатать формулу и скопировать ее в Paint, оттуда сохранить на диск с расширением .jpg, запомнить имя и затем импортировать файл в PictureBox1 (Choose Image -> Local Resource).

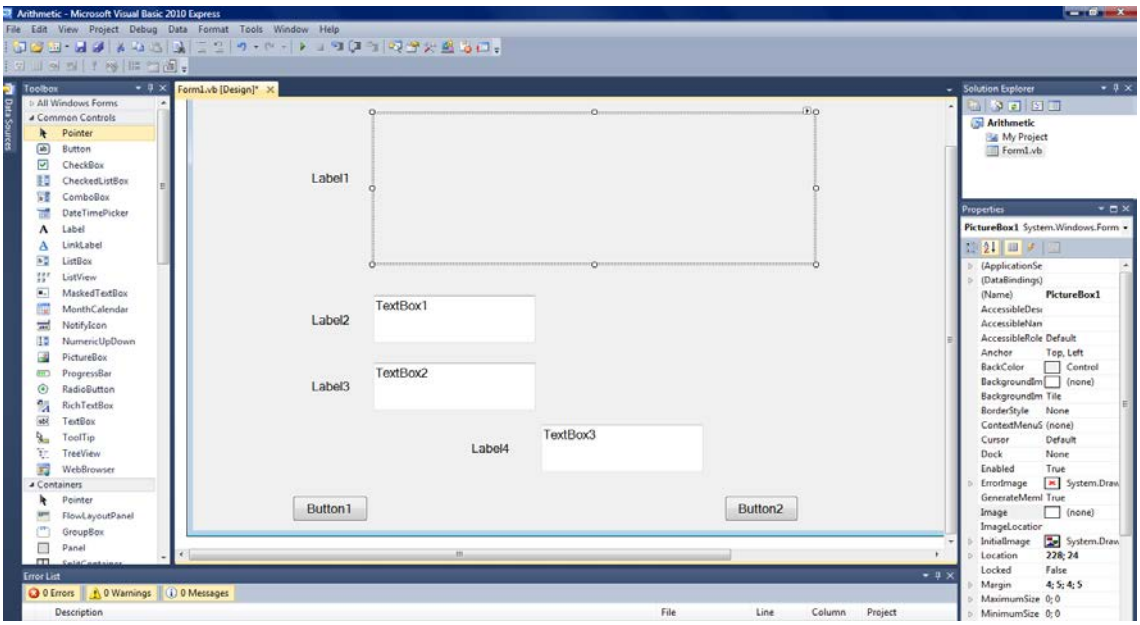


Рис. 1.1.4

Написание программы в Visual Basic 2010

Строка кода в программе Visual Basic называется *программным предложением* (или *оператором*). Программным предложением является любое сочетание зарезервированных слов Visual Basic, имен объектов, переменных, чисел, специальных символов и других величин, образующих инструкцию, смысл которой способен распознавать компилятор Visual Basic 2010. *Переменная* представляет собой временное размещение данных в программе. Для объявления переменной в Visual Basic 2010 необходимо ввести имя переменной после **Dim**. Эта декларация резервирует место в памяти для переменной, а также задает ее тип.

Следующая инструкция объявит три переменные

Dim x, y, z As Single

Все переменные имеют один тип Single. (Можно объявлять любое количество переменных в одной строке.)

Одним из способов ввода входных данных является использование функции **InputBox**; для этого надо присвоить переменной значение данной функции. При этом текстовое значение функции InputBox необходимо конвертировать в числовое (Single) при помощи метода CSng( ). Числовые значения x, y, z для пересылки в текстовые окна (TextBox.Text) преобразуются в символьные при помощи метода CStr( ). Преобразование выполнять необходимо, поскольку включена опция Option Strict On (рис. 1.1.5).

Form1.vb* X Form1.vb [Design]*

(General) (Declarations)

```
Option Strict On
Imports System.Math
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim x, y, z As Single
        x = CSng(InputBox("Please enter x"))
        y = CSng(InputBox("Please enter y"))
        TextBox1.Text = CStr(x)
        TextBox2.Text = CStr(y)
        z = CSng(Sqrt(y + 5) + Abs(x ^ 2 + Cos(x) ^ 3) / (1 + Cos(y ^ 5) / Sin(y ^ 5)))
        TextBox3.Text = CStr(z)
    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. 1.1.5

Результат выполнения приложения показан на рис. 1.1.6.

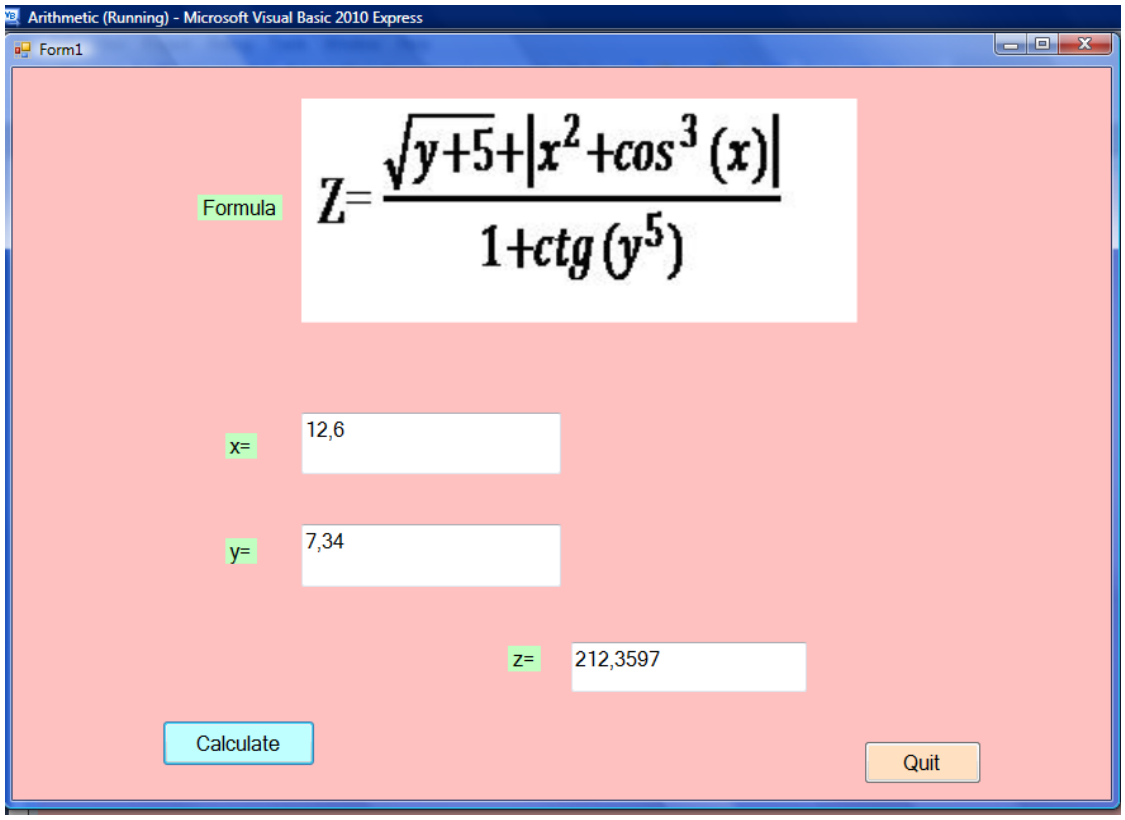


Рис. 1.1.6

Сравнение результата 212.3597, полученного путем ручного расчета, с результатом на компьютере доказывает правильность составленного алгоритма.

Т е м а 2. Освоение интегрированной среды разработки Microsoft Visual Studio. Работа с элементами управления, меню и панелями инструментов. Программирование процедур Sub и Function

Изучение этой темы позволит:

- настраивать параметры IDE;
- перемещать и изменять размеры инструментов Visual Studio;
- использовать Toolbox при создании программ;
- использовать технику структурного программирования и переменные типа Public при создании процедур;
- применять открытые переменные Public, имеющие глобальную область видимости;
- увеличивать продуктивность программ путем создания процедур типа Function и Sub;
- применять синтаксис вызова и использования пользовательских процедур;
- передавать аргументы процедур по значению;
- работать с элементами управления Scrollbar;
- использовать таймер.

Непрерывное развитие компьютерных технологий приводит к тому, что для сохранения необходимого уровня программисты должны переучиваться каждые несколько лет. Рост технических возможностей компьютера открывает новые области его применения, что также требует создания новых средств компьютерных технологий.

Для решения этой проблемы компания Microsoft в 2002 году создала платформу .**NET Framework**, это выдающийся результат в сфере создания компьютерных технологий, который объединяет в себе все лучшее из существующих в настоящее время систем.

Основная идея концепции .NET Framework состоит в том, что различные программные приложения предлагаются пользователю как сервисы, способные взаимодействовать друг с другом, в соответствии с конкретными решаемыми задачами, и доступны на самых разных устройствах — от мощного компьютера до мобильного телефона.

Графические средства этой платформы используются для построения пользовательского интерфейса программ Microsoft Visual Basic 2010. Элементы управления, расположенные на форме, обычно используются для создания действий, вызываемых простым щелчком мыши или какими-либо другими ее движениями. Все объекты имеют свои собственные свойства, методы и события. Это относится и к формам и к элементам управления. *Свойства* определяются как атрибуты объекта, *методы* — как его действия и *события* — как реакция на действия. Построение пользовательского интерфейса (его видимой части, с помощью которой пользователи взаимодействуют с системой) представляет собой процесс добавления элементов управления с панели инструментов на форму.

Панель инструментов Toolbox обычно располагается с левой стороны Visual Studio и состоит из таких элементов, как **TextBox**, **Panel**, **Components** и всех форм Windows. Эти элементы могут быть добавлены в любое разрабатываемое приложение, для этого достаточно просто перетащить их на форму. Кроме этого, пользователь должен настроить данные элементы, задавая их свойства.

Элемент управления **панель** (*Panel Windows Form*) используется для группировки других элементов управления. В процессе разработки приложения (в режиме design) все элементы, помещенные в Panel, оказываются сгруппированными, т.е. при перемещении

панели в какое-то место формы одновременно будут перемещаться и эти элементы. Сам элемент Panel по умолчанию отображается без форм и границ. Элемент Panel не имеет заголовка для своей идентификации, и если такой заголовок требуется, то надо использовать элемент **GroupBox**. Каждый элемент управления содержит код, определяющий, как выглядят этот элемент и какие задачи он выполняет. Например, элемент управления Button («командная кнопка») для большинства программ имеет надпись Start или Exit. Можно изменить внешний вид элемента и программным путем, для этого нужно написать свой собственный программный код в процедуре этого элемента. Собственный программный код необходимо писать и в том случае, если требуется, чтобы соответствующий элемент управления выполнял какие-либо заданные функции.

В данной работе необходимо освоить элемент управления **Scrollbar** («полоса прокрутки»), который позволяет обеспечивать быстрый выбор и навигацию в большом списке элементов (как правило, этот список состоит из чисел, образующих последовательность с заданным шагом).

Имеются **HScrollBar** (горизонтальная) и **VScrollBar** (вертикальная) полосы прокрутки (не следует путать их со встроенными в другие элементы управления собственными полосами прокрутки). Набор событий, свойств и методов элемента управления ScrollBar отличается от подобных событий, свойств и методов встроенных прокруток.

Рассмотрим некоторые свойства горизонтальной полосы прокрутки.

Свойство **Value** (по умолчанию равно 0) задает **целочисленное значение, соответствующее** положению бегунка в полосе прокрутки. Перемещая бегунок слева направо, можно изменить значение Value от минимального до максимального. Полученное значение зависит от положения бегунка полосы прокрутки, но оно всегда находится внутри диапазона между **минимальным** и **максимальным** значениями, заданными пользователем.

Когда пользователь нажимает клавишу PAGE UP или PAGE DOWN или щелкает в панели прокрутки слева от бегунка, значение свойства изменяется в соответствии со значением, установленным в свойстве **LargeChange**.

Текущее положение бегунка полосы прокрутки хранится в ее свойстве Value, и его можно сделать значением переменной (например, p), как это показано в следующем примере (рис. 1.2.1).

p = HScrollBar1.Value

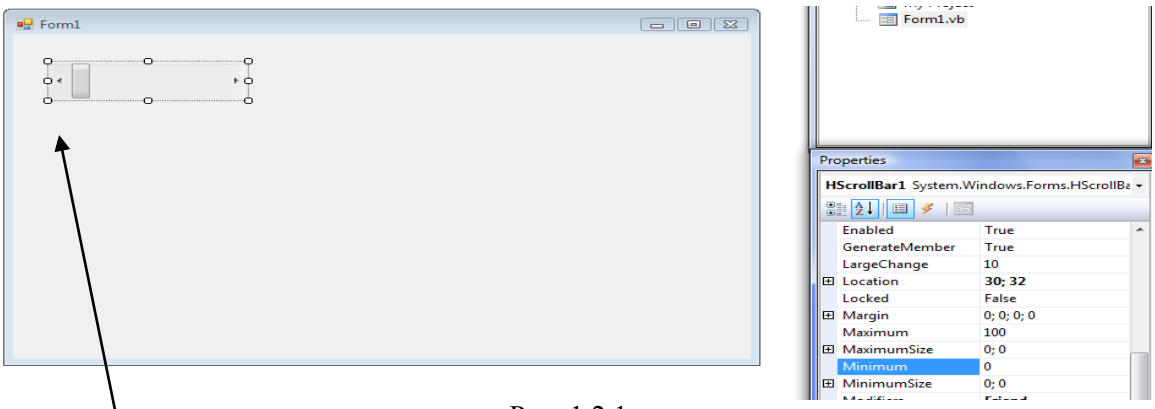


Рис. 1.2.1

Бегунок находится в позиции минимального значения (**HScrollBar1.Minimum**)

В этой работе необходимо создать проект для обработки числовых данных, используя процедуры типа Function и Sub. В Microsoft Visual Basic 2010 все выполняемые операторы должны быть помещены внутри некоторой процедуры.

Для повышения эффективности программирования могут быть созданы модули, а также **определяемые пользователем функции и процедуры Function и Sub**. Модули представляют собой отдельные области в пределах программы, которые содержат глобальные (**Public**) переменные, а также функции и Sub-процедуры. Необходимо понять и научиться использовать открытые (**Public**) переменные, которые особенно часто требуются в больших проектах.

Для объявления глобальной переменной надо использовать ключевое слово **Public**, за которым необходимо указать имя переменной и ее тип. Например, объявление

Public x as Integer

позволяет ввести открытую переменную целого типа с именем x.

Написание и вызов функции Function. Функция Function — это группа инструкций, расположенных между Function и End Function. Операторы функции могут обеспечивать обработку текста, ввод или расчет некоторых числовых значение и т.п.

Аргументы функции являются данными, используемыми в процессе вычисления функции, и они должны быть заключены в круглые скобки и разделены запятыми.

Базовый синтаксис функции выглядит так:

Function FName ([аргументы]) As Type
Операторы функции
[Возвращаемое значение]
End Function

Fname — это имя функции, которое вводит пользователь.

Тип (*Type*) функции определяет возвращаемое значение. Если тип не объявлен, то по умолчанию он будет объявлен как **Object**.

Каждый аргумент должен объявляться. Если аргумент объявлен с помощью ключевого слова **ByVal** (по значению), то тогда любые изменения этого аргумента в функции не будут возвращаться после ее выполнения. Однако с помощью зарезервированного слова **ByRef** указывается, что переменные должны быть переданы процедуре по ссылке, и тогда любое изменение переменной в процедуре передается обратно в вызывающую программу. Поэтому необходимо использовать ByVal, когда не требуется изменять аргумент в процедуре. В случае сомнения удобнее использование зарезервированного слова ByRef.

Предположим, например, что нужно написать функцию **Func**, вычисляющую следующее выражение:

$$x + \cos(x) - \sin(x)$$

Эта функция будет выглядеть так:

```
Function Func (ByVal x as Single) As Single
 Func = x + Cos(x) – Sin(x)
End Function
```

В качестве альтернативы можно использовать синтаксис Visual Basic 2008 и возвращать значение в вызывающую процедуру с помощью инструкции **Return**:

```
Function Func (ByVal x as Single) As Single
     Return x + Cos(x) – Sin(x)
End Function
```

Надо помнить, что функция не может возвращать более одного значения.

Для вызова функции Func в событийной процедуре используется инструкция, аналогичная следующей:

```
Dim x, y As single
x = 5
y = Func(x)
```

Если в приложении имеется элемент TextBox1, то вызов можно осуществить следующим образом:

TextBox1.Text = **Func(x)**.

Написание и вызов подпрограмм Sub. Процедура Sub аналогична функции Function, за исключением того, что процедура Sub не возвращает значение, связанное с ее именем, и может возвращать более одного аргумента.

Синтаксис процедуры:

```
Sub Имя процедуры ([аргументы])
  Операторы процедуры
End Sub
```

Каждый аргумент должен объявляться как определенный тип. Если объявление не сделано, то Visual Studio добавляет ByVal для каждого аргумента. Объявления аргументов с помощью ByRef позволяет передавать обратно в вызывающую процедуру имеющиеся изменения этих аргументов. Аргументы при объявлении и процедуры называют также формальными параметрами.

Для вызова процедуры Sub в программе необходимо указать ее имя и аргументы. Например: **Имя процедуры (аргументы)**.

При вызове процедуры Sub число и тип аргументов должны соответствовать числу и типу аргументов в процедуре, и аргументы должны быть заключены в круглые скобки. Более того, при вызове процедуры аргументы должны быть записаны в том же порядке, в каком они записаны при ее определении. Такая передача аргументов называется **позиционной**. Вместе с этим существует и другая возможность передачи аргументов — это передача аргументов по имени. В этом случае порядок записи аргументов не играет никакой роли. Однако следует заметить, что принцип объектно-ориентированного программирования, называемый **полиморфизмом**, позволяет менять при обращении и типы аргументов процедуры и их количество, и при этом процедура будет по-прежнему выдавать правильные результаты.

Рассмотрим пример Sub:

Найти площадь треугольника, заданного длинами его сторон, используя формулу Герона. (Эта формула приписывается Герону из Александрии, однако она была известна еще Архимеду.) Пусть a, b, c — длины сторон треугольника, и s — его площадь.

Тогда формула имеет вид

$$s = \sqrt{(p \cdot (p-a) \cdot (p-b) \cdot (p-c))},$$

где p = (a+b+c)/2 или периметр/2.

Пусть имя процедуры будет **triangle**. (Можно объявить процедуру в модуле, но в данном примере это объявление сделано в классе формы). Эта процедура Sub объявляется в форме Form1 и вызывается в событийной процедуре **Button1_Click**.

```
Imports System.Math
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim a, b, c, s As Single
        a = 5 : b = 6 : c = 3
        triangle(a, b, c, s)  ←——‘  Вызов процедуры
        TextBox1.Text = s
    End Sub

    Sub triangle(ByVal a As Single, ByVal b As Single, ByVal c As Single, ByRef s As
Single)
        Dim p As Single
        p = (a + b + c) / 2
        s = Sqrt(p * (p - a) * (p - b) * (p - c))
        MsgBox("s=" & s)
    End Sub
End Class
```

Краткое описание некоторых терминов в этом разделе:

Method — набор команд для выполнения какого-либо действия.

Sub — метод, который содержит набор команд некоторой процедуры и может возвращать любое количество аргументов.

Function — метод, который содержит набор команд некоторой функции и может возвращать только одно значение.

Return — окончание выполнения Sub или Function и возвращение значения для функции.

Dim — объявляет и определяет новую переменную.

Module — блок кода, который не является классом, но может содержать методы Sub и Function.

ЛАБОРАТОРНАЯ РАБОТА № 2

Целью работы является изучение процедур Function, Sub и модулей, а также использование простых линейных алгоритмов.

Вопросы, которые должны быть изучены:

1. Как применяется Visual Studio IDE?
2. Как изменить свойства элементов управления?
3. Как перемещать, изменять размеры, закреплять на форме и автоматически скрывать инструменты окон?
4. Как сохранить изменения в проекте?
5. Как создавать функции, определенные пользователем?
6. Как программируется процедура типа Function?
7. Как программируется процедура типа Sub?
8. Как вызываются Function и Sub?
9. Как передавать аргументы по значению и по ссылке?

Задание

1. Выбрать вариант индивидуального задания.
2. Написать код программы для этого задания.
3. Запустить Visual Studio 2010. В меню Visual Studio File щелкнуть New Project.
4. В окне New Project выбрать шаблон Windows Application.
5. В текстовом поле Name ввести имя **lab2** и нажать кнопку *ОК*. Открывается новый проект Windows Forms.
6. Из Toolbox перетащить элемент Panel на форму.
7. Из Toolbox перетащить в Panel элементы HScrollBar1, Label1 и TextBox1 и задать их свойства (в частности, свойство dock).
8. Из Toolbox перетащить две *кнопки* **Button**, три Label и три TextBox, размещая их на форме ниже элемента Panel.
9. Выбрать команду «Добавить новый элемент» (Add New Item) в меню Project. Откроется диалоговое окно.
10. В этом окне выбрать шаблон **Module**. Имя по умолчанию Module1.vb. Щелкнуть на кнопке **Add** («Добавить»). Модуль отображается в редакторе кода, туда нужно ввести код процедуры Sub.
11. Дважды щелкнуть *Button1*, чтобы открыть редактор кода этого элемента.
12. В процедуре обработки события **Button1_Click** вызвать процедуру модуля, указав ее имя и список аргументов. После этого необходимо ввести код программы процедуры Button2 _Click и код процедуры HScrollBar1_Scroll.
13. В опции меню *Debug*, интегрированной среды разработки Visual Basic, нажать кнопку *Start Debugging*.
14. Получить результаты на компьютере.
15. Доказать правильность результатов.
16. Написать отчет о проделанной работе.

Индивидуальные задания

Создать алгоритм простой линейной структуры. Пользовательский интерфейс должен содержать следующие элементы:

1. Элемент управления Panel.
2. Два элемента управления Button.
3. Не менее трех элементов управления Label.
4. Не менее трех элементов управления TextBox.
5. Элемент управления HScrollBar или таймер.

Программная часть проекта должна включать модуль и три процедуры обработки событий:

1. Sub Button1_Click — для вызова пользовательской процедуры Sub и отображения результата в текстовых полях.
2. Sub button2_Click — для завершения работы с использованием MsgBox.
3. Sub HScrollBar1_Scroll — для выбора числа.

Процедура Sub HScrollBar1_Scroll предназначена для изменения (с помощью бегунка) параметра, который должен быть использован в процедуре модуля. Каждая лабораторная работа должна содержать открытые переменные (Public) и **определяемые пользователем Sub процедуры,** или **Function**. Процедура может быть определена в код программы в форме, однако создание процедур в модуле является более полезным, поскольку в этом случае их видимость распространяется на весь проект.

Таблица 9

1	Ввести два трехзначных числа a и b. В качестве изменяемого полосой прокрутки параметра взять число x. Используя пользовательскую функцию, найти произведение первой и последней цифр для суммы a + b + x.
2	Ввести трехзначные числа a и b. Использовать эти числа в качестве нижней и верхней границ полосы прокрутки. Выбрать число из этого диапазона и найти сумму его цифр.
3	Ввести трехзначные числа a и b. Использовать эти числа в качестве нижней и верхней границ полосы прокрутки. Выбрать число из этого диапазона так, чтобы среди его цифр не было нулей, и найти произведение этих цифр.
4	Ввести целые числа a и b так, чтобы одно из чисел было в два раза больше второго. Подобрать с помощью полосы прокрутки третье число c так, чтобы a, b, c образовывали треугольник (т.е. a + b > c, a + c > b, b + c > a). Найти площадь треугольника.
5	Ввести координаты двух точек (a, b) и (c, d). Подобрать параметр d так, чтобы стало b = d. Написать функцию для вычисления расстояния между этими точками r = sqrt((a – b)^2 + (c – d)^2). Вывести результат в текстовое окно.
6	Ввести два натуральных числа a и b, при a < b. Подобрать параметр b так, чтобы a стало больше b. Написать функцию для вычисления среднего арифметического этих чисел. Вывести это значение, а также значение среднего геометрического a и b в текстовые окна.
7	Ввести два натуральных числа a и b. Подобрать с помощью полосы прокрутки такие значения, чтобы разность между средним арифметическим и средним геометрическим была меньше единицы, а сами числа были бы не равны друг другу. Для вычисления среднего арифметического использовать функцию.
8	Пусть C и F — температуры по шкале Цельсия и Фаренгейта соответственно. Известно, что C = (5/9)*(F – 32). Для значений температуры по шкале Фаренгейта от 20 до 80 вывести в текстовое окно соответствующие им значения по шкале Цельсия. Использовать полосу прокрутки для изменения температуры по шкале Фаренгейта.
9	Ввести трехзначные целые числа a и b (a < b). Использовать эти числа в качестве нижней и верхней границ полосы прокрутки. Выбрать число из этого диапазона и найти среднее арифметическое его цифр.
10	Ввести двузначные целые числа a и b. С помощью полосы прокрутки выбрать в этом диапазоне число x и вывести его в текстовое окно так, чтобы его цифры располагались в обратном порядке (например, если x = 27, то надо вывести 72).
11	Ввести двузначные целые числа a и b. С помощью полосы прокрутки выбрать на этом диапазоне число x и вывести его в текстовое окно. Рассматривая это число как радиус, вывести в другие окна длину окружности, площадь круга и объем шара для этого радиуса.
12	Ввести координаты двух точек плоскости (a, b) и (c, d).с помощью полосы прокрутки подобрать параметр d, при котором a = d. Написать функцию для расчета расстояния между этими точками $R = \sqrt{(C - A)^2 + (D - B)^2}$. Показать результаты в текстовом поле.
13	Ввести два трехзначных числа a и b. Используя полосу прокрутки, подобрать в диапазоне от a до b число x. При помощи пользовательской функции найти сумму первой и последней цифр числа x.
14	Ввести трехзначные целые числа a и b. С помощью полосы прокрутки выбрать на этом диапазоне число x и вывести его в текстовое окно так, чтобы его цифры располагались в обратном порядке (например, если x = 234, то надо

	вывести 432). Все окна, куда выводятся числа, должны быть раскрашены в разные цвета.
15	Используя три полосы прокрутки, ввести три различных числа a, b и c соответственно. Подобрать эти числа (перемещая бегунки полос) так, чтобы значение $d = b^2 - 4 \cdot a \cdot b \cdot c$ было положительным числом. Вывести в текстовые поля найденные значения a, b и c.
16	Имеется три положительных числа. Два из них вводятся с помощью функции InputBox(), а третье должно быть получено с помощью элемента управления HScrollBar. Найти среднее арифметическое квадратов этих чисел, а также среднее геометрическое любых двух из них. Среднее геометрическое двух чисел равно квадратному корню из них.
17	Найти величину определителя порядка $2\begin{vmatrix} a & b \\ c & d \end{vmatrix} = a \cdot d - b \cdot c$. Переменные вводятся с помощью текстовых окон, а величина переменной c должна быть найдена при использовании элемента HScrollBar1, так что значение определителя будет больше нуля.
18	Используя элемент управления HScrollBar control, найти четырехзначное число, все цифры которого образуют возрастающую последовательность. Записать каждую из этих цифр в отдельное текстовое окно.
19	Последовательность чисел, в которой каждый очередной член получается добавлением одного и того же фиксированного числа, называется арифметической прогрессией. Используя элемент HscrollBar, найти четырехзначное число, цифры которого образуют арифметическую последовательность.
20	Поместить две полосы прокрутки в середину формы. Установить значение одной из них равным X и другой $Y(0 < X < Y < 100)$. Используя процедуру Sub, найти сумму первой цифры числа X и последней цифры числа Y.
21	Пусть A и B — трехзначные натуральные числа. Используя полосу прокрутки, подобрать в диапазоне от A до B число X. Создать функцию, которая определяет сумму первой и последней цифр числа X и умножает эту сумму на цифру, находящуюся в середине числа X.
22	Используя пример выполнения индивидуального задания № 2, создать проект для вычисления за фиксированное время данного арифметического выражения $Y = \frac{12 \cdot x + \sqrt{23 + x \cdot 17}}{x^2 + \sqrt{7 + x}} + \sqrt{93 + 24 \cdot x^3}$.
23	Используя пример выполнения индивидуального задания № 2, создать проект для вычисления за фиксированное время корней квадратного уравнения $10x^2 - 29x + 10 = 0$.
24	Используя пример выполнения индивидуального задания № 3, создать лабиринт, отличающийся от лабиринта в примере № 3; при этом кнопка Start должна находиться в левом нижнем углу формы, а метка Finish — в правом верхнем углу. Кроме этого, звук должен включаться и при достижении метки Finish.
25	Используя пример выполнения индивидуального задания № 3, создать лабиринт, отличающийся от лабиринта в примере № 3; кроме этого, после каждого касания стенки должен раздаваться звук и появляться окно, в котором указывается, сколько еще остается касаний стенок с сохранением возможности дальнейшего прохождения лабиринта.

Содержание отчета:

1) название работы;
2) титульный лист;
3) индивидуальное задание;
4) программные коды модуля и процедур;
5) форма проекта с элементами управления;
6) полученные результаты;
7) доказательство правильности программ.

Пример выполнения индивидуального задания № 1

Задание на разработку проекта. Создать проект для решения следующей задачи:

Используя элемент управления HscrollBar, отобразить в текстовом поле трехзначное число. Создать модуль и поместить внутрь подпрограмму процедуры, которая реализует алгоритм выделения цифр заданного числа. Все три цифры должны помещаться в текстовые поля.

Решение

1. Создать новый проект.
2. Создать графический интерфейс пользователя.
3. Установить свойства элементов управления.
4. Написать программу процедуры Sub.
5. Запустить Visual Basic и получить результаты.
6. Выполнить анализ результатов.

Новым элементом в этом проекте является создание и использование модуля. Для этого надо выбрать команду Add New Item («Добавить новый элемент») в меню Project и затем шаблон Module («модуль») (рис. 1.2.2).

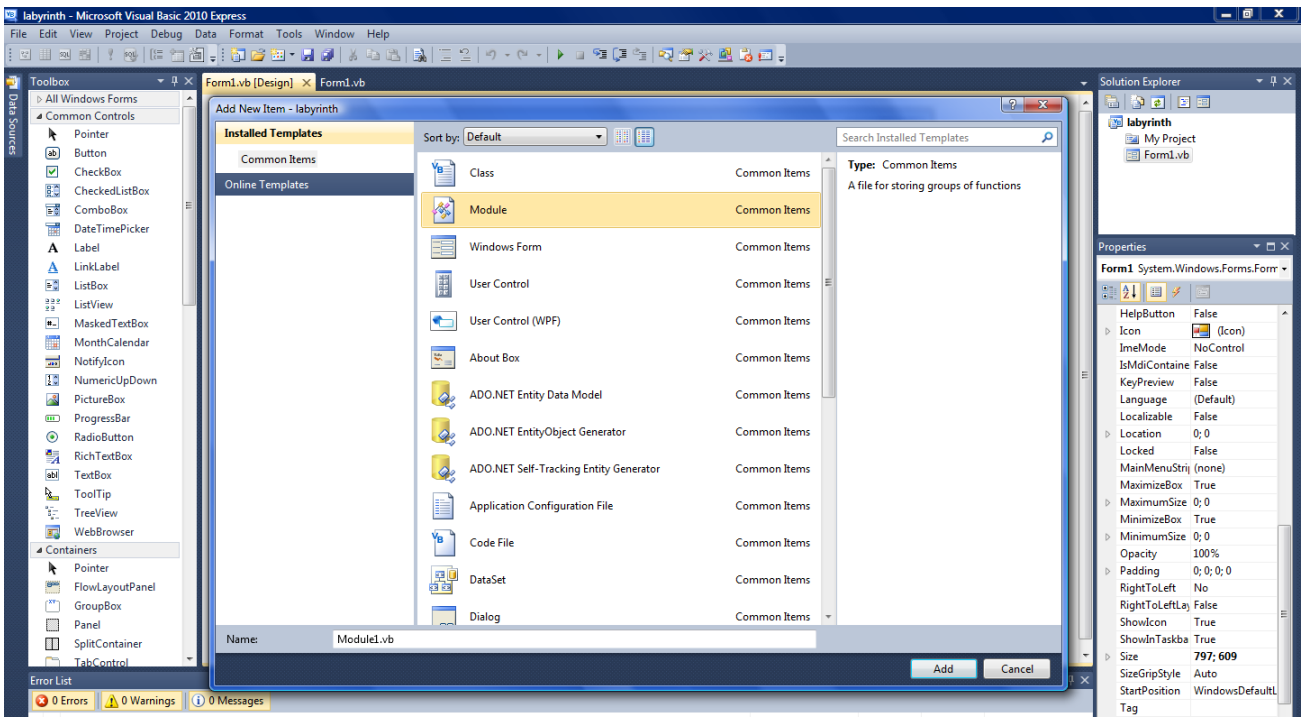


Рис. 1.2.2

Выбрать модуль в Solution Explorer. Поместить операторы процедуры между ключевыми словами *Sub* и *End Sub* и присвоить ей имя **digit** ( ). Для отделения каждой цифры x из числа h надо использовать оператор **h mod x** , который находит остаток от деления h на x, например: 125 mod 10 = 5 и 125 mod 100 = 25. Для определения всех трех цифр необходимо написать следующий код:

a = (h Mod 1000) \ 100
b = (h Mod 100) \ 10
c = h Mod 10.

Блок-схема алгоритма для процедуры Sub Button1_Click (рис. 1.2.3).

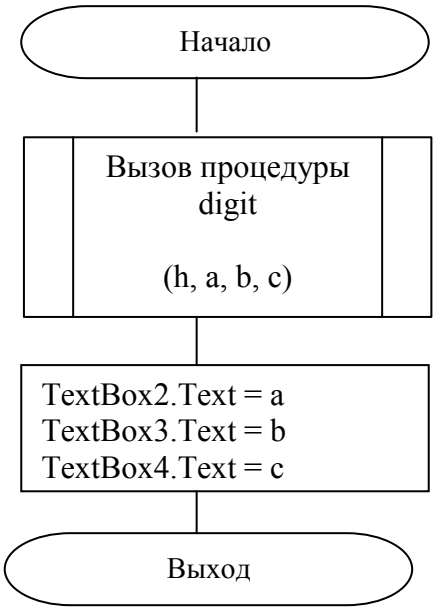


Рис. 1.2.3

Событийная процедура для командной кнопки **Button2_Click** включает в себя диалоговое окно **MsgBox**, в котором помещается информация о завершении программы, либо ее продолжении (Do you want exit?). Для этого надо написать следующие строки кода:

```
Dim n As Integer
  n = ("Do you want exit?", vbYesNo)
     If n = vbYes Then
        End
          End If
```

При выборе дизайна проекта следует создать следующий графический пользовательский интерфейс (рис. 1.2.4).

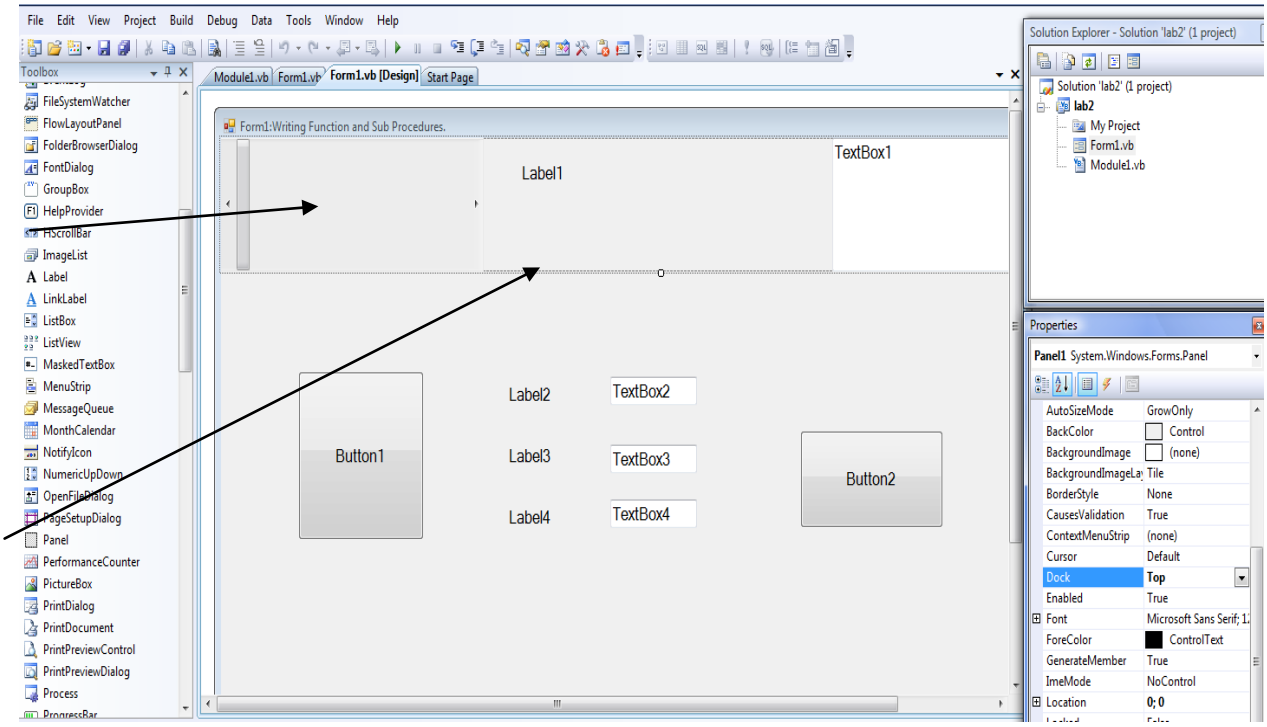


Рис. 1.2.4

Свойства для формы и ее объектов:

Таблица 10

Объект	Свойство	Значение
Panel	Dock	Top
HScrollBar1	Maximum Minimum SmallChange Dock	1000 100 1 Left
Button1	Text	"Показать цифры числа"
Button2	Text	"Выход или продолжение"
Label1	Text Dock	"Перемещая бегунок подберите число" None
Label2	Text	"первая цифра"
Label3	Text	"вторая цифра"
Label4	Text	"третья цифра"
TextBox1	Multiline Dock	True Right
TextBox2	Multiline	False
TextBox3	Multiline	False
TextBox4	Multiline	False
Form1	Font Text	Microsoft Sans Serif; 14,25 pt "Writing Function and Sub Procedure"

Программы на Visual Basic 2010. В меню **Project** щелкнуть на команде **Add New Item** «Добавить новый элемент», затем выбрать в окне шаблонов (Templates) Module и нажать кнопку Add («Добавить»). Новый модуль появится в редакторе кода. Напечатать следующий код процедуры **digit** между **Module Module1** и **End Module** (рис. 1.2.5):

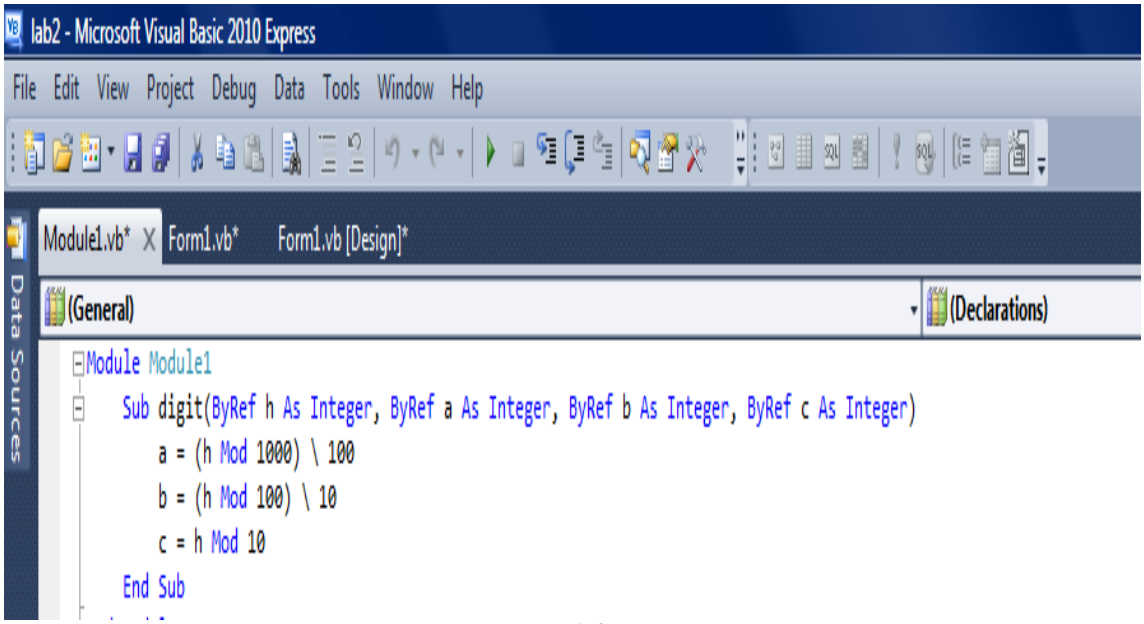


Рис. 1.2.5

Далее надо перейти в редактор кода для формы. Для этого либо щелкнуть на вкладке Form1.vb, либо перейти в Дизайнер (View -> Designer) и щелкнуть два раза мышью на кнопке Button1. Все три событийные процедуры приводятся ниже. Переменные целого типа Integer h, a, b, c объявлены как глобальные, поскольку они декларированы как Public, рис. 1.2.6. В данной работе опция Ontion Strict по умолчанию установлена в Off (отключена), и все преобразования данных производятся автоматически. В подобных случаях пользователь должен самостоятельно отслеживать корректность согласования типов, что не вполне удобно для сложных задач.

```
Button1                                                    Click
Public Class Form1
    Public h, a, b, c As Integer
    Private Sub HScrollBar1_Scroll(ByVal sender As System.Object, ByVal e As System.Windows.Forms.ScrollEventArgs) Handles HScrollBar1.Scroll
        TextBox1.Text = HScrollBar1.Value
        h = Val(HScrollBar1.Value)
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        digit(h, a, b, c)  'Calling a Sub Procedure
        TextBox2.Text = a
        TextBox3.Text = b
        TextBox4.Text = c
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
        Dim n As Integer
        n = MsgBox("Do you want exit ?", vbYesNo)
        If n = vbYes Then
            End
        End If
    End Sub

End Class
```

Рис. 1.2.6

Следует заметить, что передача аргументов при вызове процедуры digit (h, a, b, c) осуществляется позиционно, т.е. если при объявлении процедуры в модуле сначала записывается шаг h, а потом длины сторон a, b, c, то и при ее вызове в событийной процедуре кнопки Button1_Click соблюдается этот же порядок. Однако можно вызвать эту процедуру и так, чтобы аргументы передавались по имени. Этот вызов может быть например, таким:

digit(a:= a, h:=h, c:=c, b:=b)

При вызове одной и той же процедуры можно использовать одновременно оба эти способа, но тогда параметры, передаваемые позиционно, должны быть указаны первыми и в том порядке, в котором они заданы при определении процедуры. Например, процедуру можно вызвать так:

digit(h, a, c:=c, b:=b)

После того как создан интерфейс и напечатаны все процедуры, необходимо запустить созданное приложение. Для этого надо в опции Debug щелкнуть на Start Debugging. Запустить приложение можно также простым нажатием клавиши **F5**. После запуска на экране должен появиться созданный интерфейс. Используя бегунок полосы прокрутки, расположенный в верхнем левом углу формы, надо подобрать число, например 625, и нажать на кнопку «Показать цифры числа», рис. 1.2.7.

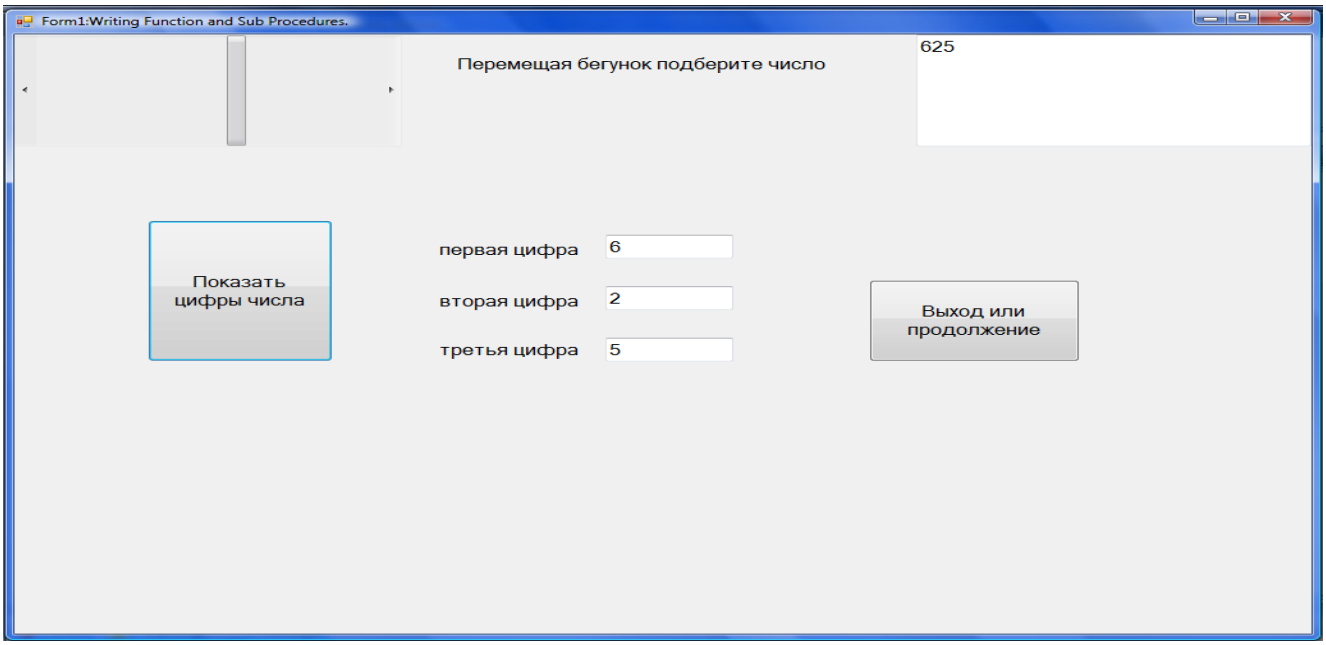


Рис. 1.2.7

После нажатия кнопки «Выход или продолжение» на экране появится диалоговое окно MsgBox( ), в котором надо выбрать «Да» или «Нет», рис. 1.2.8:

Если щелкнуть на «Да», то приложение заканчивает свою работу, если на «Нет», то можно снова перемещать бегунок для выбора нового числа.

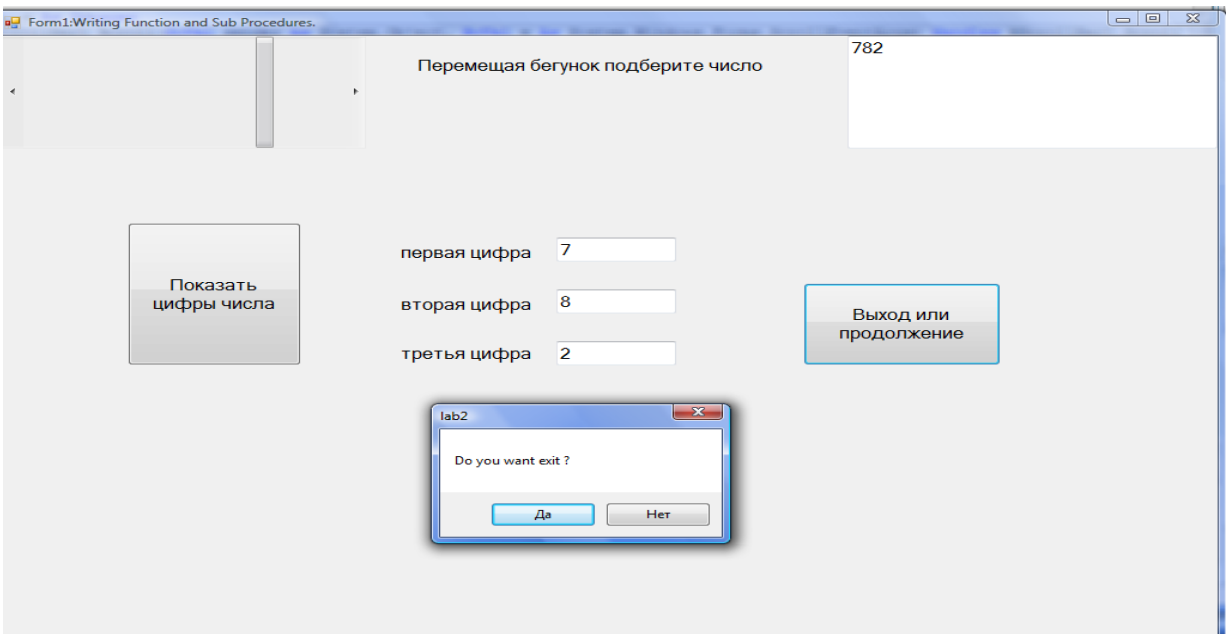


Рис. 1.2.8

Пример выполнения индивидуального задания № 2

Задание на разработку проекта. Создать проект для решения следующей задачи:

Отобразить на форме арифметическое выражение, зависящее от *x*, с помощью элемента *PictureBox*:

$$Y = \frac{2x^2}{9 + \frac{(x-2.5)(x-4.7)}{x+4+\frac{x}{x+7}}}$$

Используя элемент управления *NumericUpDown*, предназначенный для ввода числовой информации и представляющий собой текстовое поле с двумя кнопками для уменьшения или увеличения числа, помещаемого в это поле, — установить некоторое значение *x*. Установить также таймер, который обрабатывает данные системных часов. Для запуска таймера в момент, когда необходимо начать вычисление выражения, использовать метод Start(). Количество остающихся для вычисления выражения секунд должно отображаться в свойстве Text метки, которой необходимо присвоить имя *timeLabel*.

После запуска приложения и нажатия кнопки Start пользователь должен вручную вычислить выражение и занести его в тестовое окно TextBox1. Если результат верный, то программа выдаст об этом сообщение, если нет, или пользователь не уложился в заданное время, то об этом также будет выдано соответствующее сообщение.

Создать модуль и поместить внутрь подпрограмму-функцию *Function formula*(*x*), которая позволяет вычислить заданное выражение.

Решение

1. Создать новый проект.
2. Создать графический интерфейс пользователя.
3. Установить свойства элементов управления.
4. Написать программы для всех процедур Sub.
5. Запустить Visual Basic и получить результаты.
6. Выполнить анализ результатов.

Создадим следующий графический пользовательский интерфейс (рис. 1.2.9). На форме имеется единственная кнопка управления Button, которой присвоено имя startButton, а ее свойству Text присвоено значение Start. Если свойству элемента Enabled присваивается значение False, то он перестает быть активным; чтобы он был активен, это свойство должно быть True.

На форму также надо установить Timer1, он отображается в нижней части конструктора формы. При запуске приложения этот элемент не виден. Если необходимо, чтобы время менялось через одну секунду, в свойстве таймера Interval должно быть значение 1000 (интервал активизации таймера задается в миллисекундах).

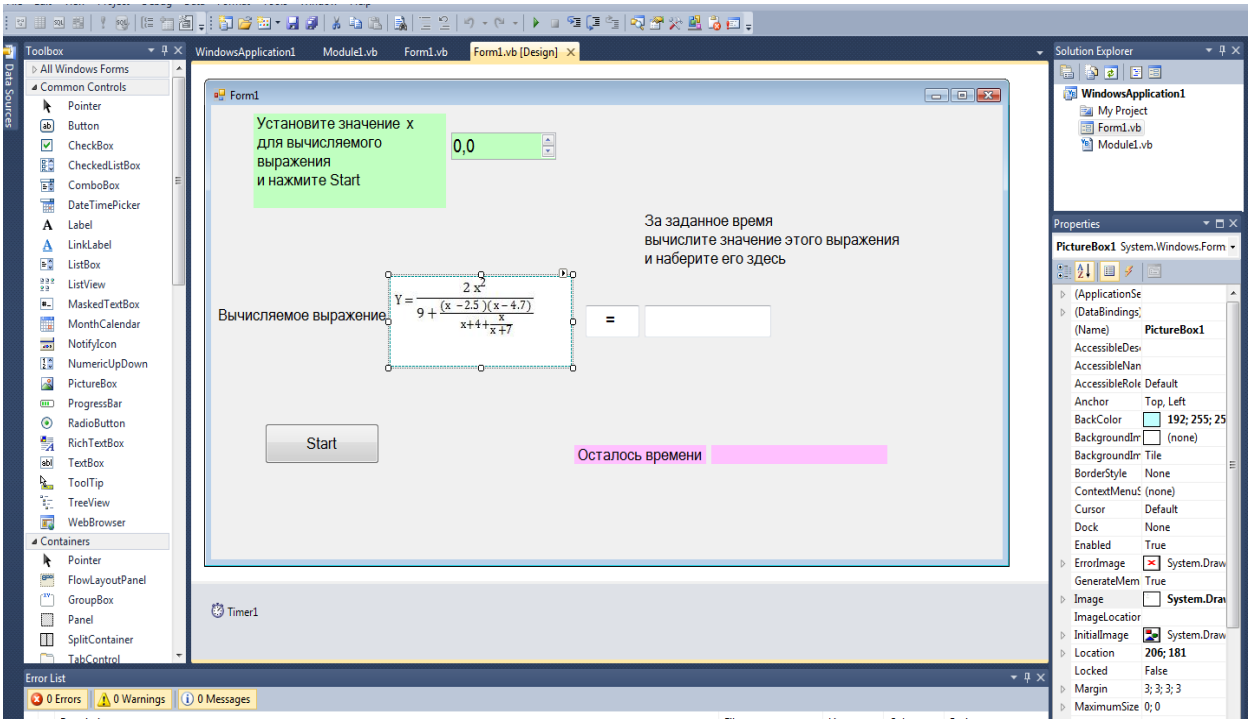


Рис. 1.2.9

Программные коды модуля и класса формы.

Код модуля:

```
Module Module1

    Public Function formula(ByRef x As Single)
        Return 2 * x ^ 2 / (9 + (x - 2.5) * (x - 4.7) / (x + 4 + x / (x + 7)))
    End Function

End Module
```

Код класса формы Form1:

```
Public Class Form1
    Private x, y, yk As Single
    Private timeLeft As Integer
```

```
    Private Sub startButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles startButton.Click
        Start()
        startButton.Enabled = False      'Элемент становится недоступным
    End Sub

    Public Sub Start()
        timeLeft = 25  'Задается 25 секунд, в течение которых в текстовое окно должно
быть  записано значение выражения
        timeLabel.Text = "25 секунд"
        Timer1.Start()
    End Sub

    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Timer1.Tick
        If timeLeft > 0 Then
            timeLeft -= 1
            timeLabel.Text = timeLeft & "сек."
        Else
            Timer1.Stop()
            x = CSng(NumericUpDown1.Value)
            y = CSng(formula(x))   'Обращение к функции, определенной в модуле
            yk = CSng(TextBox1.Text)
            If Math.Abs(y - yk) < 0.1 Then
                MsgBox("Найден правильный результат")
            Else
                timeLabel.Text = "Время закончилось!"
                MsgBox("Вы не нашли ответ за отведенное время" & vbCrLf & "Ответ =" & y)
            End If
            startButton.Enabled = True   'Активизация командной кнопки
        End If
    End Sub

End Class
```

Результат работы приложения представлен на рис. 1.2.10.

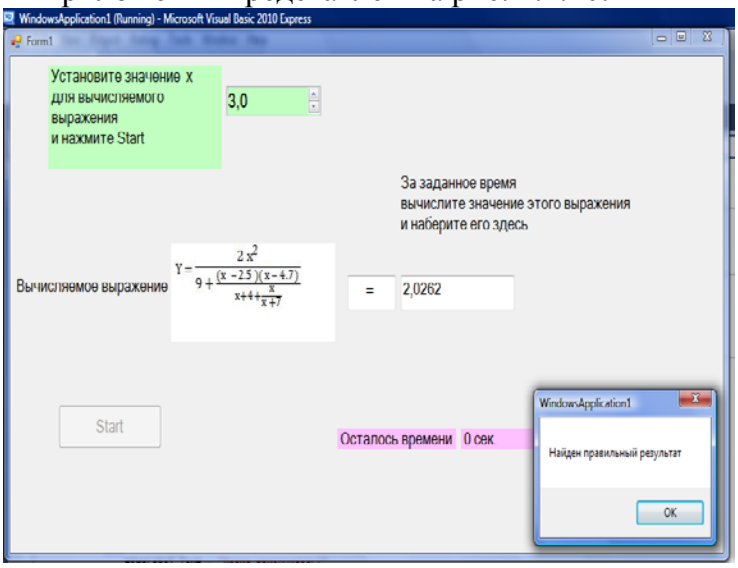


Рис. 1.2.10

Пример выполнения индивидуального задания № 3

Задание на разработку проекта. Создать проект для решения следующей задачи:

Используя метки (label), построить на форме лабиринт. Также поместить на форме элемент Button с текстом Start и еще одну метку с текстом Finish, как показано на рис. 1.2.11.

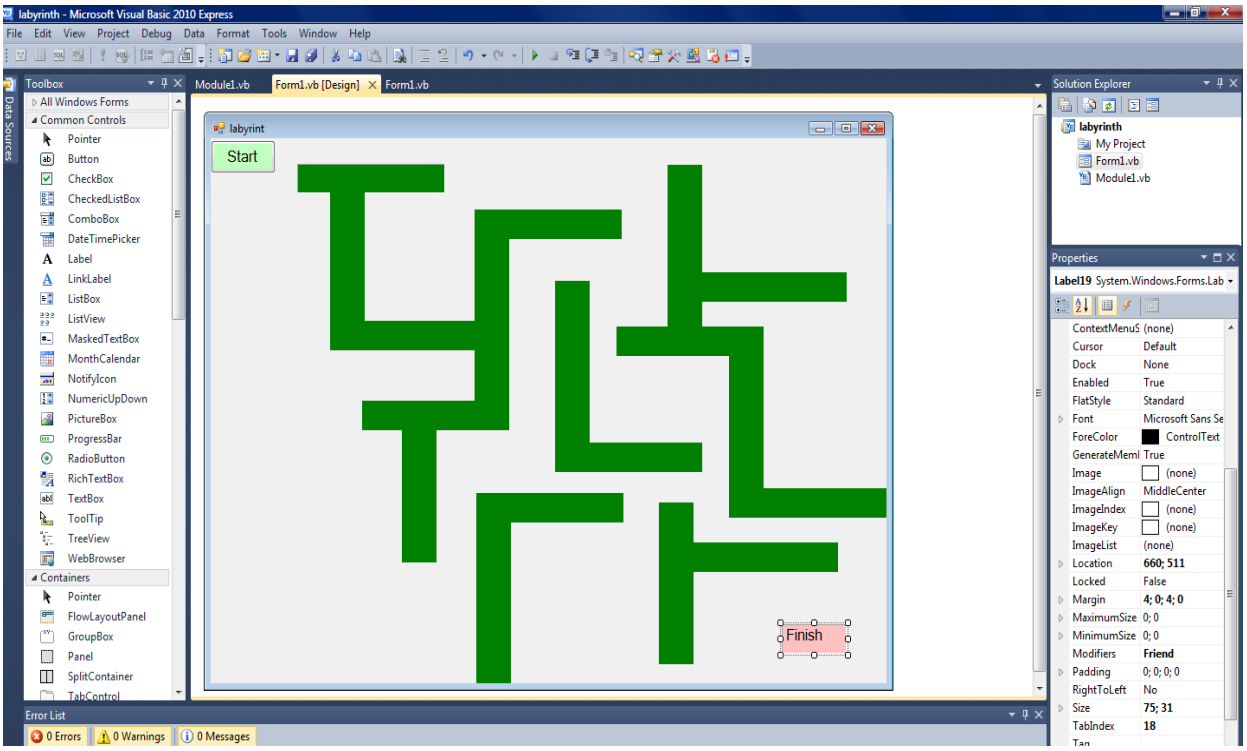


Рис. 1.2.11

Требуется переместить указатель мыши от кнопки Start до метки Finish, при этом допускается касание стенок лабиринта не более заданного числа раз. Необходимо также создать модуль, где необходимо разместить процедуру route() для ввода допустимого числа касания курсором стенок лабиринта, процедуру disp() для вывода в модульное окно результата перемещения по лабиринту.

Решение

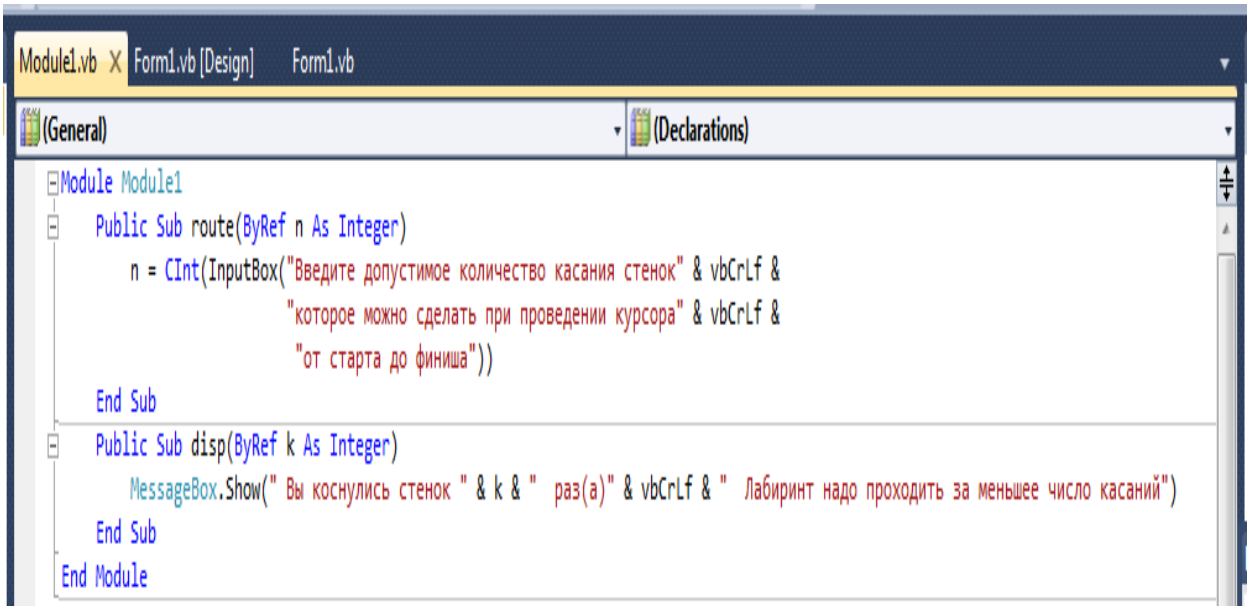
В классе формы создадим следующие процедуры:

MoveToStart() для установления курсора в начальное положение, которое помещается в позицию (0, 0) кнопки Start.

wall_MouseEnter() для включения звука (объект startSoundPlayer позволяет запускать файл C:\Windows\Media\chord.wav) и перемещения курсора к стартовой позиции.

wallFinish() для окрашивания экрана в зеленый цвет и выдачи сообщения о прохождении лабиринта.

Код модуля:



Код класса формы Form1:

```
Public Class Form1
    Private startSoundPlayer = New
Sysem.Media.SoundPlayer("C:\Windows\Media\chord.wav")
    Public k As Integer = 0
    Public n As Integer = 0

    Private Sub MoveToStart()
        Dim startingPoint = Button1.Location
        startingPoint.Offset(0, 0)
        Cursor.Position = PointToScreen(startingPoint)
    End Sub

    Private Sub wall_MouseEnter() Handles Label2.MouseEnter, Label3.MouseEnter,
Label4.MouseEnter, Label5.MouseEnter, Label6.MouseEnter, Label7.MouseEnter,
Label8.MouseEnter, Label9.MouseEnter, Label10.MouseEnter, Label11.MouseEnter,
Label12.MouseEnter,Label13.MouseEnter, Label14.MouseEnter, Label15.MouseEnter,
Label16.MouseEnter, Label17.MouseEnter, Label18.MouseEnter
        If n <> 0 Then
            If k < n Then
                startSoundPlayer.Play()
                k += 1
                MoveToStart()
            Else
                disp(k)
                Close()
            End If
        End If
    End Sub
```

```
    Private Sub wallFinish() Handles Label19.MouseEnter
        If n <> 0 Then
            Me.BackColor = Color.Green
            MessageBox.Show(Вы успешно прошли лабиринт")
            Close()
        End If
    End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        route(n)
        MoveToStart()
    End Sub

End Class
```

Результат работы приложения (рис. 1.2.12):

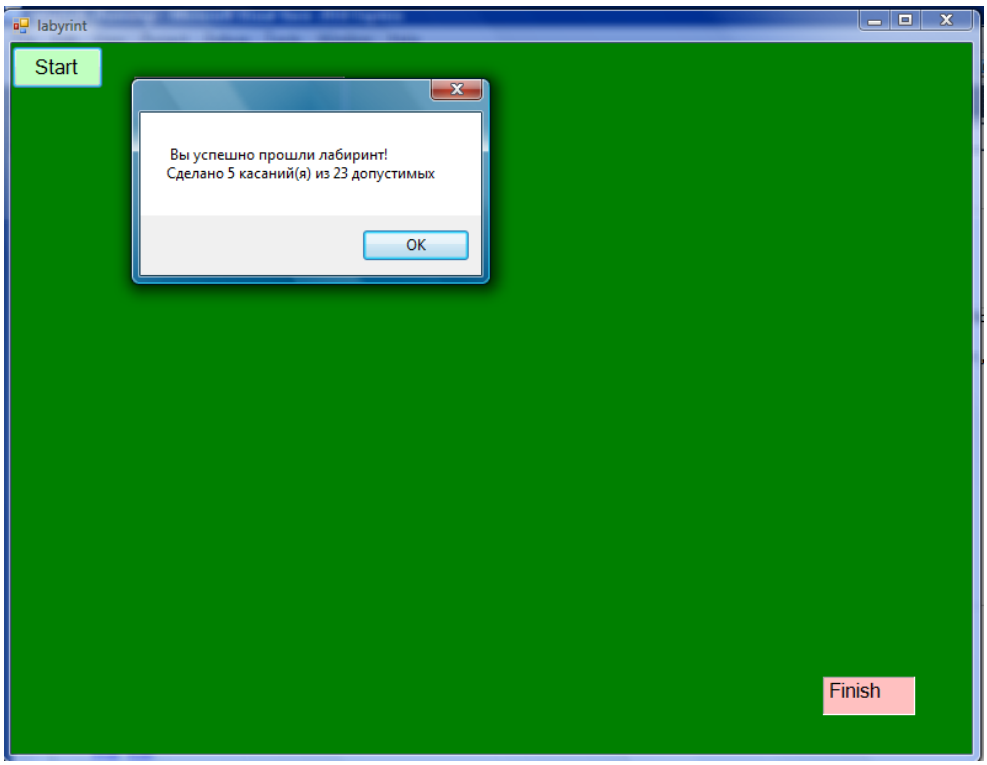


Рис. 1.2.12

Т е м а 3. Введение в объектно-ориентированное программирование. Создание классов и объектов. Программирование алгоритмов линейной структуры

Изучение этой темы позволит:

- понимать терминологию объектно-ориентированного программирования;
- создавать классы, их методы и свойства;
- создавать в классе новое свойство;
- создавать в классе новый метод;
- работать с объектами и ссылками на объекты;
- понимать, что такое конструктор класса.

В последнее время становится необходимым при создании практических программ умение понимать и использовать технику объектно-ориентированного программирования (ООП). Изменения, связанные с ООП, набирают силу: в последних версиях Visual Basic 2010 и Visual Basic 2010 перестали отставать от языка C++, поскольку теперь они также поддерживают *наследование* — механизм, при котором один класс дает возможность приобрести интерфейс и поведение другого класса.

Классы являются стандартными блоками в программах Visual Basic. При изучении классов имеется потенциальная опасность непонимания из-за перегрузки терминологии, вместе с тем в данной работе рассматривается материал, в котором предпринимается попытка свести к минимуму эти трудности.

Класс в Visual Basic 2010 — это представление или описание, которое определяет структуру какой-либо сущности реального мира. Класс является кодом, определяющим объект; используя класс, можно создать любое количество объектов, которые называются **экземплярами** (instance) класса.

Объект — это код, создаваемый на основе абстракций рассматриваемой сущности. Это могут быть существующие процессы или явления, или любые технические системы, или отношения, определяющие их поведение. Например, объектом может быть *Лектор*, созданный на основе реального лектора, такого как преподаватель Московского университета, или объектом может быть *Файл*, представляющий файл C:\bd.mdb на жестком диске компьютера.

Visual Basic 2010 поддерживает все четыре основные определяющие концепции, при которых язык становится полностью объектно-ориентированным.

Абстракция — метод решения задачи, при котором объекты разного рода объединяются общим понятием, а затем рассматриваются как элементы единой категории. Абстракция — это просто возможность языка создавать код какого-либо понятия в виде «черного ящика» для реализации этого понятия внутри программы. Например, можно создать объект *Customer* («*Клиент*»), который будет являться абстрактным представлением клиентов реального мира.

Инкапсуляция — техника, при которой несущественная с точки зрения интерфейса объекта информация прячется внутри него. Основная идея инкапсуляции состоит в разделении интерфейса и реализации, т.е. можно создавать интерфейс (методы, свойства, события в классе), и пока этот интерфейс существует, приложение может взаимодействовать с объектами. Это будет выполняться, даже если полностью переписать код внутри какого-то метода — таким образом, интерфейс не зависит от реализации.

Полиморфизм — свойство, позволяющее использовать один и тот же интерфейс для различных действий; полиморфной переменной, например, может соответствовать несколько различных методов.

Наследование представляет собой идею, что любой класс может получить интерфейс и поведение созданного ранее класса.

В предыдущих работах под словом «интерфейс» понимался графический интерфейс, однако этот термин имеет более широкое значение. **Интерфейс** определяется как множество методов (Sub- или Function-процедур), свойств (Property-процедур), событий и полей, которые объявлены как Public.

Например, имеется следующий код некоторого класса:

```
Public Function Routine () As Single
…
End Function,
```

этот метод объявлен с помощью ключевого слова Public, поэтому является частью интерфейса и может быть вызван приложением, использующим объект этого класса.

Если же методы или свойства объявлены как Private, они не будут частью интерфейса и не смогут вызываться программами, написанными для использования какого-либо объекта; например, метод Schedule () может быть вызван только в коде класса, где он объявлен

```
Private Sub Schedule ()
…
End Sub
```

В тоже время можно создать еще один метод Routine ():

```
Public  Sub Routine ()
Schedule ()
End Sub,
```

который позволит вызывать метод Schedule () за пределами класса, где он объявлен. В этом случае удается вызвать Private метод Schedule () изнутри Public метода.

Код внутри метода называется реализацией (implementation).

Интерфейс используется для доступа к данным и поведению объекта. Клиентское приложение можно рассматривать как объект, подобный черному ящику, с доступом к нему только через интерфейс. Это ключевая идея объектно-ориентированной концепции, и именно ее называют инкапсуляцией. Идея состоит в том, что любая программа, которая использует данный объект, не будет иметь прямого доступа к его поведению или данным; более точно можно сказать, что программа скорее имеет возможность использовать интерфейс этого объекта.

Инкапсуляция является синтаксическим инструментом — она дает возможность коду выполняться без изменений. Однако это не семантика, означающая, что если код работает, то это обязательно нужно.

Важной частью объекта являются его данные. Каждый экземпляр класса является абсолютно идентичным с точки зрения его интерфейса и его реализации; единственное, что может изменяться — это данные внутри этого конкретного объекта.

Поля являются переменными, которые объявляются с тем, чтобы они были доступны всем кодам в пределах класса. Они будут доступными для каждого индивидуального объекта, при запуске приложения. Каждый объект получает собственный набор данных — в основном, каждый объект получает собственную копию из полей.