Лабораторная работа №1

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И МЕХАТРОНИКИ

Кафедра информационных технологий и мехатроники

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

по проведению лабораторных работ по дисциплине «Алгоритмизация и программирование»
для студентов специальности 6.050101 “Компьютерные науки”

Разработчик - доцент кафедры информационных технологий и мехатроники
кандидат технических наук, старший научный сотрудник
Тимонин Владимир Алексеевич

Харків 2015

Лабораторная работа №1
Исследование возможностей интегрированной среды разработки Visual C# для создания консольных приложений.

Цель работы – исследовать возможности интегрированной среды разработки Visual Studio 2010 и получить практические навыки по созданию консольных приложений ОС Windows.

1. Теоретические сведения

Интегрированная Среда Разработки Visual Studio 2010 (Integrated Development Environment - IDE, в дальнейшем будем использовать для нее аббревиатуру ИСР) — это среда, позволяющая облегчение процесса создания программ и содержащая все необходимое для проектирования, запуска и тестирования при ложений (редактор кодов, отладчик, инструментальные панели, редактор изображений, инструментарий баз данных и др.). ИСР предоставляет возможность расширять меню, включая в него необходимые дополни тельные программы, в том числе и собственные.

1.1. Структура главного окна ИСР Visual Studio 2010

Чтобы начать работу в Microsoft Visual Studio, надо запустить Microsoft Visual Studio - сделать щелчок на кнопке **Пуск** и в меню **Все програм мы -> Microsoft Visual Studio 2010** выбрать **Microsoft Visual Studio 2010.** На экране появится ос новное окно ИРС, представляющее собой совокупность нескольких окон (рис.1).

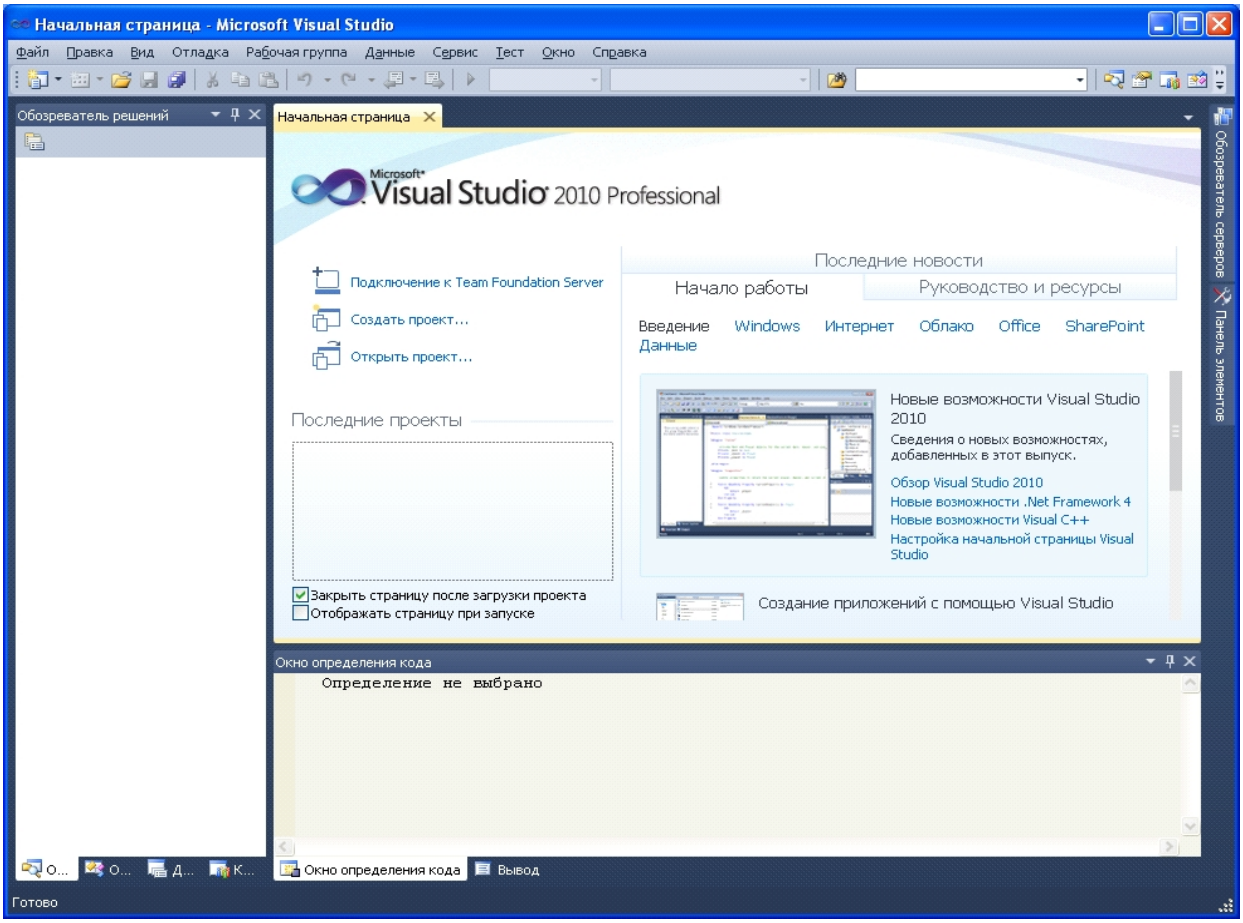

Рис. 1. Вид главного окна ИРС после загрузки Visual Studio

Самое верхнее окно имеет заголовок **Microsoft Visual Studio**, которое отражает название среды разработки и имя нового решения, из которого будет получена работающая программа На строке заголовка проекта находятся кнопки свертывания, восстановления и закрытия окна.

Под заголовком размещается строка главного меню, которая предоставляет доступ ко всем функциям и командам среды разработки. Ее состав частич но зависит от варианта Visual Studio. При вызове этих команд (**Файл, Правка, …**) открываются так называемые «выпадающие меню», представляющие собой набор команд.

Справа от названия элемента меню может находить ся комбинация клавиш быстрого доступа для вывода элемента меню, используя клавиатуру. В зависимости от ситуации в среде некоторые пункты меню недоступны для использования (отображаются светло-серым цветом). Напри мер, нельзя воспользоваться командой сохранения файла, если нет открытых файлов.

Главное меню состоит из следующих подменю:

Файл - позволяет создать новое решение, от крыть ранее созданное решение, сохранить решения проекты или формы в фай лах с заданными именами;

Правка - позволяет выполнять обычные для приложений операции обмена с буфером Clipboard, осуществлять поиск и контекстные замены в коде приложения, которые свойственны большинству известных тексто вых редакторов;

Вид - позволяет вызывать на экран различные окна, необходимые для проектирования;

Проект - позволяет добавлять и убирать из проекта элементы, задавать опции проекта;

Построение - позволяет работать с решениями и проектами (создание, модификация, удаление и др.);

Отладка – дает возможность выполнять проект в нормальном или отладочном режимах, продвигаясь по шагам, останавливаясь в указанных точках кода, просматривая значения переменных и т.д.;

Рабочая группа – позволяет работать с группами проектов;

Данные – позволяет использовать инструментарий для работы с базами данных;

Формат - позволяет выравнивать группы размещенных на форме компонентов по размерам и местоположению;

Сервис – включает ряд подразделов, позволяющих настраи вать ИСР и выполнять различные вспомогательные программы, работать с программами, конфигури рующими базы данных и т.д. Кроме того, в это меню можно сами включить любые разделы, вызывающие те или иные приложения, и таким образом расши рить возможности главного меню ИРС, приспособив его для своих задач;

Тест – позволяет работать с тестовыми проектами;

Окно - по зволяют ориентироваться среди массы окон, обычно одновременно открытых в процессе проектирования и переключаться в нужное окно;

Справка - содержит разделы, помогающие работать со встроенной в Visual Studio справочной системой.

Ниже строки главного меню размещаются инструментальные панели, содержащие кнопки быстрого вызова, дублирующие некоторые наиболее часто используемые ко манды меню. Все эти кнопки имеют всплывающие подсказки (при наведении курсора мыши на кнопку появляется подсказка о том, для чего предназначена кнопка). Рядом с такими кнопками могут быть дополнительные кнопки для раскрытия списка значений основной кнопки. Так как все кнопки не помещаются в отведенное им место на рабочем столе, то они свернуты в небольшие полосы с кнопками их развертывания.

В центре окна расположена рабочая область, в которой в виде вкладок будут открываться файлы. На данный момент в рабочей области открыта HTML-страничка, в которой можно создать новый или открыть существующий проект, а также почитать последние новости из мира разработки.

В Visual Studio любая программа заключается в проект. Проект — сово купность файлов, необходимых для создания программы. Он обладает определенными свойствами (например, платформа и язык, для которого создан проект) и может содержать файлы с исходным кодом программы, который необходимо скомпилировать в исполняемый файл Проекты могут объединяться в решения.

Для того чтобы приступить к созданию программы (консольного приложения) в Visual Studio или, как принято говорить, начать работу над проектом, надо:

1. В меню **Файл** окна **Microsoft Visual Studio** выбрать команду **Файл->Создать->Проект** или на начальной странице выбрать пункт «**Создать проект**» или с помощью кнопки быстрого вызова «**Создать проект**» (первая слева в строке, расположенной ниже строки главного меню).

2. В открывшемся окне **Создать проект** (рис. 2) раскрыть список **Visual C#** и выбрать тип приложения — **Windows**.

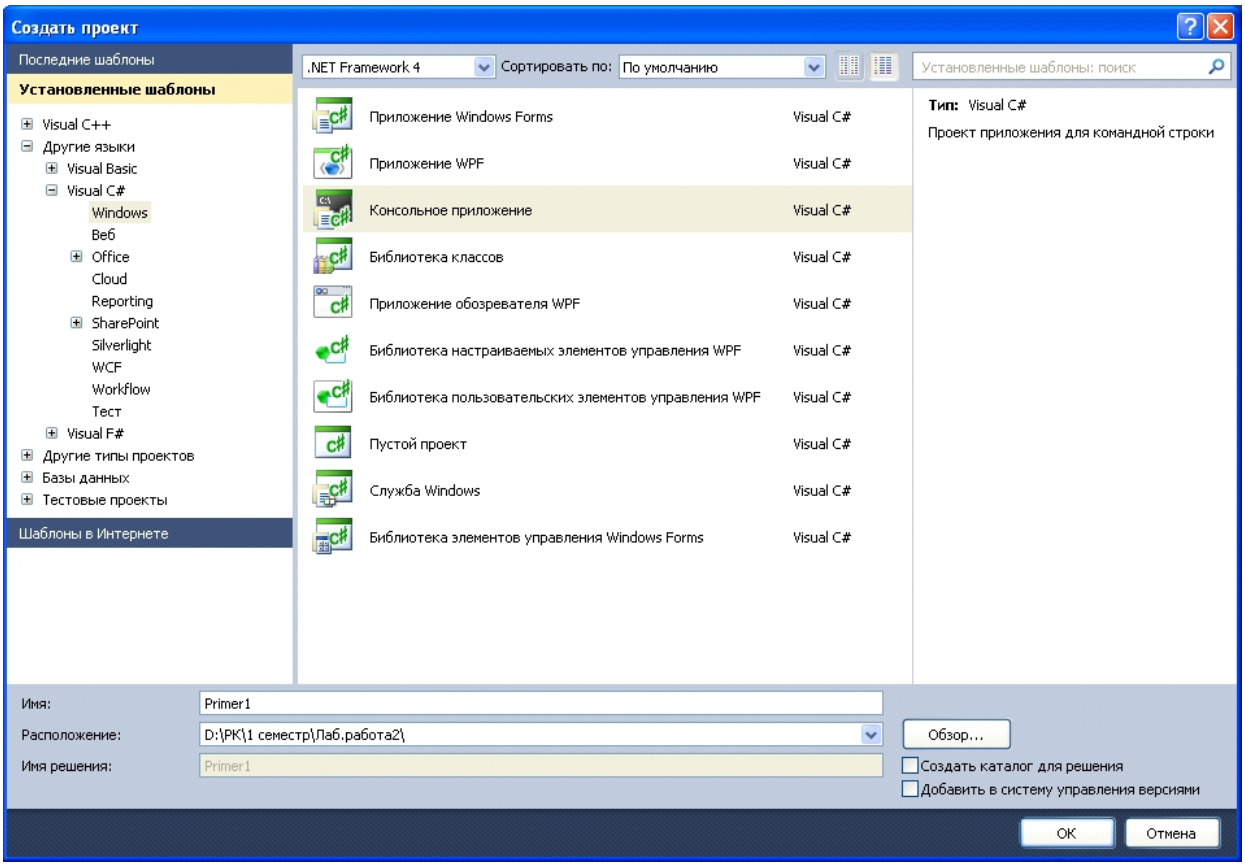

Рис. 2. Вид окна «Создать проект»

3. Выбрать вид приложения — **Консольное приложение Visual C#.**

4. В поле **Имя** ввести имя проекта (по умолчанию это ConsoleApplication1), в поле **Расположение** ввести папку размещения проекта (по умолчанию это C:\documents and settings\User\Мои документы\Visual Studio 2010\Projects\, гдеUser — имя пользователя в систе ме) и нажать кнопку **ОК.**

В результате описанных действий будет создан проект консольного приложенияVisual C# (установленный флажок **Создать каталог для решения** сбросить). Имя папки проекта определяет содержимое поля **Имя** (имя проекта).

Окно среды Microsoft Visual Studio 2010 в начале работы над новым проектом приведено на рис. 3. В заголовке окна отображается имя проекта, над кото рым в данный момент идет работа.

В верхней части окна находится строка меню и область отображения панелей инструментов. По умолчанию в области отображения панелей инструментов выводится панель **Стандартная**. Чтобы сделать доступными другие па нели инструментов, надо выбрать команду **Сервис->Настройка** и в раскрывшем ся списке сделать щелчок на имени нужной панели (например, на рис. 3 показана дополнительно панель «**Макет**»).

Центральную часть окна Visual C# занимает окно редактора кода (отображена закладка **Program.cs**). В нем находится заготовка консольного приложения

Заготовка состоит из 2-х разделов: раздел имен пространств (группа операторов **using**) и раздел главной функции, оформленной как класс **Program** (**class Program**) и определенной в пространстве с именем **Primer1** (**namespace Primer1**), который состоит из заголовка главной функции **static void**

Main(string[] args) и тела, ограниченного фигурными скобками {}.

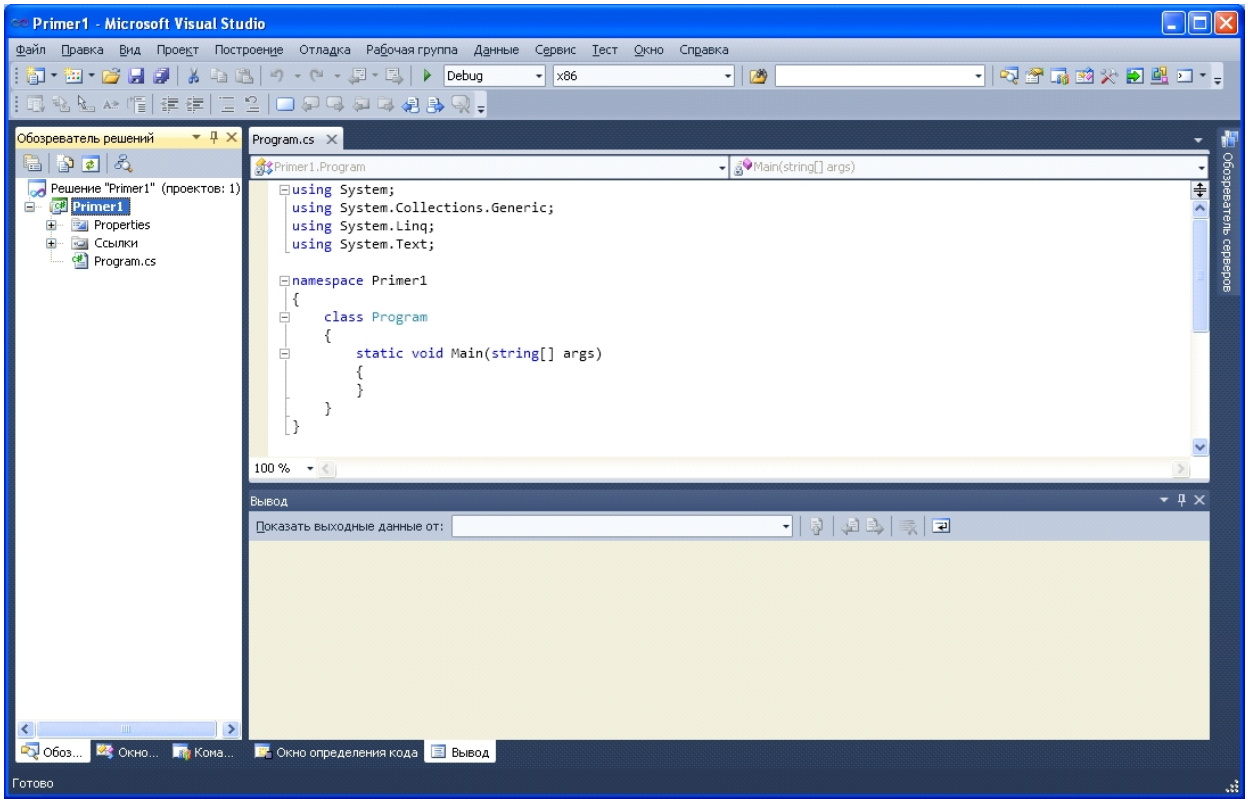

Рис. 3. Вид окна «Primer1 - Microsoft Visual Studio»

Visual Studio предоставляет множество окон, которые отображают информацию, необходимую для создания приложений. Некоторые окна (**Обозреватель решения, Обозреватель серверов** и др.) отображаются по умолчанию. Полный список доступных окон расположен в главном меню (пункт **Вид**).

В левой части главного окна размещается окно **“Обозреватель решения”** (рис. 3), в котором отображается структура файлов проекта и среди них – программный модуль **Program.cs**

Когда формируется заготовка консольного приложения, то при задании имени (**Имя**) приложения формировалось и некое поле **Имя решения**, так как среда Visual Studio оформляет создаваемое прило жение в виде двух контейнеров, вложенных один в другой. Один (главный контейнер) называется **Решение**, а другой — **Проект**.

Проект определен как конфигурация (каркас, контейнер), объединяющая группу файлов. В рамках проекта создается программа, подлежащая исполнению. Каждый проект содержит две подконфигурации: отладочную и исполнительскую. Это задается в выпадающем меню, которое по умолчанию установлено на опцию **Debug** (отладка). Выпадающее меню находится в строке окна среды, распо ложенной ниже строки главного меню.

Проекты могут объединяться в одно решение, являются частью другого каркаса, другого контейнера, который на зывается **Решение** и который отражает взаимосвязь между проек тами. Файл ре шения по умолчанию имеет имя, такое же, как у первого проекта, но его можно изменить. В качестве расширения используется **sln**.

Код программы на C# хранится в файлах с расширением **cs**, например, **Program.cs**. Это простой текст без определенного формата.

Помимо этого, среда разработки создает две папки: **bin** и **obj** в папке проекта. В папке **obj** сохраняются временные файлы, которые используются для компиля ции, а в папке **bin** — результат компиляции. Исполняемый файл (файл с расширением **exe**, например, **ConsoleApplication1.exe**), находящийся в папке **bin/Debug**, не содержит ничего лишнего, и такие файлы поставляют заказчику или включают в установочные па кеты.

Основной файл, кото рый необходимо открывать в Visual Studio, — файл с расширением **csproj**. В каче стве имени файла будет выступать имя проекта. При открытии файла проекта **csproj** с помощью текстового редактора отображается текст похожий на XML-файл. В файле проекта с помощью XML-тегов описываются файлы, которые входят в проект. Когда с помощью команды меню **Файл->Открыть->Проект** открывается этот файл, то по служебной информации среда разработки загружает все необходимое и устанавливает соответствующие параметры.

Окно **“Окно классов”** отображает иерархию пространств имен и классов в виде дерева, которую можно развернуть для получения подробной информации о том, какие классы содержатся в пространстве имен и какие элементы содержаться в классах.

Окно **“Окно определения кода”** выводит исходный код для объекта или элемента

Окно **“Вывод”** отображает информацию о результатах компиляции

Окно **“Обозреватель серверов”** используется для получения сведений о компьютерах в сети в процессе написания кода.

1.2. Возможности редактора кода Visual C#

Visual Studio 2010 имеет несколько текстовых редакторов (редакторов кода). Все редакторы кода, независимо от языка, на котором пишется код, построены по одинаковому принципу и предоставляют основной набор функциональных возможно стей, такие как подсветка синтаксиса, поля выбора, способность сворачивать вло женные элементы.

Редактор кода обрабатывает отступы и пробельные символы для того, чтобы код был понятным и читабельным. Он обеспечивает технологию IntelliSense и дописывание операторов (для того чтобы освободить разработчика от необходимости поиска или запоминания каждой библиотеки объектных модулей или ключевого слова). Он группирует код в блоки, обеспечивает расцветку ключевых слов и комментариев, выделяет ошибки, выделяет новый код относительно ранее компилировавшегося.

Небольшие знаки "-" слева окна (рис. 3) отмеча ются пункты, где редактор кода предполагает начало нового блока программы. С их помощью код группируется в логические области. Можно использовать знак минуса для того, чтобы закрыть целый класс, метод, свойство или другую по добную группу. Эта возможность позволяет скрывать тот код, который в данный момент не используется.

Можно создавать собственные (именованные) области кода. Если необходима дополнительная группировка кода, можно указать свои собственные блоки свертывания кода с помощью директив препроцессора **#region** и **#endregion** (рис. 4).

Редактор кода автоматически обнаруживает директиву **#region** и помещает новый знак "-" около директивы, разрешая пользователю закрыть область (рис. 5). Включение этого кода в области означает, что вы можете заставить редактор кода закрывать блоки программы, отмечая область директивой **#region**. Компилятор игнорирует директивы и компилирует код программы как обычно

Новый код внутри областей помечается цветной линией (см. рис. 4 и 5). Желтый цвет используется для нового кода, который еще не сохранен. Линия становится зеленой после сохранения и исчезает, когда вы закроете и вновь откроете файл. Эта возможность позволяет отслеживать места выполненных изменений в коде. Имя открытого кодового файла показано в заголовке окна кода Звездочка указывает, что код уже изменился с момента последнего сохранения.

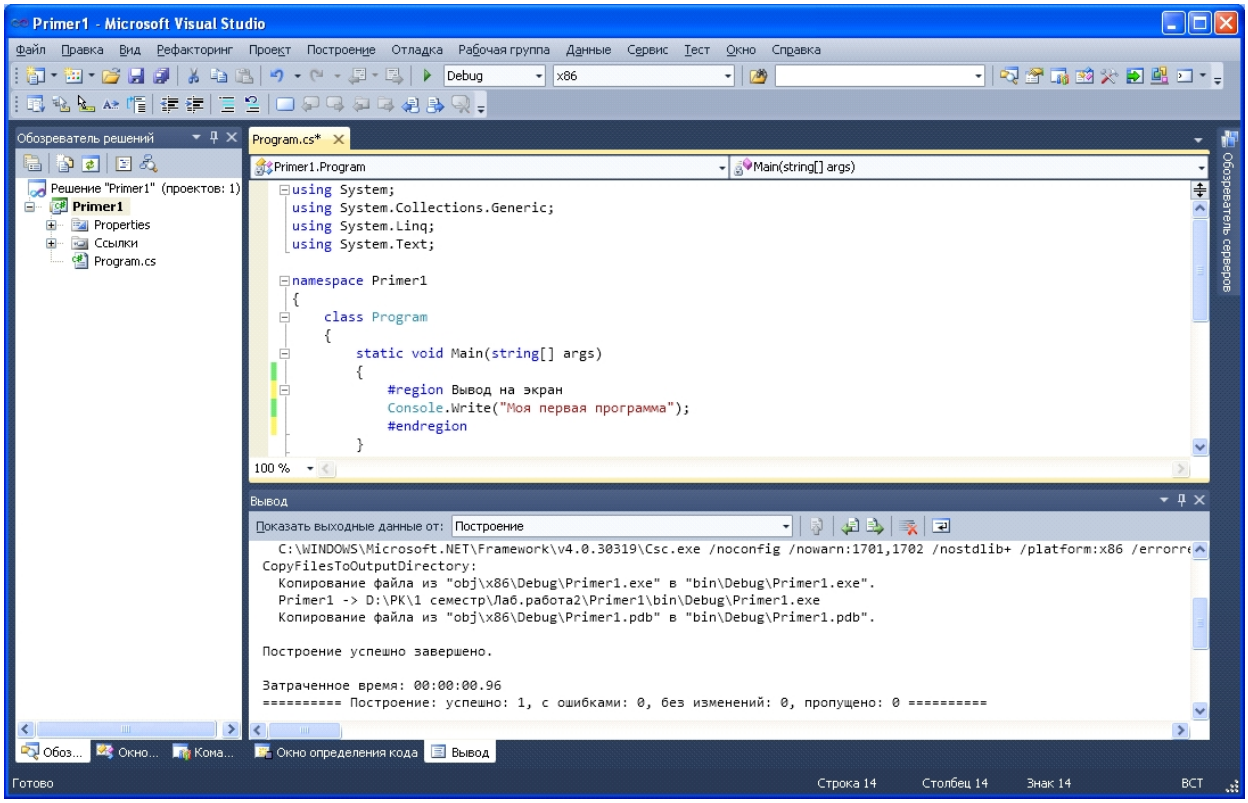

Рис. 4. Создание именованной области кода

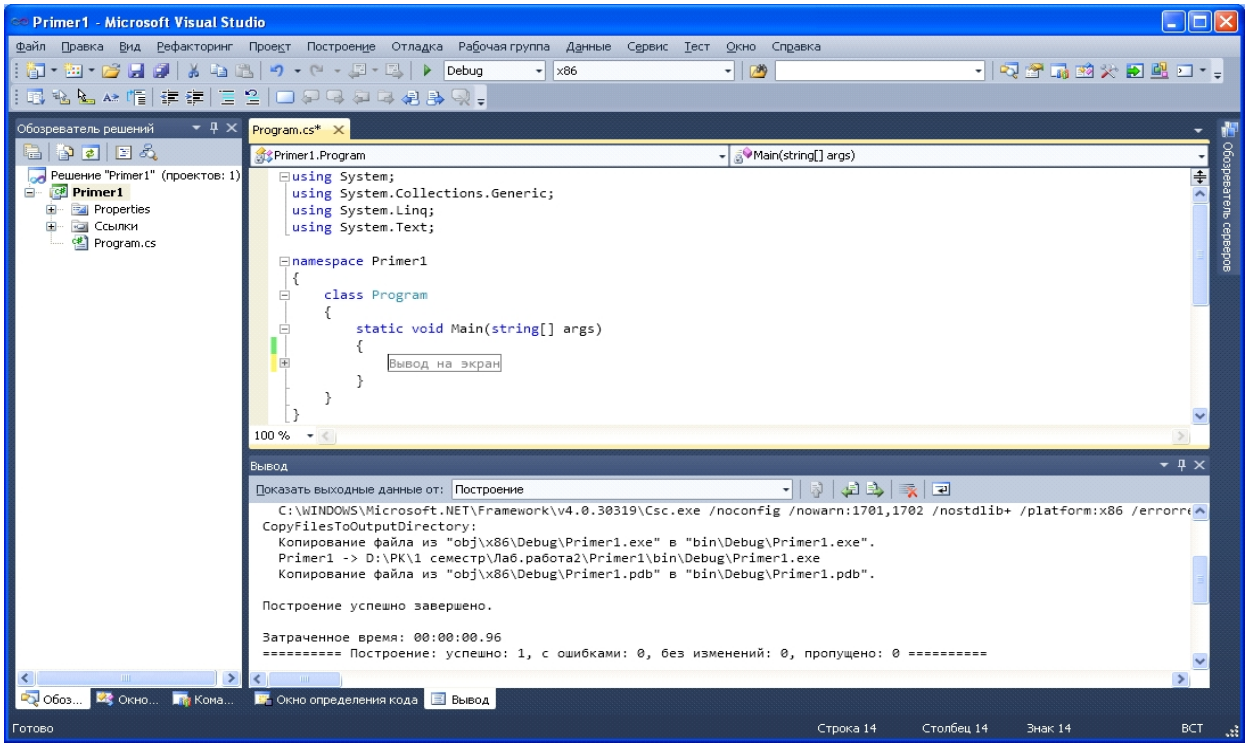

Рис. 5. Закрытый блок программного кода

При наборе кода запускается функция IntelliSense. Для быстрого нахождения в списке нужного элемента можно использовать клавиши со стрелками. При наведении указателя мыши на элемент будут показаны подробности данного элемента (текст подсказки будет справа) (рис. 6). Можно нажать клавишу <Tab> для дописывания элемента внутри IntelliSense.

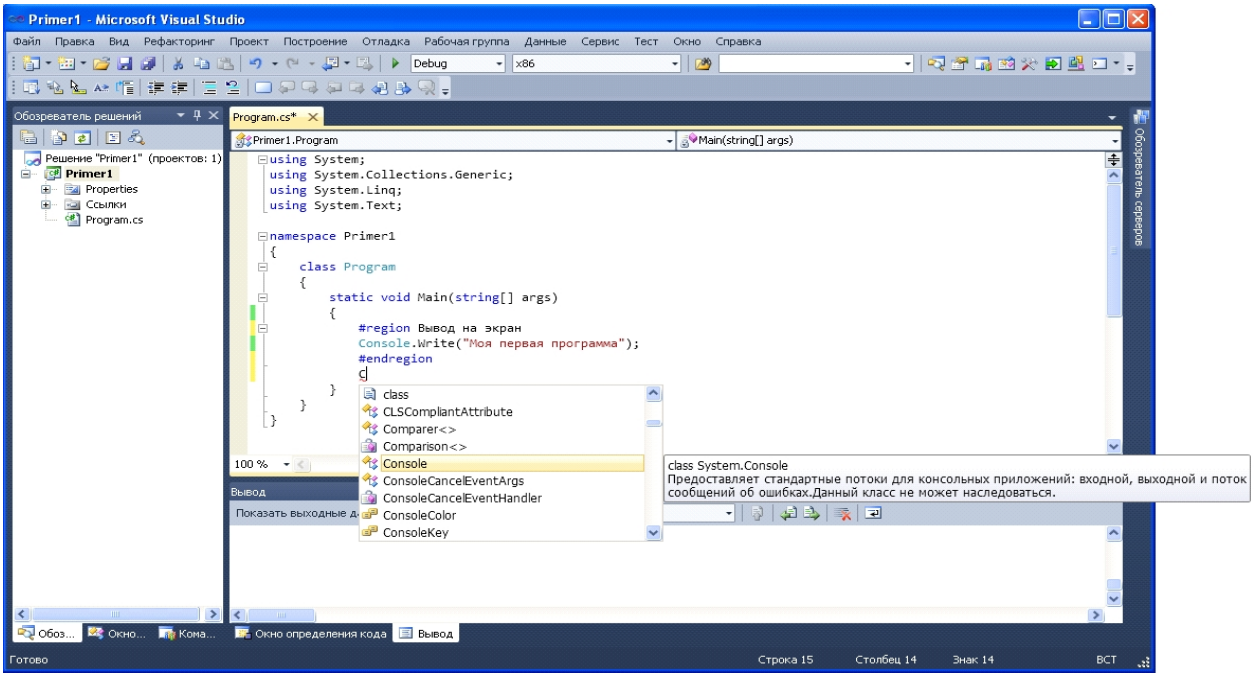

Рис. 6. Редактор кода с работающей функцией IntelliSense

Код выделяется различными цветами. По умолчанию ключевые слова имеют синий цвет, комментарии— зеленый, текст— черный, пользовательские ти пы — голубой, строковые значения — красный цвет и т. д.

Два выпадающих списка в верхней части редактора кода позволяют пере мещаться между классами в файле (левый выпадающий список) и методами, по лями и свойствами данного класса (правый выпадающий список).

Редактор кода также проводит анализ написанного кода и подчеркивает боль шинство синтаксических ошибок короткой волнистой линией. При наведении кур сора мыши на подчеркнутый текст появляется подсказка, объясняющая ошибку.

Если вас не устраивает форматирование и подсветка синтаксиса, предлагаемые редактором кода по умолчанию, можно использовать диалоговое окно **Параметры,** ко торое вызывается из главного меню командой **Сервис->Параметры,** для изменения фоно вого цвета редактора или цвета и шрифта различного текста внутри редактора. Можно также включить нумерацию строк и управлять отступами (табуляцией) и пробельными символами. Можно также настроить язык и специфические для редактора опции (рис. 7).

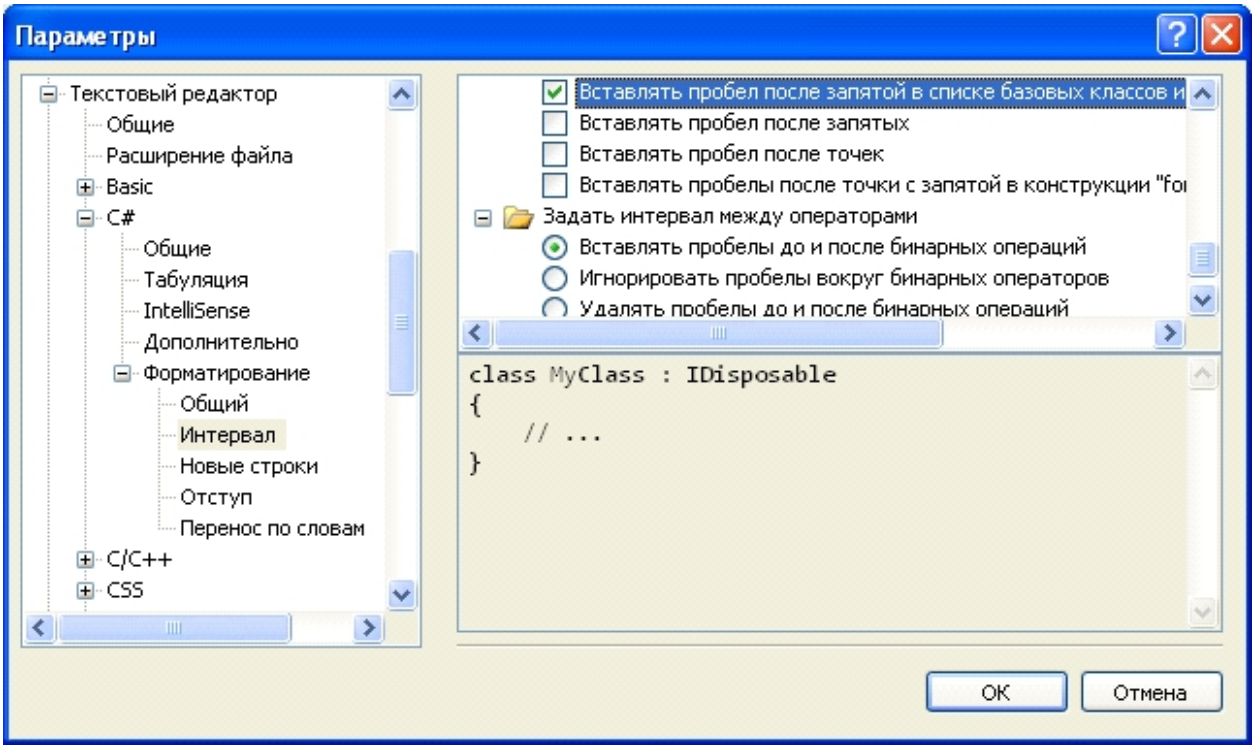

Рис. 7. Диалоговое окно “Параметры”

Текстовые редакторы также имеют несколько инструментов для отображения нумерации строк, пометки строк кода, поиска и замены текста в исходных файлах. Нумерацию строк можно включить для любого документа, загруженного в ре дактор кода. Нумерация строк включается в диалоговом окне **“Параметры”** в узле **Текстовый редактор->C#->Общие.**

1.3. Методика создания консольного приложения Visual C#

Консольное приложение — это программа, которая для взаимодействия с пользователем использует консоль — клавиатуру и монитор, работающий в режиме отображения символьной информации (буквы, цифры и специальные знаки). Консольные приложения удобны для решения задач, в которых не предъяв ляется особых требований к интерфейсу. Они широко используются для ре шения системных задач. Многие утилиты Microsoft .NET Framework реализованы как консольные приложения.

1.3.1. Создание консольного приложения Visual C#

Пример 1. В качестве примера создадим консольное приложение, которое выводит на экран результат сложения 2-х целых чисел. Исходными данными являются целыми числа, вводимые с клавиатуры, результатом операции сложения – целое число, выводимое на экран монитора. Первому числу присвоим имя **a**, второму – **b**, переменной, в которой будет храниться результат - **y**. Так как ввод и вывод осуществляется в виде совокупности символов, то необходимо предусмотреть переменную, в которой будет храниться эта совокупность (имя промежуточной переменной – **Str**).

Для создания консольного приложения воспользуемся шаблоном консольного приложения C#. Для этого необходимо выполнить следующие шаги:

1. Загрузить среду Visual Studio.

2. Выполнить команду главного меню **Файл->Создать->Проект.** Откроется диалоговое окно «Создать проект», показанное на рис. 2. В открывшемся окне **Создать проект** раскрыть список **Visual C#** и выбрать тип приложения — **Windows**.

3. В этом окне выбрать опцию **Консольное приложение Visual C#,** задать в его нижней части имя будущего проекта в поле **Имя,** которое продублируется в поле **Имя решения,** затем с помощью кнопки **Обзор …** установить папку, в которую будет помещен проект.

4. Затем нажать кнопку **ОК**. В результате получится то, что показано на рис. 9.

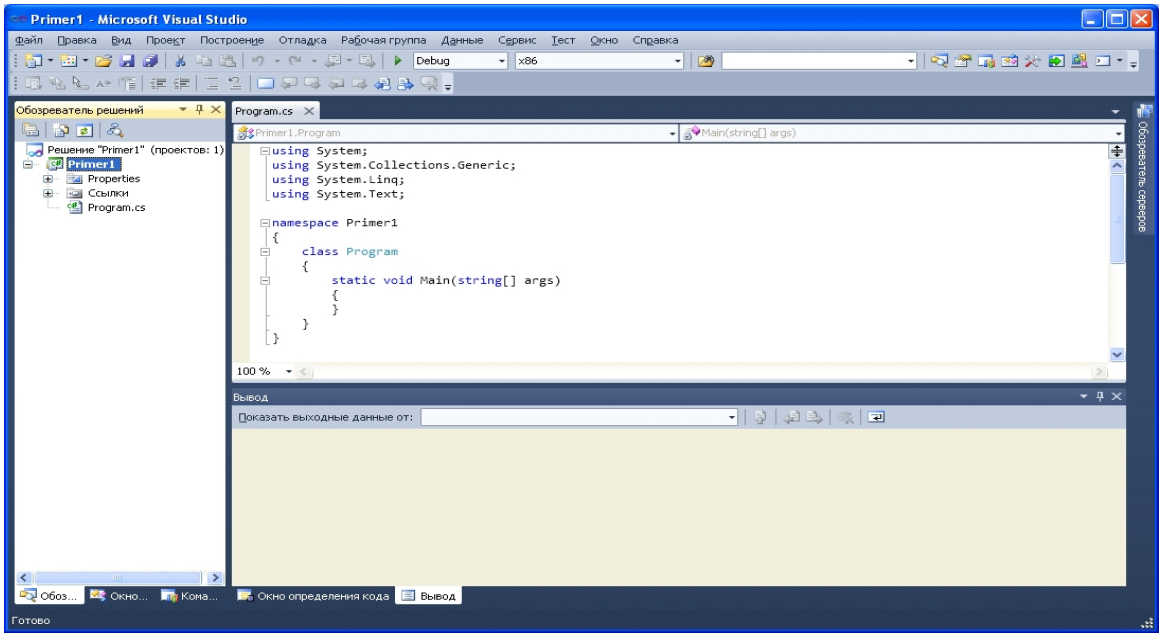

Рис. 9. Вид заготовки консольного приложения

5. С помощью редактора кода модифицируем тело функции**Main()** добавив последовательность операторов:

```
static void Main(string[] args)
        {
         int a; // имя переменной, в которой хранится первое число
         int b; // имя переменной, в которой хранится второе число
         int y; // имя переменной, в которой хранится результат
         String Str; // имя переменной, в которой хранится совокупность символов
         Console.Write("Введите первое число");
         Str = Console.ReadLine();
         a = Convert.ToInt16(Str);
         Console.Write("Введите второе число");
         Str = Console.ReadLine();
         b = Convert.ToInt16(Str);
         y = a + b;
         Str = y.ToString("d");
         Console.Write("Результат равен ");
         Console.Write(Str);
         Console.Write("Для завершения работы приложения нажмите <Enter>");
         Console.Read();
        }
```

В 12 – 15 строках объявлены переменные, используемые в программе (переменные **a, b, y** – целые числа, переменная **Str** – строковая переменная). В строке 16 записан оператор, с помощью которого выдается текстовая строка "Введите первое число". После выдачи текстового сообщения программа приостанавливает работу, ожидая ввода значения первого числа. Ввод значения числа и сохранения его значения осуществляется с помощью оператора, записанного в строке 17. Так как арифметические операции производятся над числами, то необходимо преобразовать строковую переменную **Str** в число и сохранить в переменной **a** (оператор 18). Действия операторов, записанных в строках 19 – 21, аналогичны операторам в строках 16 – 18, обеспечивают ввод второго числа. Оператор в 22-й строке обеспечивает сложение 2-х чисел **a** и **b** и сохранение результата в переменной **y**. Так как результат выдается на экран монитора в виде совокупности символов, то необходимо преобразовать число в строку(это выполняет оператор, записанный в строке 23). В

строках 24 и 25 записаны операторы обеспечивающие вывод результата операции сложения2-х чисел. Оператор **Console.Read();** приостанавливает выполнения программы с целью увидеть результаты работы приложения. Нажатие клавиши <**Enter**> завершает выполнения программы

В итоге консольное приложение будет иметь вид представленный на рис. 10.

6. Итак, приложение готово. Можно откомпилировать и выполнить его. Процесс преобразования исходной программы в выполняемую называется компиляцией. Этот процесс можно представить как последовательность двух этапов - компиляция и компоновка. На этапе компиляции выполняется перевод исходной про граммы (модулей) в некоторое внутреннее представление. На этапе компо новки выполняется объединение модулей в единую программу.

Процесс построения программы активизируется в результате выбора в меню **Построение** команды **Построить <*имя проекта*>**. Процесс и результат компиляции отражаются в окне "**Вывод".** Если в про грамме нет ошибок, то по завершении процесса компиляции в окне "**Вывод"** отображается сообщение о результате построения (см. рис. 11).

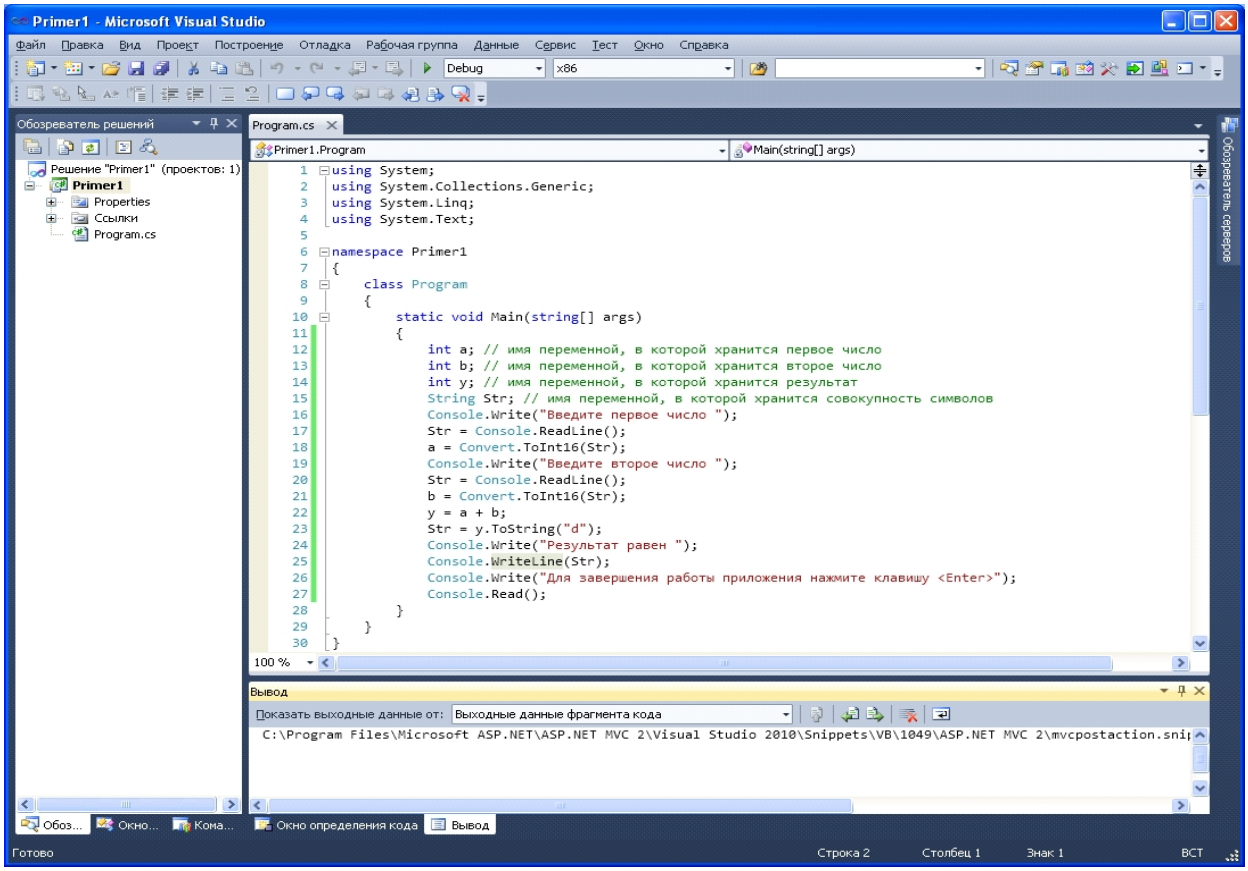

Рис. 10. Вид консольного приложения до компиляции

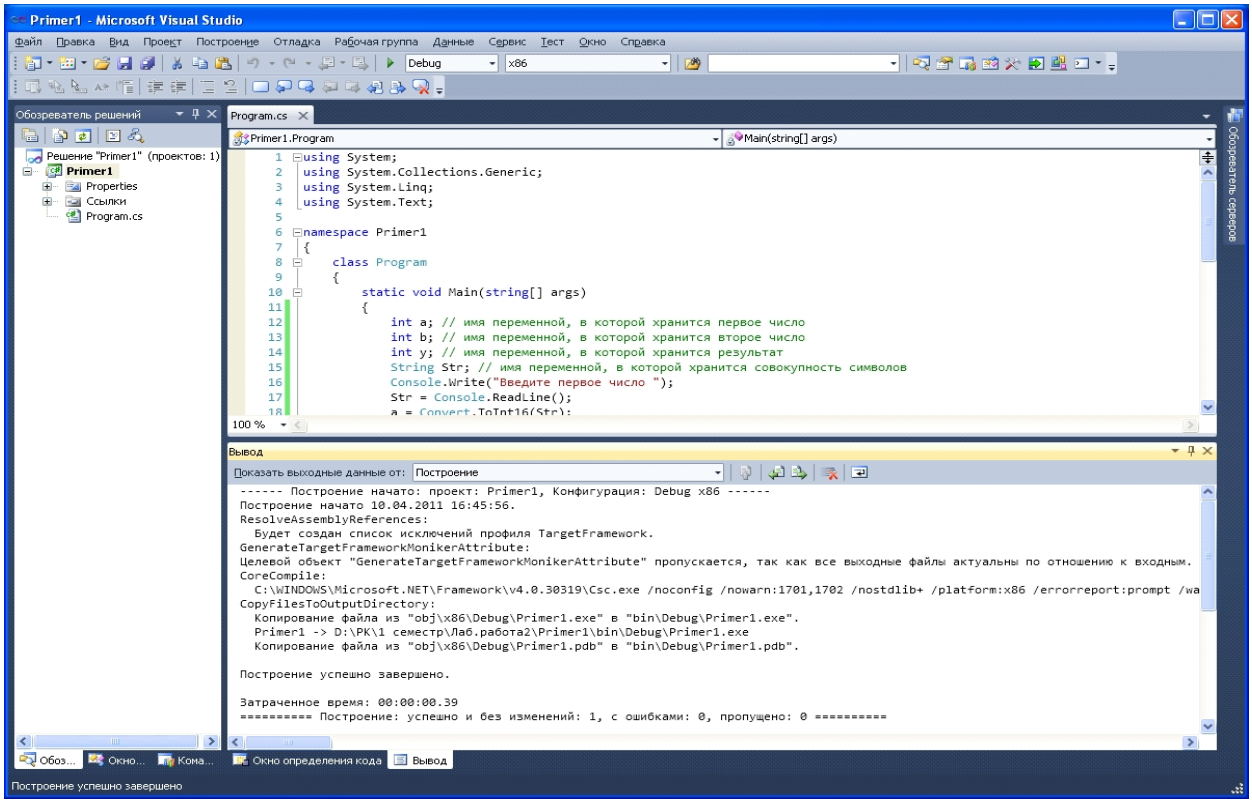

Рис. 11. Вид ИСР после завершения процесса компиляции

Если в процессе компиляции выявлены ошибки, в окне **“Вывод”** выдаются соответствующие сообщения.

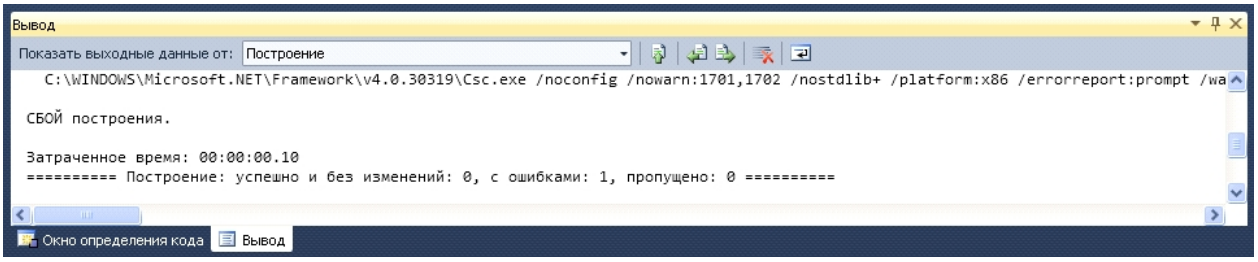

Рис. 12. Окно “Вывод” сигнализирующее об ошибках

Например, при отсутствии объявления переменной **y** выдаются сообщения, представленные на рис. 12. Для того чтобы посмотреть ошибочный код в тексте приложения достаточно дважды щелкнуть на строке об ошибке в окне **“Вывод”** и в редакторе кода будет выделена маркером строка, содержащая ошибку, и будет подчеркнуто место в строке, где возможна ошибка (для нашего примера указан оператор, где используется необъявленная переменная).

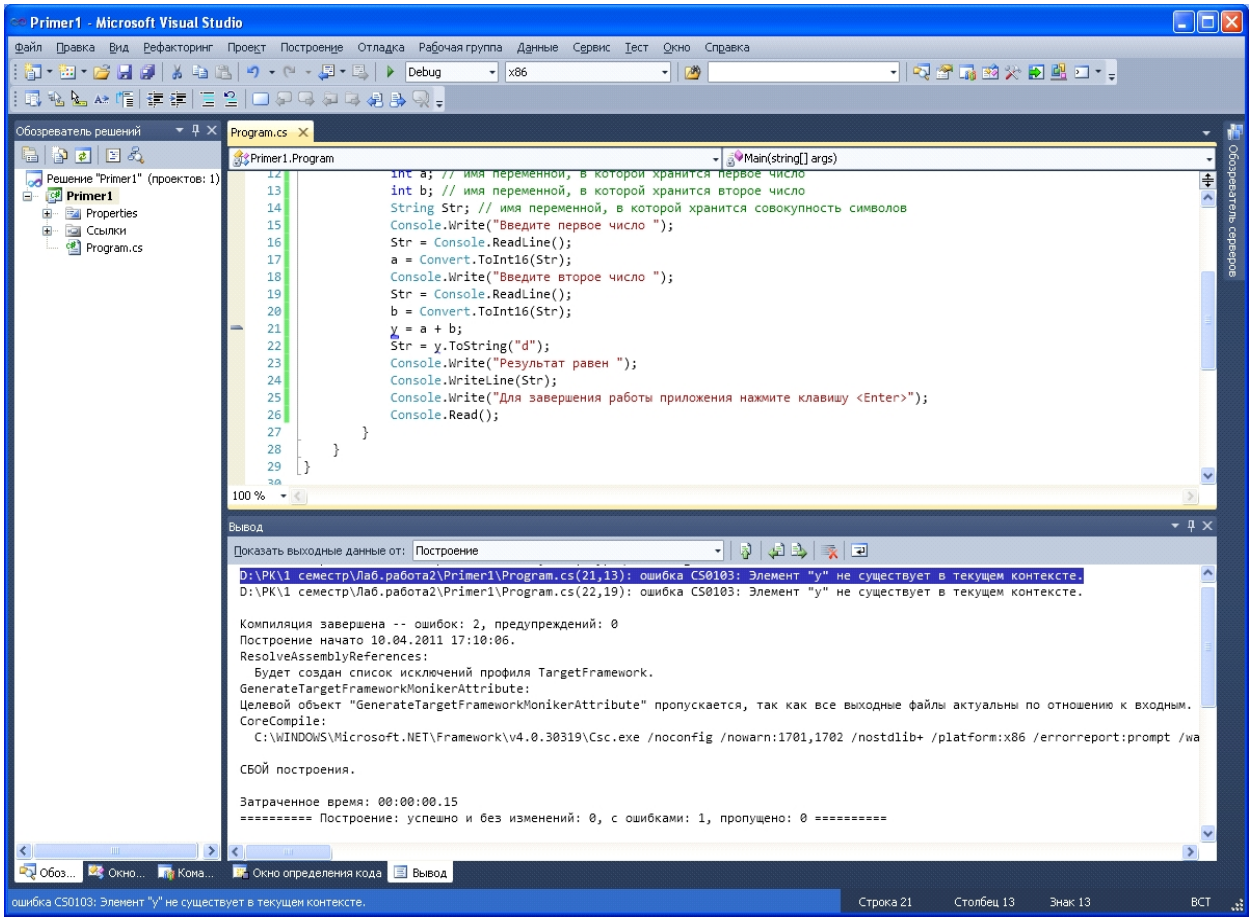

Рис. 13. Вид ИСР после завершения процесса компиляции с ошибками

7. Запуск программы из среды разработки осуществляется командой «**Начать отладку**» или «**Запуск без отладки**» меню «**Отладка**». Результатом выполнения приложения будет выдача в окне черного цвета соответствующего текста (см. рис. 14).

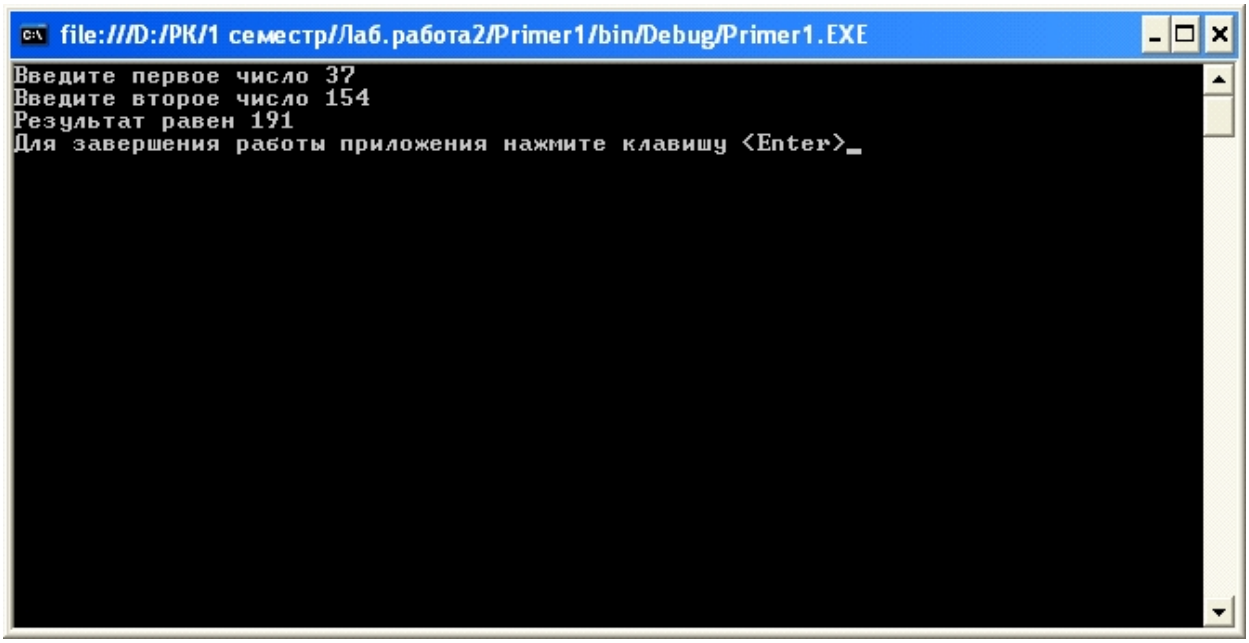

Рис. 14. Результат выполнения консольного приложения

1.3.2. Создание улучшенного консольного приложения Visual C#

Консоль в Windows — это класс **Console** определенного типа окна, и у него есть несколько свойств, которые позволяют управлять этим окном и параметрами текста в нем:

● Свойство **Title** — текстовая строка, которая отображает заголовок окна.

Оператор

Console.Title =”Моя первая программа”;

изменяет заголовок окна консоли;

● Свойство **ForegroundColor** — цвет текста. В консоли цвет описан как **ConsoleColor** (объявлено в системе как **public enum ConsoleColor**). Чтобы узнать доступные цвета, можно в редакторе кода набрать **ConsoleColor** и нажать точку (без пробелов). В ответ должен по явиться выпадающий список с доступными значениями (рис. 15). Если выпадаю щий список не появился, поставьте курсор сразу за точкой и нажмите комбинацию клавиш **<Ctrl>+<пробел>**.

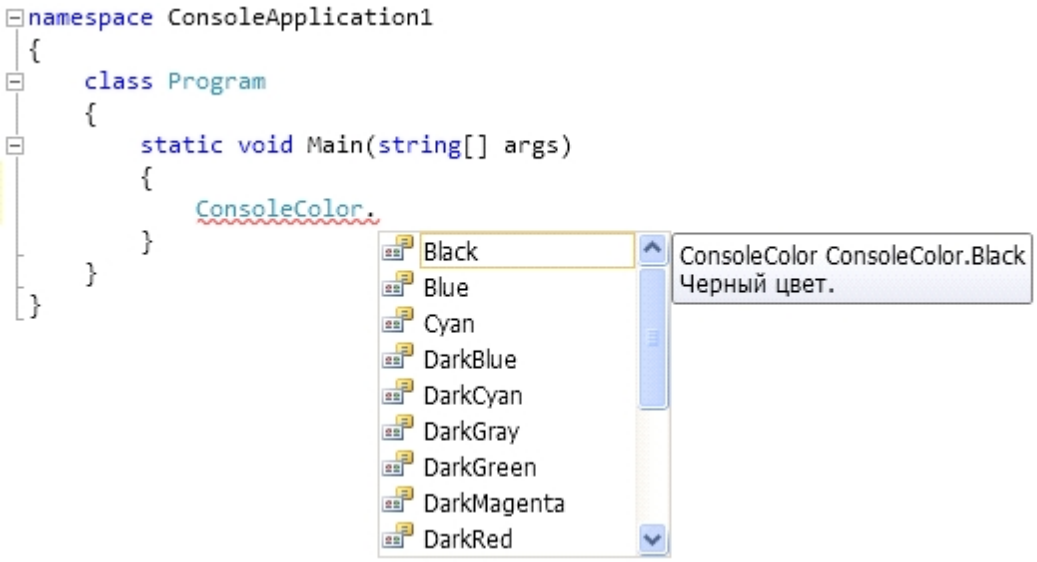

Рис. 15. Выпадающий список со значениями ConsoleColor в редакторе кода

Оператор

Console.ForegroundColor = ConsoleColor.Green;

изменить цвет текста в консоли на зеленый.

● Свойство **BackgroundColor** определяет цвет фона текст и имеет тип **ConsoleColor**. Свойство изменяет цвет фона только текста, а не всего окна консоли.

Оператор

Console.BackgroundColor = ConsoleColor.Yellow;

изменить цвет фона текста в консоли на желтый.

● Если необходимо изменить цвет всего окна, то после изменения цвета фона нужно очистить окно консоли, для чего используется метод **Clear()**.

Операторы

Console.BackgroundColor = ConsoleColor.White;

Console.Clear();

изменят цвет фона текста в консоли на белый и очистят консоль

● Свойство **CapsLock** имеет тип **bool** и возвращает **true**, если нажата клавиша **<CapsLock>**. Это свойство только для чтения.

● Свойство **NumberLock** позволяет определить, нажата ли клавиша **<NumLock>** в данный момент. Если свойство вернет **true**, то клавиша нажата.

● Свойства **WindowHeight** и **WindowWidth** позволяют задать высоту и ширину окна соответственно. Значения задаются в символах и зависят от разрешения экрана.

● Свойства **WindowLeft** и **WindowTop** позволяют задать левую и верхнюю позиции окна относительно экранного буфера.

Консоль — это не просто текстовое окно, в которое можно только последовательно выводить информацию. Консоль — это целый буфер с памятью, куда данные могут выводиться даже хаотично. Когда окно консоли запущено, то справа появляется полоса прокрутки. Это потому что буфер строк по умолчанию очень большой и рассчитан на 300 строк. Буфер колонок (символов в ширину) всего 80 символов, поэтому горизонтальной прокрутки нет. Но если уменьшить окно, то появится и горизонтальная полоса прокрутки.

● Чтобы просмотреть размеры буфера можно воспользоваться свойствами **BufferHeight** и **BufferWidth** (высота и ширина буфера). Оба значения возвращают количество символов.

● С помощью свойств **CursorLeft** и **CursorTop** можно узнать или изменить позицию курсора относительно левой и верхней границ буфера соответственно.

Пример 2. Нижеприведенный код выводит приблизительно в центре окна (если оно имеет размеры по умолчанию) названия допустимых к использованию в консоли цветов:

```
ConsoleColor[] colors = { ConsoleColor.Blue, ConsoleColor.Red,
                          ConsoleColor.Cyan, ConsoleColor.White,
                          ConsoleColor.Yellow, ConsoleColor.Green };
 foreach (ConsoleColor color in colors)
        {
           Console.CursorLeft = (Console.BufferWidth - color.ToString().Length) / 2;
           Console.CursorTop = 10;
           Console.ForegroundColor = color;
           Console.WriteLine(color);
           System.Threading.Thread.Sleep(1000);
           Console.Clear();
        }
```

Перед циклом созда ется массив из нескольких значений цветов, доступных для консоли. После этого запускается цикл **foreach**, который просматривает весь этот массив.

Внутри цикла в первой строке устанавливается левая позиция так, чтобы сообщение получилось посередине окна. Для этого из ширины буфера вычитается ширина строки с именем цвета и делится пополам. Про цесс определения размера имени цвета заключается в следующем. Так как цвет у нас тип **ConsoleColor**, то вызвав метод **ToString()** получаем имя цвета в виде строки, а у строки имеется свойство **Length**, в котором содержится размер строки.

В качестве смещения сверху выбираем 10 символов. Это примерно середина окна, если его разме ры не трогали.

Теперь можно изменить цвет текста на текущий, чтобы визуально увидеть его и вывести его название. Обратите внимание, что консольному методу **WriteLine()** передается переменная **color**, которая имеет тип **ConsoleColor**, и метод **ToString()** не вызывается явно. Дело в том, что если нужно привести тип к строке, то метод **ToString()** вызывается автоматически.

Оператор **System.Threading.Thread.Sleep(1000)**; делает задержку на количество миллисекунд, указанных в круглых скобках, т.е. делает задержку в секунду.

Последняя строка вызывает метод **Clear()**, чтобы очистить консоль.

2. Рабочее задание

Задание 1. Руководствуясь теоретическим материалом раздела 1.1 изучить возможности интегрированной среды разработки языка C# и выполнить практически все действия, описанные в этом подразделе.

Задание 2. Руководствуясь теоретическим материалом подраздела 1.2 изучить возможности редактора кода языка C# и выполнить практически все действия, описанные в этом подразделе.

Задание 3. Руководствуясь теоретическим материалом подраздела 1.3.1 изучить методику создания консольного приложения и выполнить практически пример, описанный в этом подразделе. Проект и консольное приложение сохранить под именем **Zadanie3**.

Задание 4. Руководствуясь теоретическим материалом подраздела 1.3.2 изучить дополнительные возможности создания улучшенного консольного приложения и выполнить практически пример, описанный в этом подразделе. Проект и консольное приложение сохранить под именем **Zadanie4**.

Задание 5. Модифицировать консольное приложение задания №3 таким образом, чтобы заголовок приложения содержал строку «Модифицированное задание №3», цвет фона был белый, цвет текста – зеленый. Проект и консольное приложение сохранить под именем **Zadanie5**.

3. Контрольные вопросы

1. Что такое интегрированная среда разработки Visual Studio 2010?
2. Перечислите состав и назначение элементов главного окна ИСР.
3. Перечислите состав и назначение пунктов главного меню.
4. Что такое проект?
5. Перечислите последовательность действий по созданию консольного приложения.
6. Назначение редактора кода и его возможности.
7. Перечислите элементы, позволяющие управлять окном приложения и его содержимым.

Литература

1. Голощапов А.Л. Microsoft Visual Studio 2010. – СПб.: БХВ-Петербург, 2011. – 544 с.: ил.
2. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 1. Пер. с англ. - М.: «Русская Редакция», 2002.- 576 с.: ил.
3. Петцольд Ч. Программирование для Microsoft Windows на C#. В 2-х томах. Том 2. Пер. с англ. - М.: «Русская Редакция», 2002.- 624 с.: ил.
4. Троелсен Э. Язык программирования C# 2010 и платформа .NET 4.0. Пер. с англ. - М.: Издательский дом "Вильямс", 2011. — 1392 с.: ил.
5. Фленов М.Е. Библия C#. - СПб.: БХВ-Петербург, 2011.– 560с.: ил.
6. Шилдт Г. C# Учебный курс. – СПб.: Питер, Издательская группа BHV, 2003. – 512 с.: ил.