**Практическое задание 1**

**Работа в IDE, данные, ввод и вывод, отладка**

План

[Тема 1. Работа в интегрированной среде разработчика Visual Studio 2](#_Toc19479952)

[1.1 Роль, назначение, идеология 2](#_Toc19479953)

[1.2. Запуск Visual C++ 2](#_Toc19479954)

[1.3. Проекты и решения C++ 3](#_Toc19479955)

[1.4. Создание консольного приложения 3](#_Toc19479956)

[1.5. Сохранение проекта 5](#_Toc19479957)

[1.6. Открытие проекта 5](#_Toc19479958)

[1.7. Добавление файлов в проект 5](#_Toc19479959)

[1.8. Исключение файлов из состава проекта 6](#_Toc19479960)

[1.9. Окно редактора кода 7](#_Toc19479961)

[1.10. Запуск программы на исполнение 8](#_Toc19479962)

[1.11. Сообщения об ошибках 9](#_Toc19479963)

[1.12. Схема выполнения задачи в IDE разработчика 9](#_Toc19479964)

[Тема 2. Данные в языке С++. Основные типы данных. Константы и переменные. Ввод и вывод данных 11](#_Toc19479965)

[2.1. Данные: определение и концепция типов 11](#_Toc19479966)

[2.2. Основные базовые типы 12](#_Toc19479967)

[2.3. Средства языка для ввода и вывода данных 12](#_Toc19479968)

[2.4. Упражнения 13](#_Toc19479969)

[Тема 3. Средства отладки программ 14](#_Toc19479970)

[Тема 4. Составляющие IDE Visual C++ 16](#_Toc19479971)

[4.1. Рабочее пространство 16](#_Toc19479972)

[4.2. Краткий обзор команд меню и панелей инструментов 17](#_Toc19479973)

[4.2.1. Пункт меню Файл 17](#_Toc19479974)

[4.2.2. Пункт меню Правка 18](#_Toc19479975)

[4.2.3. Пункт меню Вид 19](#_Toc19479976)

[4.2.4. Пункт меню Проект 19](#_Toc19479977)

[4.2.5. Пункт меню Построение 20](#_Toc19479978)

[4.2.6. Пункт меню Отладка 20](#_Toc19479979)

[4.2.7. Пункт меню Сервис 20](#_Toc19479980)

[4.2.8. Пункт меню Окно 21](#_Toc19479981)

[4.2.9. Пункт меню Справка 21](#_Toc19479982)

[4.2.10. Роль горячих клавиш и контекстного меню 21](#_Toc19479983)

Тема 1. Работа в интегрированной среде разработчика Visual Studio

1.1 Роль, назначение, идеология

Microsoft Visual Studio, это интегрированная среда разработчика (**IDE** – Integrated Development Environment), предназначенная для разработки программных комплексов на языках C++, Basic, C#, F# и других.

Основное назначение IDE – решительно ускорить цикл разработки сложных программных проектов для различных предметных областей.

Среда поддерживает проекты разного типа – от консольных до типовых шаблонных решений, в том числе интернет-приложений.

Для изучения основ программирования на языке С++ мы будем использовать подсистему Visual C++ , встроенную в IDE, но имеющую самостоятельное значение.

1.2. Запуск Visual C++

Запуск осуществляется выбором команды *Visual Studio* в списке приложений по легко узнаваемой пиктограмме .

**Примечание**

Запуска по ассоциации с именем файла (расширение \*.c или \*.cpp или \*.h) следует избегать, так как для работы приложения необходим **проект.** Проект, это совокупность файлов, обеспечивающих работу приложения в IDE.

Откройте Visual Studioи ознакомьтесь с начальной страницей.

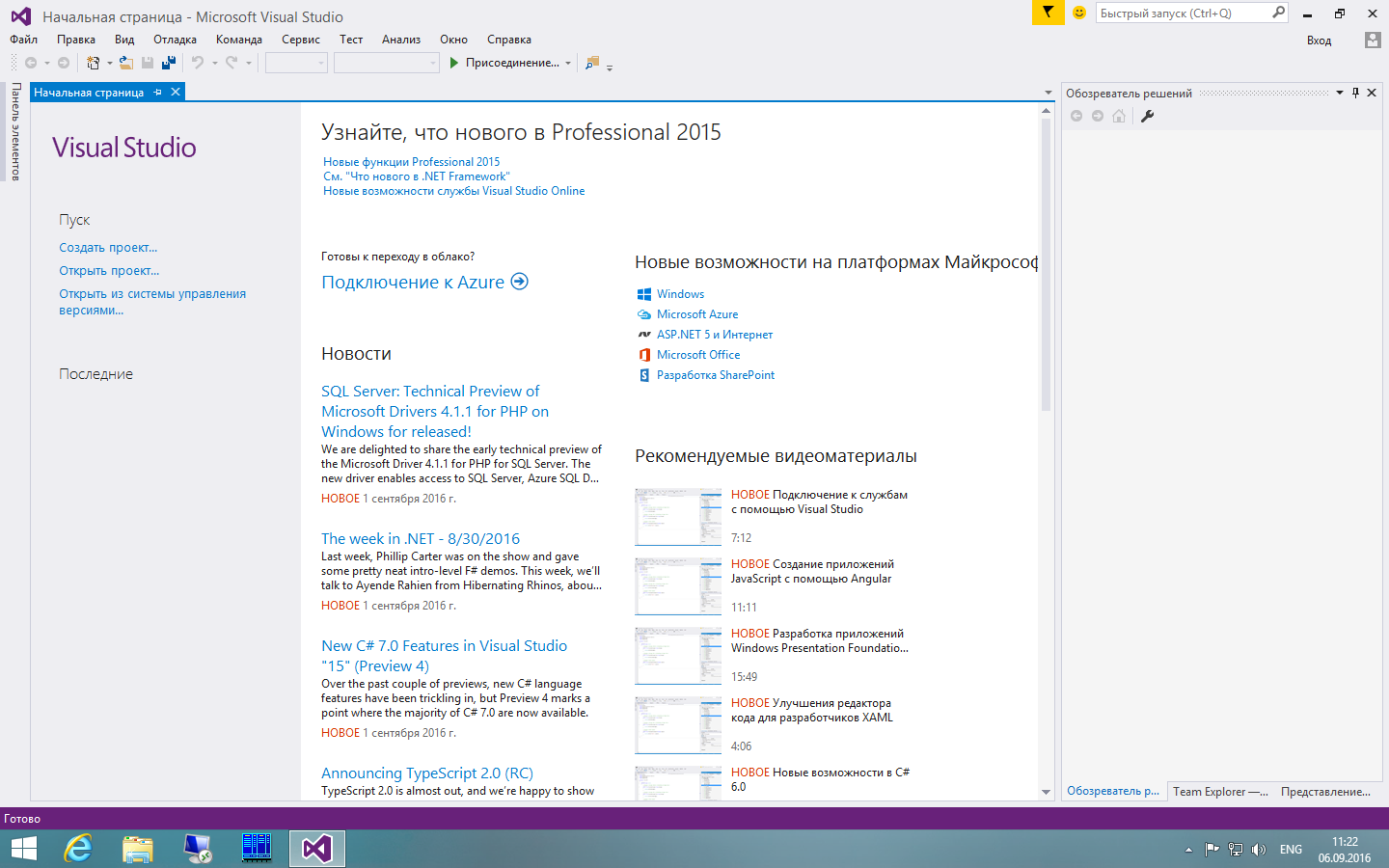


Рис. 1. Начальная страница Visual Studio 2015

На стартовой странице «Начальная страница» в панели слева есть возможность открыть один из проектов, что ранее был в работе (*«Последние проекты»* или *«Открыть проект»*), или создать новый проект (*«Создать проект»*), а также получить интерактивную справку о среде разработки, языке программирования и о многом другом. Так, пункт «*Узнайте, что нового*» дает возможность получения развернутой справки о возможностях среды и о работе в ней.

1.3. Проекты и решения C++

В Visual С++ для организации работы служат проекты и решения. Решение может включать в себя несколько проектов, например библиотеку DLL и ссылающийся на нее исполняемый файл. Первым этапом работы является выбор типа проекта, потому что для каждого типа проекта Visual С++ устанавливает параметры компилятора и генерирует стартовый код.

1.4. Создание консольного приложения

Консольные приложения, это приложения без графического интерфейса, которые запускаются пользователем посредством командной строки или, в рамках IDE, командой запуска. Проект создается на основе шаблона консольного приложения, включает в себя необходимые составляющие, в которые разработчик приложения добавляет файлы исходного кода.

Проект создается выбором пункта меню *Файл→Создать→Проект.* В окне диалога «Создание проекта» нужно выбрать **язык** программирования – Visual C++, тогда в области «*Установленные шаблоны*» появятся доступные **типы** проектов.

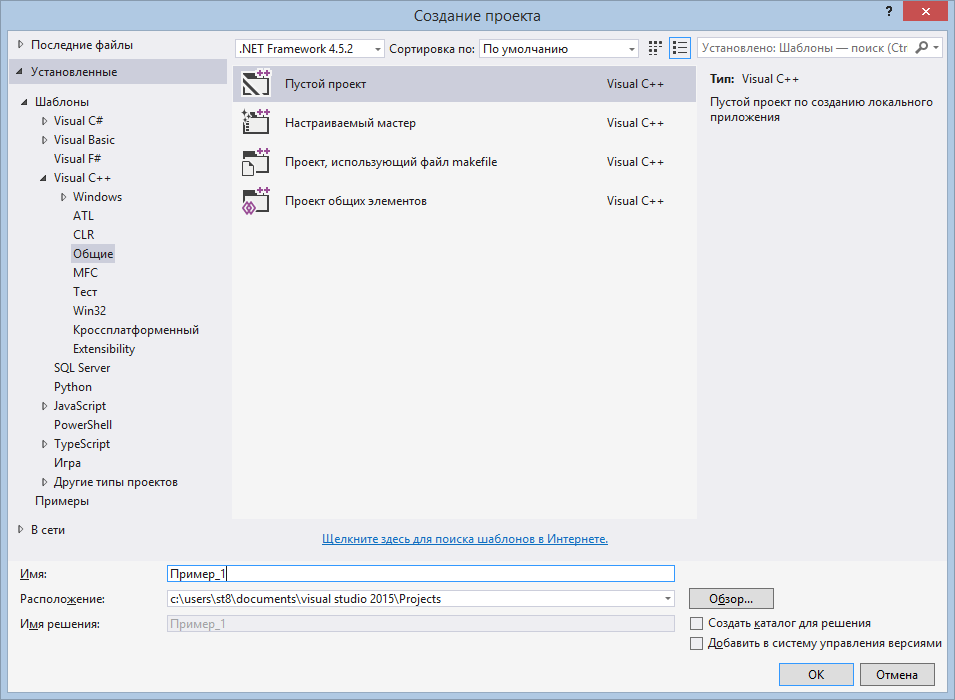


Рис. 2. Диалог создания проекта

Здесь необходимо задать **имя** проекта и выбрать **место** его сохранения на диске, соответственно, в пунктах «Имя» и «Расположение». По умолчанию проект сохраняется в «Документы». Можно принять место размещения по умолчанию, ввести иное место размещения или перейти к каталогу, в который требуется сохранить проект. Для каждого проекта будет создана отдельная папка, а внутри нее решение и проект с именем, указанным в качестве имени проекта.

Флажок *Создать каталог для решения* нужно снять, так как для решений из одного проекта нет смысла создавать отдельный каталог.

Создайте пустой проект выбором шаблона «Пустой проект».

Состав и структура проекта отображаются в окне обозревателя решений, припаркованном к краю рабочего стола. Обозреватель решений упрощает работу с файлами и другими ресурсами, включенными в решение. Если обозревателя решений нет, его отыскивают в меню *Вид→Обозреватель решений.*

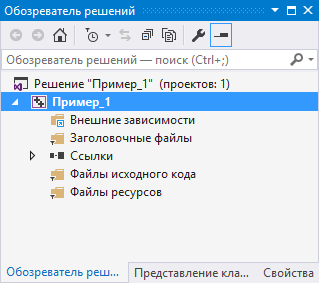


Рис. 3. Окно обозревателя решений для пустого проекта

В обозревателе решений состав проекта отображается как пустой – в нем пока нет файлов кода.

В состав проекта входят файл решения, файл проекта, файл ресурсов и другие. Если открыть директорию, в которой проект сохранен, то мы увидим, что проект содержит файлы, необходимые для создания законченного приложения.

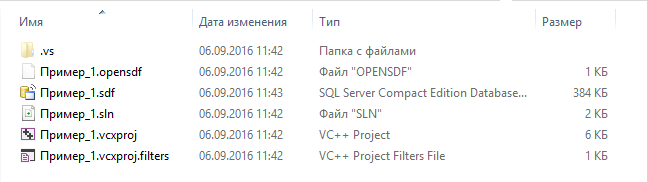


Рис. 4. Состав файлов пустого проекта

Пример\_1.sln – это файл решения, содержащий описание структуры приложения в целом. Одно решение может содержать некоторое количество проектов. Среда разработчика автоматически формирует создаваемое приложение как решение, пусть даже содержащее только один проект.

Пример\_1.vcproj – это файл проекта, содержащий элементы, обеспечивающие существование приложения.

В группе «Файлы исходного кода» могут находиться несколько модулей, содержащих код программы и описания используемых ею данных. Все файлы исходного кода в составе проекта будут компилироваться совместно. В группе «Заголовочные (header) файлы» могут находиться файлы, содержащие описания данных и функций, используемых в модуле.

Поскольку создан пустой проект, то в его составе нет ни заголовочных, ни файлов исходного кода. Далее программист сам добавляет файлы в состав проекта.

1.5. Сохранение проекта

При сохранении проекта выполняется сохранение всех его компонент. Все имена и местоположение были определены при создании проекта, поэтому пункт *Файл→Сохранить как* неактуален. Для сохранения проекта удобнее всего использовать стандартную пиктограмму сохранения , которая сохраняет все изменения в составе проекта, или команду меню *Файл→ Сохранить все.*

1.6. Открытие проекта

Чтобы открыть проект, лучше всего запустить среду программированияVisual C++, и выполнить команду меню *Файл →Открыть решение или проект*, выбирая файл решения (расширение \*.sln).

Другим хорошим вариантом является окно «Последние проекты», в котором находится список проектов, с которыми в среде работали недавно. Щелчок на имени проекта также откроет решение.

Категорически не рекомендуется открывать проект по ассоциации или через открытие одного из его файлов.

1.7. Добавление файлов в проект

Добавление файлов к проекту необходимо во многих случаях, особенно когда создан пустой проект, Поскольку проект есть объект сложный, необходимо пользоваться командами меню *Проект* или использовать контекстное меню решения или отдельных его групп файлов. Контекстные меню обозревателя решений позволяют выбрать различные действия по организации решения в целом. Обратите внимание на состав контекстного меню всего решения, контекстного меню проекта и контекстных меню его отдельных составляющих.

Так, чтобы добавить файл исходного кода, контекстное меню нужно вызвать на пиктограмме «Файлы исходных кодов».

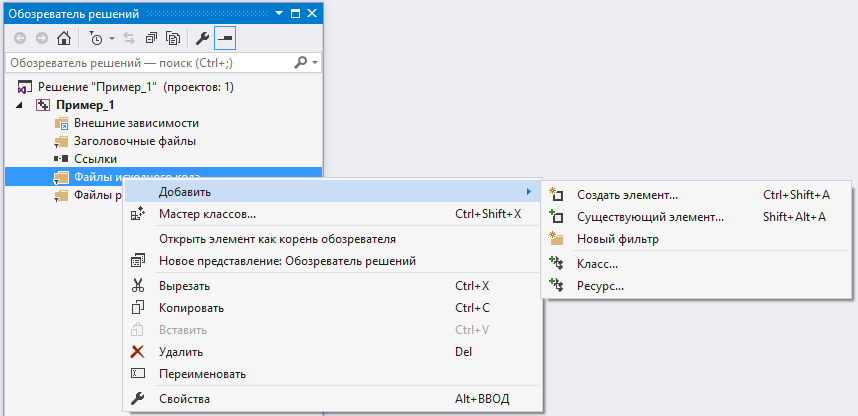


Рис. 5. Добавление к проекту файла исходного кода

Как видим, к проекту можно добавить новый элемент или уже существующий, который должен находиться в папке проекта.

При выборе команды *Создать элемент* в окне диалога нужно выбрать тип файла и задать его имя.

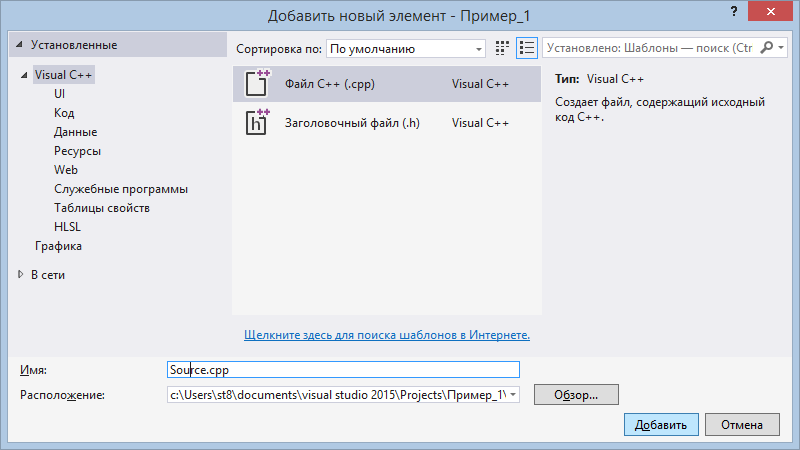


Рис. 6. Добавление к проекту файла исходного кода

Добавьте в проект файл исходного кода.

1.8. Исключение файлов из состава проекта

Отдельные файлы можно включать в состав проекта, а также и исключать, и даже удалять. Контекстное меню, вызванное на имени любого файла проекта, позволяет решать эти задачи.

Исключение из проекта означает, что файл не прекратит своего физического существования, а удаление означает то, что означает.

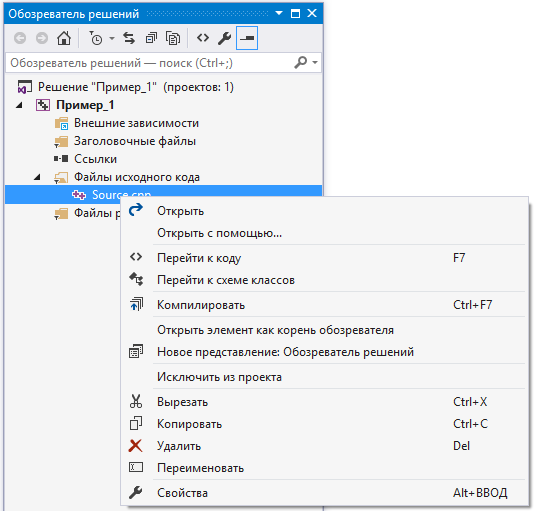


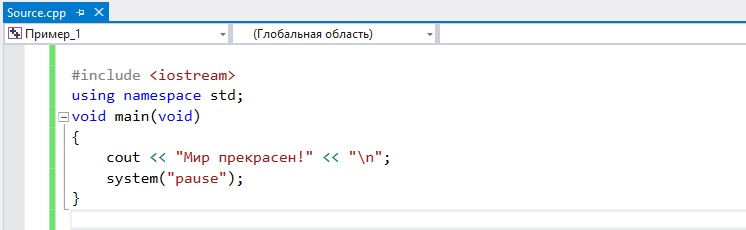
Рис. 7. Контекстное меню файла исходного кода

1.9. Окно редактора кода

Окно редактора кода расположено в центре рабочего пространства. Имеет вид многостраничного блокнота с вкладками, в каждой из которых можно редактировать текст файла. Название редактируемого файла написано на заголовке вкладки.

При создании простого консольного приложения редактируется единственный файл модуля. В окне редактирования создается заготовка программы, содержащая описание единственной его функции с именем **main.** Цель программирования – наполнить эту заготовку необходимым содержанием.

Введите следующий код.



Здесь пишем код программы

Рис. 8. Окно редактора кода

Механизм ввода текста и символьного редактирования – обычный, как в любом текстовом редакторе. Строка состояния отображает местоположение курсора и состояние режима ввода (вставка или замена). Доступны все операции редактирования текста. Имеется возможность выполнить откатку (стандартный Ctrl/Z).

Редактор кода имеет множество достоинств. Обратите внимание на цветовое выделение элементов кода, автоматическое добавление отступов, формирующих структуру программы, выделение структурных единиц (блоков) и так далее.

**Пояснения к коду**

Программа состоит из одной функции, ее имя **main**.

Тело функции заключено в фигурные скобки.

Каждый оператор заканчивается знаком «;».

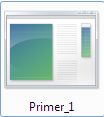
Для ввода-вывода данных подключается библиотека **<iostream>**.

Директива **using namespase std;** позволяет получать доступ к библиотеке шаблонов, содержащей много полезного.

1.10. Запуск программы на исполнение

Наиболее простой способ запуска отладки, а если построение прошло успешно, то и последующего запуска приложения, это кнопка Run . При нажатии оной выполняется последовательно компиляция, сборка и, если нет ошибок, то и выполнение программы.

Результатом компиляции (Compile) является объектный код в файле с расширением .obj. Смыслом сборки (Build) является подключение необходимых библиотек, в результате чего будет получен исполнимый код в файле с расширением .exe. Этот файл может быть выполнен независимо от среды разработчика.

При запуске отладчика в папке проекта создается внутренняя директория с именем Debug, в которой находится вся отладочная информация и исполнимый файл, имеющий стандартную пиктограмму такого вида:.

Посмотрите содержание папки решения.

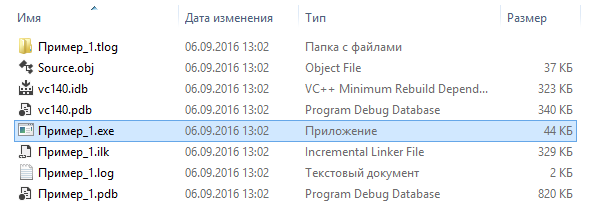


Рис. 9. Содержимое папки Debug

Экран пользователя при запуске этого варианта кода не выведет сообщение «Мир прекрасен», что связано с проблемами русскоязычной кодировки. Чтобы увидеть текст, нужно русифицировать консольный поток.

Выполните два действия:

1) добавьте директиву

#include <locale>

2) в main используйте настройку

setlocale (LC\_ALL,".1251");

Внесите эти изменения и запустите отладчик снова.

1.11. Сообщения об ошибках

В процессе выполнения компиляции, сборки и выполнения возможно появление ошибок. Сообщение об ошибке (Error) или предупреждение (Warning) появляются в окне сообщений внизу рабочего стола. После их исправления необходим повторный запуск программы.

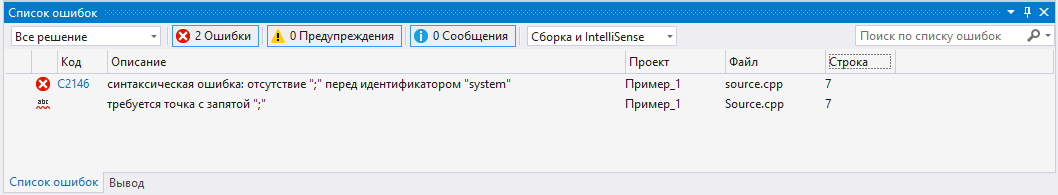


Рис. 10. Окно сообщений «Ошибки» с сообщением об ошибке

В окне появляется сообщение об ошибке [C++Error]. Для ошибки определены:

* локализация – имя файла и номер строки (Source.cpp (7));
* код ошибки (**С2146**);
* краткое описание смысла: синтаксическая ошибка: отсутствие ″;″.

Сообщение об ошибке в окне вывода связано с ее местом в коде программы. Так, щелчок мышью на тексте сообщения локализует местоположение ошибки в программном коде.

Удалите знак «;» в конце пятой строки, запустите отладку,   
и получите результат, как на рисунке выше.

Справку об ошибках получаем от автора – на MSDN. MSDN , это информационно-справочная система от Microsoft.

1.12. Схема выполнения задачи в IDE разработчика

Примерная схема прохождения задачи в интегрированной среде разработки приведена на рисунке. Код программы проходит три этапа, прежде чем будет создано работающее приложение:

* препроцессорная обработка;
* компиляция;
* компоновка (сборка).

Исходный текст  
программы **имя.срр**

Препроцессор

Полный текст  
программы **имя.срр**

Компилятор

Объектный код  
программы **имя.obj**

Исполняемый модуль

**имя.exe**

Компоновщик

Библиотечные **оbj** файлы

Включаемые файлы

**имя.h**

Рис. Схема выполнения задачи в интегрированной среде разработки

В двойной рамке показаны тексты, обрабатываемые в интегрированной среде, в одинарной рамке – модули их обработки, входящие в состав интегрированной среды. Исходный текст программы на С++ представляет собой обычный текстовый файл (один или несколько).

Три обязательных этапа обработки текста программы – это :

* препроцессорное преобразование;
* компиляция;
* компоновка.

Задачей препроцессора является преобразование текста программы до ее компиляции. Препроцессор сканирует текст, находит в нем команды, называемые директивами препроцессора, и выполняет их. В результате происходят изменения в исходном тексте, вставки фрагментов текста и замена некоторых его фрагментов. Полученный текст называется полным текстом программы.

Далее этот текст проходит этап компиляции, на котором исходный код преобразуется во внутреннее машинное представление, некоторую последовательность команд, которая понятна компьютеру. Результатом является построение объектного кода (obj файлы).

На этапе компоновки происходит редактирование связей и сборка исполнимого текста программы. Компоновщик обрабатывает все вызовы библиотечных функций и выполняет их подключение. Таким образом, к компилированному исходному коду добавляются необходимые функции стандартных библиотек. Готовый код является исполнимым и может быть выполнен компьютером при его запуске.

Каждый из этапов может быть выполнен отдельно с помощью пунктов меню *Построение* – только компиляция (*Компилировать* – Ctrl/F7) проекта или отдельного модуля, общая сборка (*Построить решение*– F7) или инкрементная сборка (*Перестроить решение*). Компиляция и сборка инкрементны, то есть перекомпилируются только те модули, в код которых были внесены изменения.

**Итоги**

**Вопросы для проверки**

1. Чем отличается алгоритм от программы?

2. Каково назначение языков программирования?

3. Чем отличаются языки программирования от естественных языков?

4. Что такое синтаксис и семантика?

5. Зачем нужны интегрированные среды разработки программ?

6. Какие функции имеет интегрированная среда разработки программ?

7. Каков смысл понятия «Проект», чем проект отличается от решения?

8. Какие файлы входят в состав решения?

9. Какие этапы проходит программа, прежде чем может быть выполнена?

10. В чем смысл препроцессорной обработки программы?

11. В чем смысл компиляции программы?

12. В чем смысл сборки программы?

13. Что находится на рабочем пространстве IDE?

14. Как управлять окнами IDE?

15. Какова роль окна обозревателя решений?

16. Какие операции доступны в менеджере файлов IDE?

17. Каково назначение отладчика IDE?

18. Когда работает отладчик IDE?

19. Каковы основные средства отладки?

20. Что означает «пошаговое исполнение программы»?

Тема 2. Данные в языке С++.  
Основные типы данных. Константы и переменные. Ввод и вывод данных

2.1. Данные: определение и концепция типов

Данные в языке программирования, это все, что подлежит обработке с помощью программы независимо от формата представления (числовые данные, текстовые, графика, звук и прочие). Для машинной обработки данные должны быть представлены в формализованном виде, то есть структурированы.

Классификация данных:

а) **по типам** – базовые и конструируемые, где базовые типы встроены в реализацию языка и предопределены ею.

б) **по способу организации** – независимо от типа, данные могут быть постоянными (константы) и изменяемыми (переменные).

**Тип данного** определяет способ хранения данного и представления в оперативной памяти, а именно:

1) размер выделенной памяти (в байтах);

2) представление данного внутри этого адресного пространства (различно для целых, вещественных, символьных и прочих данных);

3) как следствие, множество допустимых значений данного (зависит от объема и способа представления);

4) как следствие, множество допустимых операций над данным.

2.2. Основные базовые типы

Основные базовые типы определены ключевыми словами:

char символьный (1 байт – целое значение).

int целый (4 байта).

float с плавающей точкой (4 байта).

double двойной точности (8 байт).

Замечание: для определения размера памяти, выделенной типу данных или объекту используется операция sizeof():

sizeof (int) // Имя типа.

sizeof (My\_object) // Имя объекта.

Специальный тип void используется для адресации адресного пространства произвольной длины.

Тип констант задается двумя способами.

1. Записью в коде программы, например:

-9.9 – числовое значение с плавающей точкой.

12 – целочисленное десятичное значение.

'1' – символьная константа.

"Text" – строковая константа.

2. Объявлением именованной константы:

**const** Тип Имя = Значение;

**const** **int** a=5; // Тип **int**,значение =5,имя a.

**Переменная** – величина, изменяемая при выполнении программы. В тексте присутствует своим именем (уникальный идентификатор).

Тип переменной должен быть объявлен, например:

**float** x1, y1; // 4 байта, с плавающей точкой.

**int** a, b; // 4 байта, в обратном коде.

**char** symb = ‘I’; // 1 байт, символ ACCII.

Роль объявления – выделение памяти для размещения объекта.

2.3. Средства языка для ввода и вывода данных

Ввод данных, это присваивание значений объектам программы в процессе выполнения. Вывод, это отображение данных на экране при выполнении программы.

Все существующие способы ввода-вывода реализованы посредством библиотек. В этом разделе рассмотрим потоковый ввод-вывод, использующий библиотеку классов iostream.

Поток, это последовательность байтов (символов), не зависящая от устройства обмена данными. Обмен является буферизованным, где буфер, это область памяти, исполняющая роль промежуточной ступени при обмене информацией. С клавиатуры в поток отсылаются символьные данные. Распознаются отдельные лексемы текса, преобразуются к нужному представлению – число, строка, символ, и присваиваются переменной.

Библиотека потокового ввода-вывода подключается директивой  
#include <iostream>.

Со стандартным входным потоком (консоль) связан объект cin. Со стандартным выходным потоком (экран) связан объект cout.

Операции ввода-вывода определены знаками операции << и >>.

При извлечении данных из потока выполняется преобразование символов потока в значения переменных базовых типов:

cin >> переменная\_базового\_типа;

При выводе в поток пересылается символьное представление данного, соответствующее его внешнему представлению как данного базового типа.

cout << выражение\_базового\_типа;

При выполнении ввода данные отделяются друг от друга любым пробельным символом.

2.4. Упражнения

Теперь можно приступить к решению простых задач. В простой задаче с линейным алгоритмом нужно ввести данные, выполнить вычисления по формулам и вывести результат. В упражнениях необходимо выбрать правильно тип входных данных, использовать операции целочисленного деления, функции округления.

Упражнение 1.

Цистерну ГСМ объемом N литров нужно разлить в бочки объемом 200 л. Остаток разлить по канистрам объемом 20 л. Сколько нужно бочек, сколько канистр? Сколько литров останется?

Упражнение 2.

При вычислении процентов от денежных сумм получаются дробные значения, например, если вклад равен 1000 руб., то процент по вкладу, равный 3,5, даст последовательно значения: 350 руб., 472.5 руб., 165.375 руб. и так далее. Задача: указанную сумму перевести в рубли и копейки, используя округление.

**Итоги**

**Вопросы для самопроверки**

1. Что такое константа?

2. Что такое переменная?

3. Каким может быть имя переменной?

4. Зачем программе нужны комментарии?

5. Как записываются комментарии?

6. Какова роль директив препроцессора? Являются ли они обязательным компонентом программы?

7. Какова роль ввода и вывода данных?

8. Зачем программе нужен ввод и вывод?

9. Каковы механизмы преобразования данных при вводе и выводе?

10. Как данные выводятся в поток по умолчанию?

11. Какие средства позволяют улучшить вывод в поток?

12. Что такое тип данного?

13. Какова семантика объявления переменной?

14. Какое значение имеет переменная при объявлении?

15. Что такое инициализация переменной?

Тема 3. Средства отладки программ

Для поиска и устранения ошибок (отладки) в сложных программах необходимо использовать средства отладчика. В Visual C++ они сосредоточенные в меню *Отладка*.

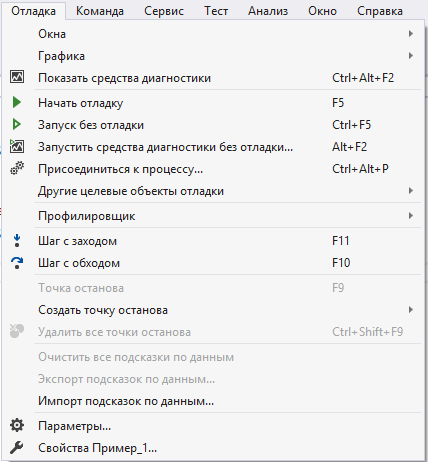


Рис. 12. Содержание пункта меню «Отладка»

Основными средствами отладки являются.

1. Пошаговое исполнение программы (*Шаг с заходом* или *Шаг с обходом*).

2. Переход до курсора (Точки останова). После этого, как правило, далее программа выполняется по шагам.

**Пошаговое** (по одной строке кода) **исполнение** выполняется при нажатии клавиши F11. Пошаговое исполнение имеет целью наблюдение изменений значений объектов программы (переменных) при выполнении алгоритма.

3. Окна наблюдения. Для отображения значений объектов существуют окна наблюдения, из которых удобным средством является окно просмотра локальных или видимых переменных. В режиме трассировки (исполнения по шагам) окна просмотра переменных включаются автоматически, и содержат значения локальных или видимых переменных. В окне редактора кода курсор трассировки выделяет исполняемую строку кода (желтая стрелка слева на рамке окна).

Выполните программу в режиме трассировки.

В окне наблюдения проследите, как изменяются переменные.

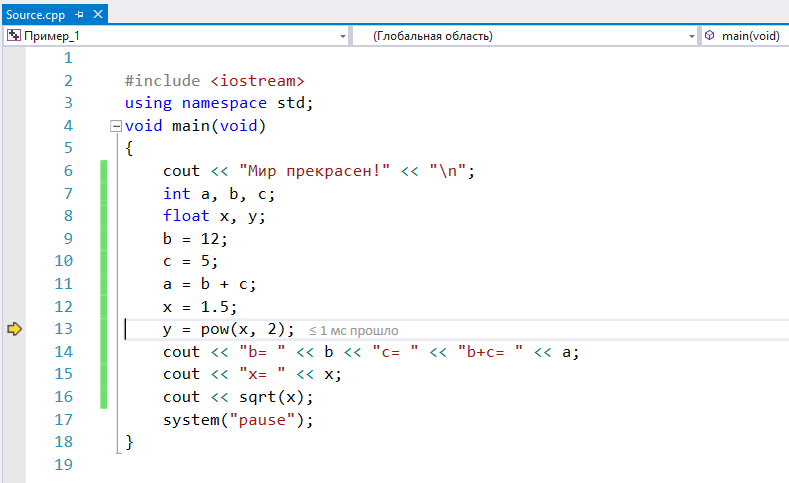


Рис. 13. Вид окна редактора при работе в режиме трассировки

Значения всех доступных объектов модуля видны в окне переменных как видимые или локальные. Эти значения изменяются при выполнении строк кода в режиме исполнения по шагам.

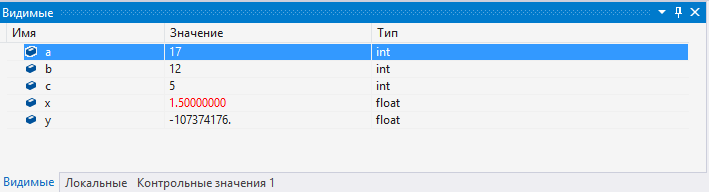


Рис. 14. Вид окна «Видимые» при работе в режиме трассировки

Точки остановки ставятся щелчком мыши на левом поле окна редактирования. При запуске программы выполнение произойдет до точки останова, где и прекратится. Далее, как правило, программу выполняют в пошаговом режиме.

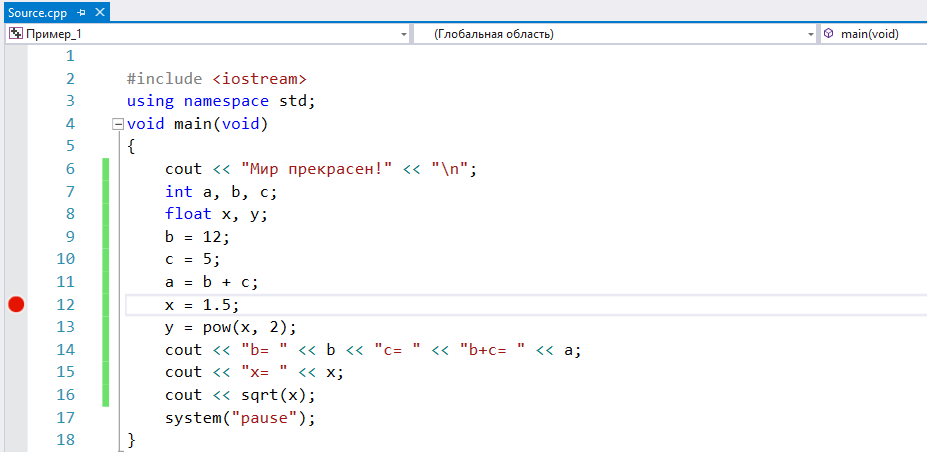


Рис. 15. Точка остановки

Не все переменные модуля должны наблюдаться при отладке. Для детального просмотра используется окно наблюдения *Контрольные значения*. Объекты в него добавляются из пункта меню *Отладка→Окна→Контрольные значения.* Объектом является имя любой переменной программы. Добавление и удаление объектов осуществляются через контекстное меню.

Тема 4. Составляющие IDE Visual C++

Интегрированная среда разработчика Visual Studio включает в себя много составляющих, которые дают разработчику программ инструменты для написания и отладки программного кода.

1. В первую очередь, это языки программирования (С#, С++ и другие).

2. Редактор текста для ввода и редактирования текста программ.

3. Компиляторы.

4. Компоновщик.

5. Отладчик для отладки программ.

6. Менеджер файлов.

7. Менеджер проектов.

8. Справочную систему.

9. Средства настройки параметров среды.

10. Для создания Windows приложений используются механизмы двунаправленной разработки приложений и библиотеки компонент, от использования которой мы отказались при создании консольного приложения.

4.1. Рабочее пространство

Рабочее пространство среды заполнено окнами самого разнообразного назначения и степени необходимости. Окна условно можно классифицировать по некоторым группам.

1. Окна редактирования программного кода – окна редактора кода выглядят как вкладки в многостраничном блокноте. Их может быть открыто одновременно много, на заголовке вкладки – имя редактируемого файла.

2. Рабочие окна: окно сообщений, окно вывода результатов, окна отладки, окно менеджера проекта и другие. Имя рабочего окна указано сверху на рамке окна.

Рабочие окна могут быть плавающими или припаркованными к определенному месту рабочего пространства. Управляет закрепленностью окна свойство Dockable, которое доступно в контекстном меню окна.

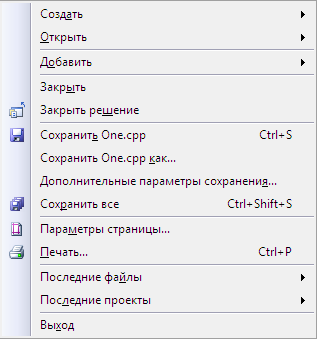
Закрыть окно можно кнопкой закрытия, открыть для работы – командой меню *Вид*.

3. Окна диалога используются для настройки параметров команд. Имеют постоянный размер и характерные элементы управления (кнопки, флажки, списки, поля ввода).

4.2. Краткий обзор команд меню и панелей инструментов

Вход в меню выполняется обычным образом. Меню является выпадающим и группирует команды среды по выполняемым действиям. Назначение пункта меню объявлено его названием.

4.2.1. Пункт меню Файл



Это менеджер файлов проекта, который позволяет выполнить файловые операции в рамках интегрированной среды разработчика.

**Создать** – позволяет создать новый проект в составе решения или новый файл определенного типа в составе существующего проекта.

**Открыть** – позволяет открыть существующее решение или проект. Также возможно открыть произвольный файл, в том числе обычный текстовый файл, в окне редактора.

**Добавить** – позволяет включить в решение новый проект или уже существующий. Также возможно добавить в состав проекта произвольный файл, например, заголовочный файл с описаниями функций.

**Закрыть** и **Закрыть решение** – позволяют закрыть файл, находящийся в активной вкладке окна редактирования или решение.

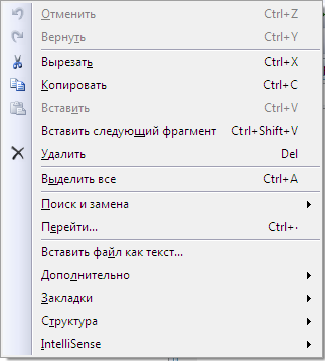
**Сохранить** и **Сохранить как** позволяют сохранить файл, находящийся в активной вкладке окна редактирования.

**Сохранить все** – принудительное сохранение всех файлов проекта.

**Параметры страницы** и **Печать** – обычные диалоги вывода на печать файлов проекта. Необходимо настроить опции печати, например, печатать текущий файл или все файлы проекта.

**Последние файлы** и **Последние проекты** – назначение очевидно.

4.2.2. Пункт меню Правка



Правка позволяет осуществить обычные операции редактирования текста, большинство из которых привычнее выполняются через кнопки панели инструментов или горячие клавиши, во-вторых, выполнить быстрый поиск по тексту любого файла проекта или поиск с заменой.

Отменить (Undo), вернуть (Redo) управляют откаткой.

Команды **Вырезать**, **Копировать**, **Вставить**, **Удалить** и **Выделить все** не нуждаются в пояснениях.

Команда **Поиск и замена** позволяет выбрать варианты поиска подстроки (возможно, с заменой), в текущем файле или в любом файле проекта или во всех файлах проекта. Открывающиеся диалоги позволяют реализовать множество вариантов поиска – в текущем файле, во всех файлах проекта, с учетом направления поиска и так далее.

Дополнительно в этом пункте меню существуют команды, которые позволяют настраивать опции отображения кода в окне редактирования, управлять закладками в коде, управлять структурой кода и вызывать команды инструмента **IntelliSense** для интеллектуальной помощи при редактировании кода.

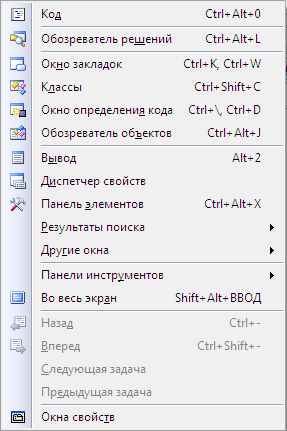
**Дополнительно** –

**Закладки** –

**Структура** –

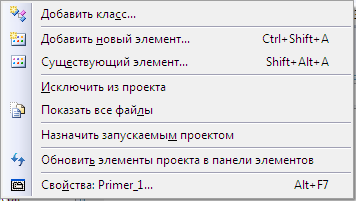
**IntelliSense** –

4.2.3. Пункт меню Вид



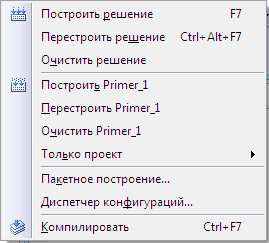
Пункт меню *Вид* позволяет управлять отображением окон, необходимых для работы в IDE.

4.2.4. Пункт меню Проект



Пункт меню *Проект* позволяет управлять составом и настраивать свойства проекта.

4.2.5. Пункт меню Построение



Пункт меню *Построение* содержит команды выбора способа построения исполнимого файла.

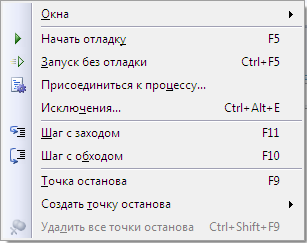
**Построить решение** и **Перестроить решение** – сборка (построение исполнимого кода) для решения в целом.

**Очистить решение** – обновляет отладочную информацию.

**Построить и Перестроить –** сборка для проекта.

**Компилировать** – только компиляция в объектный код

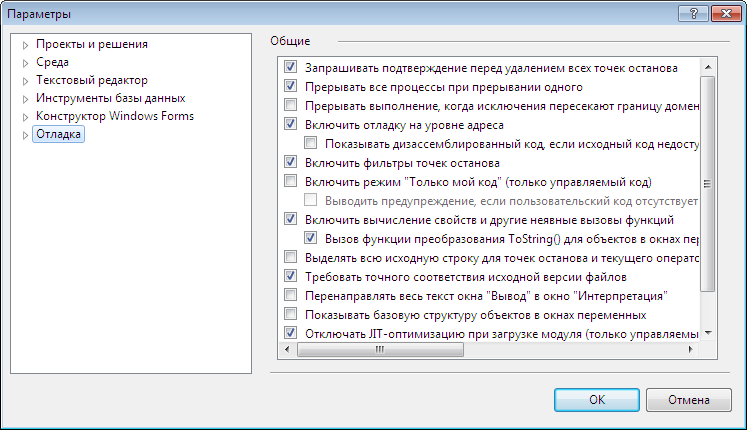
4.2.6. Пункт меню Отладка



Пункт меню *Отладка* содержит средства отладки программ.

4.2.7. Пункт меню Сервис

Пункт меню *Сервис*→*Параметры* позволяет настроить компоненты среды разработчика.



Проекты и решения – управление настройками решения.

Среда – управление средой окружения.

Текстовый редактор – настройка свойств редактора кода.

Отладка – настройка опций отладчика.

Настройки хранятся в файлах конфигурации проекта.

4.2.8. Пункт меню Окно

Управление отображением окон.

4.2.9. Пункт меню Справка

4.2.10. Роль горячих клавиш и контекстного меню