

Частное учреждение образования
«Колледж бизнеса и права»

УТВЕРЖДАЮ

Заведующий

методическим кабинетом

_____ Е.В.Фалей

« ____ » _____ 201__

| | |
|--|---|
| Специальность 2-40 01 01 «Программное обеспечение информационных технологий» | Дисциплина «Основы алгоритмизации и программирование» |
|--|---|

Лабораторная работа № 1
Инструкционно-технологическая карта

Тема: знакомство со средой разработки Microsoft Visual Studio 2019. Знакомство с типами данных языка разработки C++.

Цель: научиться создавать программы на языке C++ в среде разработки Microsoft Visual Studio 2019.

Время выполнения: 2 часа

Приводимый в примерах код набирать и компилировать.

Номер варианта задания соответствует вашему текущему номеру по списку учащихся группы (по предмету ОАиП в журнале группы).

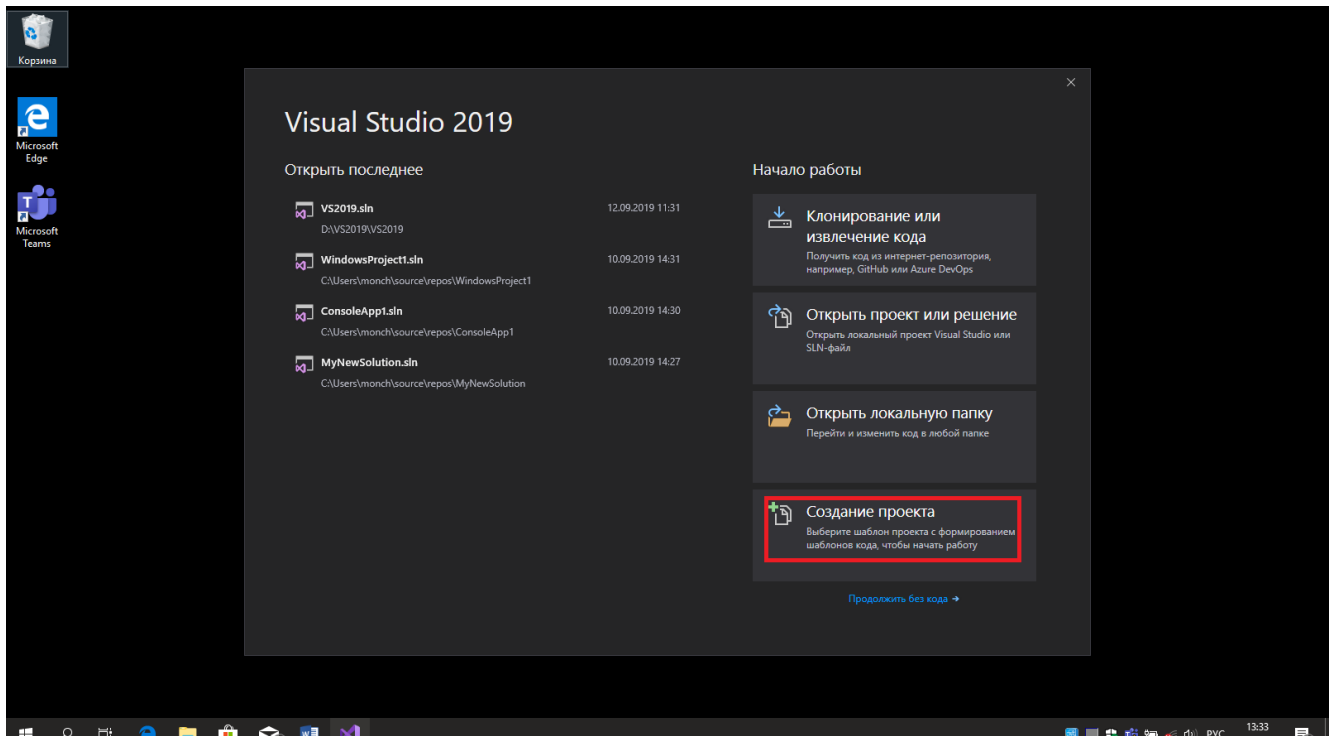
Код примеров и код программ-заданий, выполненных по варианту, иметь при себе для защиты лабораторной работы № 1.

Домашнее задание: прочитать страницы 1 – 98 книги Дейтел, Х. Как программировать на C++ / Х.Дейтел, П.Дейтел (путь на сервере: s1 / Предметы / ОАиП_Шаляпин / Дейтел Харви - Как программировать на C++.pdf).

1. Краткие теоретические сведения

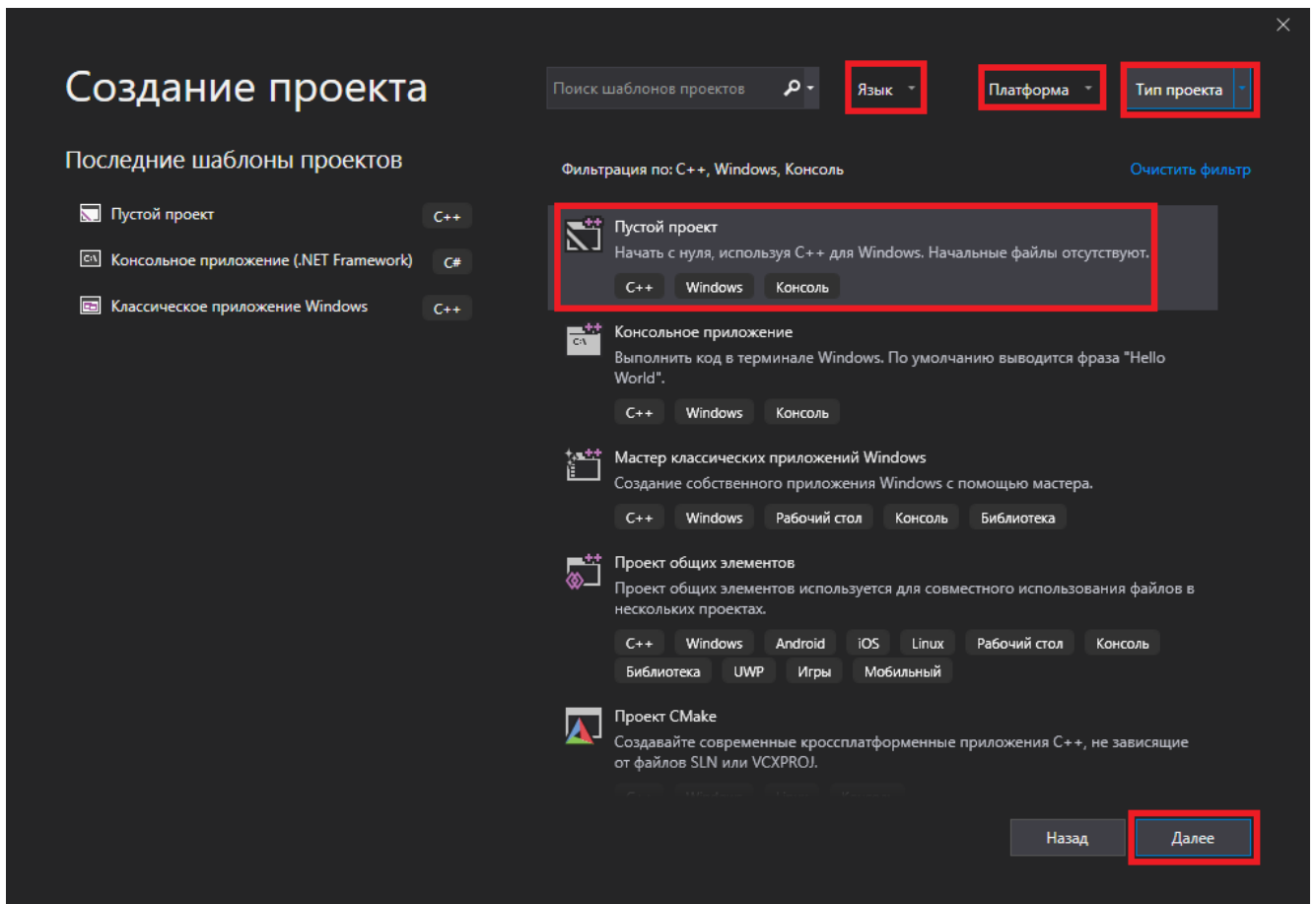
Знакомство со средой разработки Microsoft Visual Studio 2019.

1. Двойным щелчком левой кнопки мыши нажать на ярлык программы Microsoft Visual Studio Community 2019 или выбрать программу Visual Studio 2019 в списке программ на вашем компьютере (ПК – персональном компьютере). Дождаться заставки программы и появления окна начального меню:



Нажать на кнопку Создание проекта.

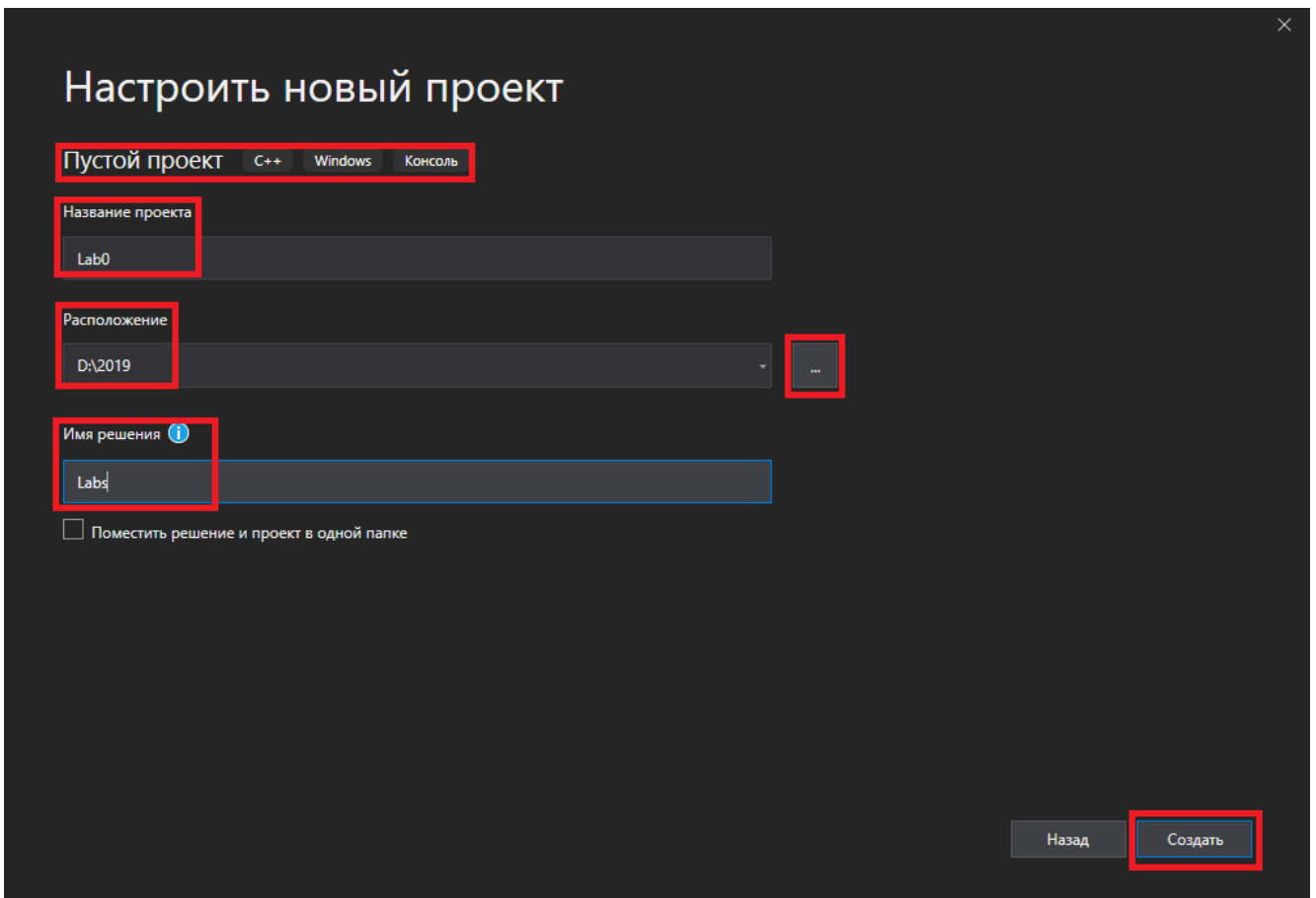
В новом окне нажать на кнопку Язык и выбрать C++, Платформа – Windows, Тип проекта – Консоль. Из предложенных вариантов выбрать Пустой проект и нажать Далее. *Если вы не увидели вариантов создания пустого проекта на C++, то, видимо, вы не установили модуль работы с C++ или установили модули других (пока не нужных) языков и проектов при установке Visual Studio 2019 (VS2019). Значит, вам нужно нажать на подчеркнутую ссылку в окне предложенных вариантов и доустановить с сайта Microsoft требуемый модуль.*



В новом окне проверяем тип и язык создаваемого проекта. Проект должен быть именно Пустым. Вводим название проекта (англоязычное краткое без пробелов), выбираем для него расположение на диске, где НЕ установлена операционная система (обычно это диск D:\). Можно самому набрать путь сохранения вашего решения и проекта, а можно нажать на кнопку «...» (три точки) и выбрать место сохранения вашего решения и проекта в проводнике.

Проект – project – это название файлов и папок той программы, которую вы собираетесь написать (создать). Работу по написанию программы называют проектом, поэтому при начале создания вами каждой новой программы должен создаваться новый проект, который представляет из себя папки с файлами (артефактами). Любой проект должен находиться в Решении – Solution – это общая папка для нескольких проектов. Всегда, когда вы создаете проект, то должны либо поместить его в уже созданное ранее Решение, либо создать новое решение и в нем создать этот новый проект. Упрощенно, Решение – это папка с проектами. Любой проект должен находиться в Решении. У файла Решения расширение «.sln» (от слова solution – решение). Если хотите открыть готовый проект в проводнике файлов и запустить его для редактирования и перекомпиляции, то нужно нажать именно на файл с расширением «.sln» – тогда в VS откроется все решение с нужным вам проектом в нем.

Даем имя решению – по логике оно должно быть более общим по смыслу, поскольку включает в себя несколько проектов. Нажимаем кнопку Создать.



Должен создаваться новый проект и решение для него, и они должны открыться в новом окне VS2019. Это собственно окно для работы. В нем справа есть дочернее окно Обозреватель решений, вверху которого отображается имя решения и количество проектов в нем. Если указано, что проектов 0 из 0, а вы открывали уже созданный проект, то это значит, что, либо открыто решение без проектов, либо решение открывалось не по щелчку по файлу «.sln», а потому редактировать и перекомпилировать «старый» проект не получится – нужно закрыть VS2019 и открыть нужный «.sln»-файл либо нажать в меню вверху: Файл / Открыть / Решение или проект... / и выбрать нужное решение с проектом.

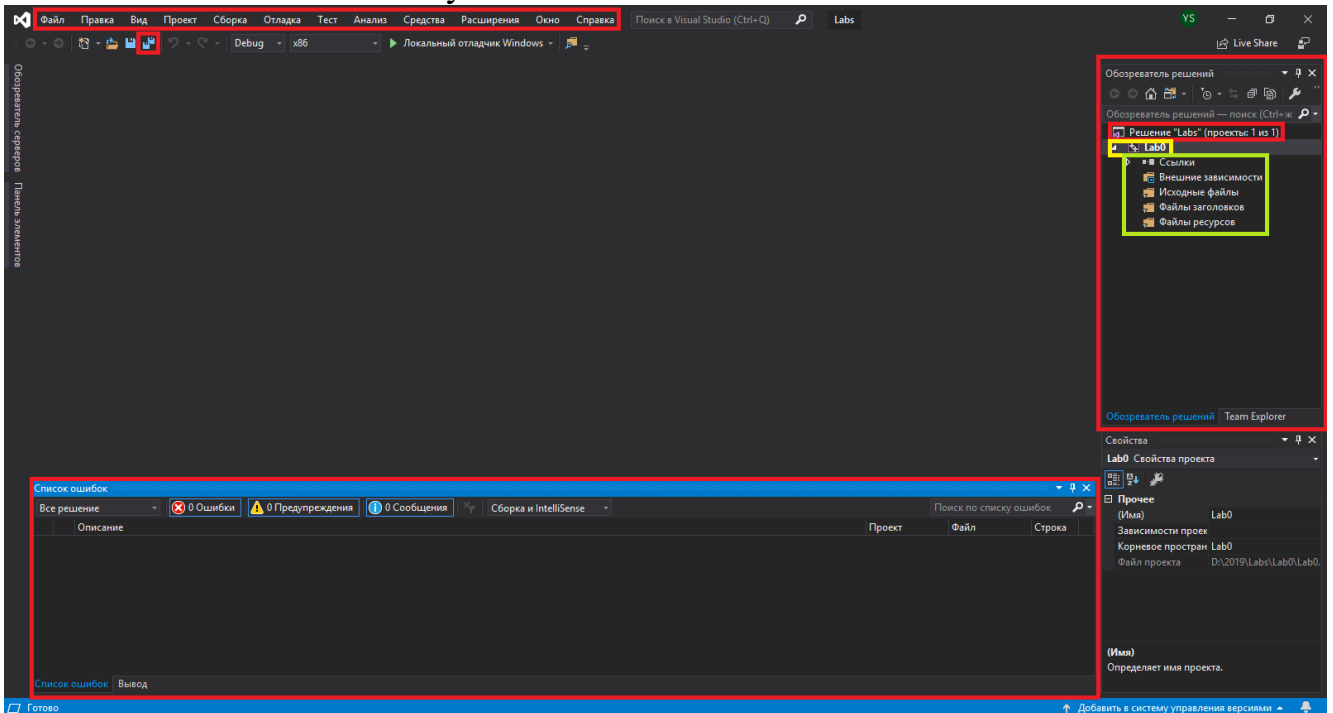
Если окна Обозреватель решений не видно, то открываем его по меню Вид / Обозреватель решений. В пункте меню Вид можно открыть и другие окна, если они вам нужны, но не отображаются в основном окне VS2019.

В моем случае Решение содержит 1 проект из всего 1 проекта, что правильно. Желтой рамкой выделено Название проекта Lab0, по которому можно нажимать левой кнопкой мыши и вызывать контекстное меню с вариантами действий. Зеленой рамкой выделено текущее содержимое проекта, которое состоит пока из пустых папок.

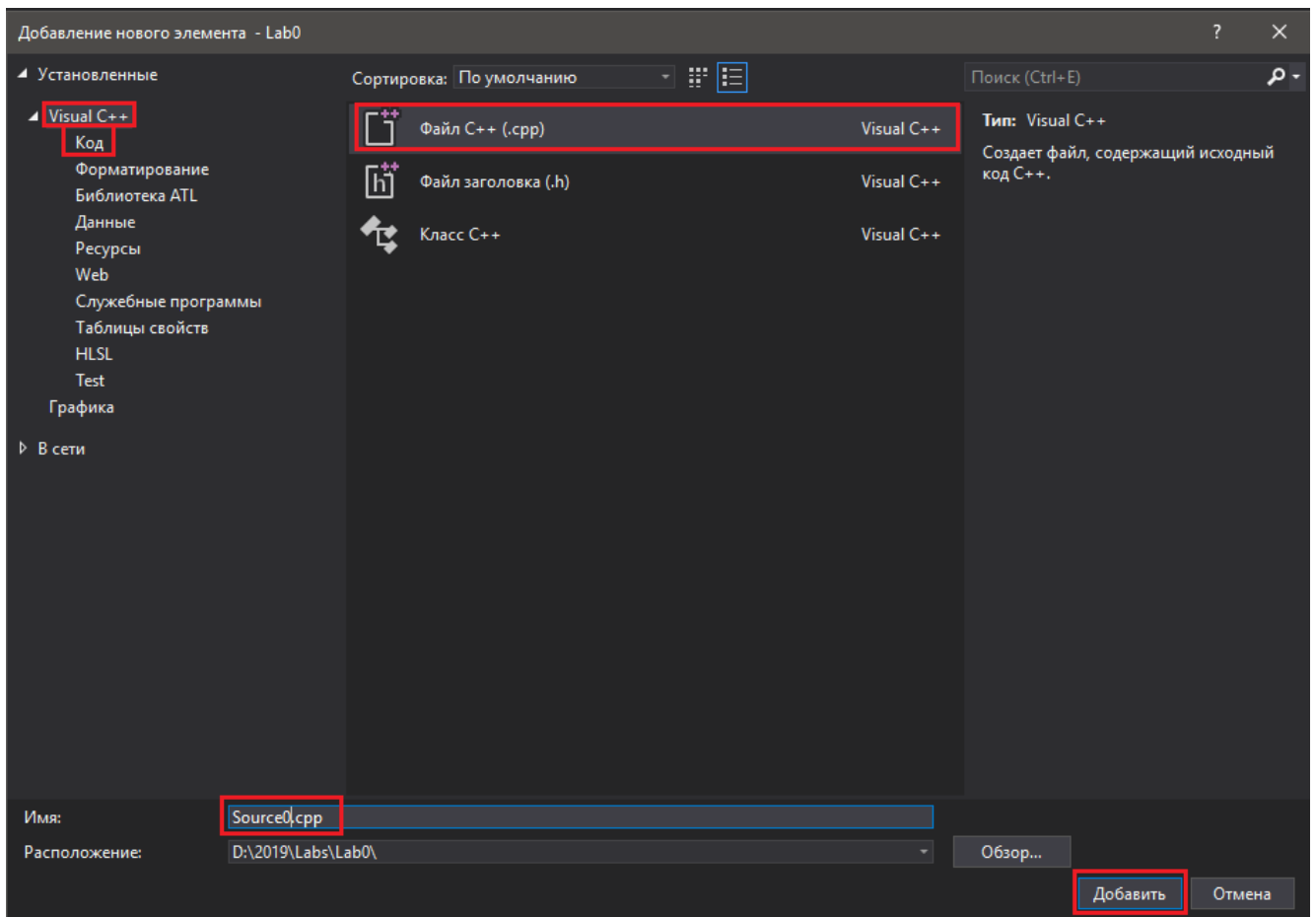
Внизу видно окно Список ошибок, в котором нужно проверять наличие ошибок в коде проекта. Ошибки не позволят скомпилировать исполнимый файл создаваемой программы «.exe». Нужно читать Предупреждения и Сообщения, поскольку там могут содержаться рекомендации по улучшению вашего кода, которые можно не выполнять. Ошибки должны быть исправлены все либо в настройки про-

екта должны быть внесены изменения, указывающие VS2019 не учитывать некоторые типы ошибок, которые раньше ошибками не считались, но сегодня считаются небезопасным кодом (кодом программы, которая может аварийно сломаться на некоторых данных или быть легко взломана злоумышленниками). Если окна Список ошибок не видно, то оно вызывается из меню Вид / Список ошибок.

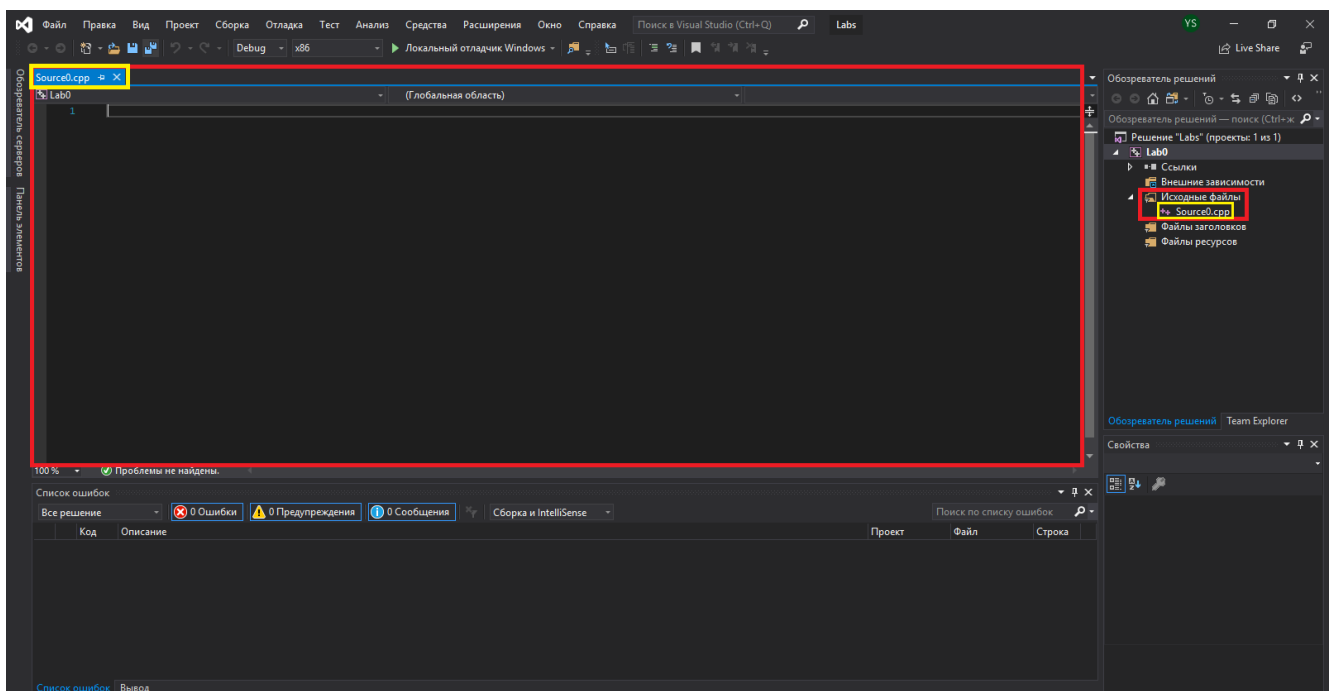
После написания очередной строки кода нажимайте на кнопку Сохранить все («Две дискеты»), чтобы сохранять вашу работу на случай выключения VS2019 или ПК. По наведению мыши на эту кнопку можно увидеть комбинацию «горячих» клавиш для ускорения работы. В VS2019 есть комбинации «горячих» клавиш на большинство часто используемых кнопок – запомните их.



В нашем решении проект есть, но этот проект не содержит файла с кодом. Для создания в проекте файла с кодом на C++ надо нажать правой кнопкой мыши по названию Проекта Lab0 и в контекстном меню выбрать Добавить / Создать элемент – появится окно:

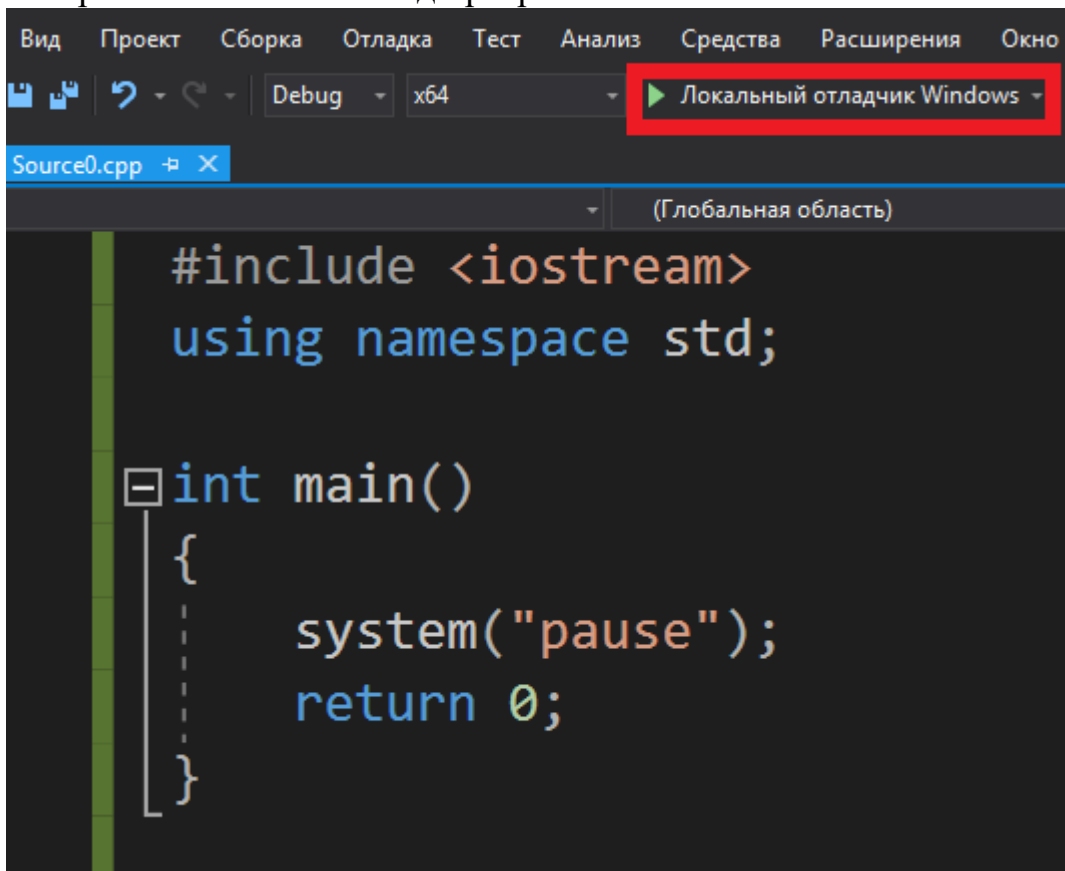


В этом окне выбираем слева Visual C++ / Код, посередине нажимаем по Файл C++ (.cpp) и внизу меняем (дописываем) Имя файла, поскольку генерируется всегда одинаковое имя, а именно cpp-файлы содержат ваш основной код, который нужно сохранять и показывать как результат вашей работы. Нажимаем кнопку Добавить.



Теперь у нас есть проект с пока пустым файлом, в который нужно писать код. Проверьте в окне Обозреватель решений, что в проекте Lab0, **именно** в папке Исходные файлы появился файл Source0.cpp. Слева этот файл должен раскрыться для ввода кода, поэтому вверху появится вкладка с именем нашего файла Source0.cpp. Если файл не открылся, то в окне Обозреватель решений дважды нажимаем левой кнопкой мыши по имени файла Source0.cpp. Если в окне Обозреватель решений папки свернулись и их содержимого не видно, то нужно дважды левой кнопкой мыши нажать на них и раскрыть, чтобы увидеть находящиеся в них файлы (артефакты). Если файлов не видно, то есть вероятность, что они не были созданы в предыдущем пункте данной работы.

Поставим курсор в начало первой строки открытого файла Source0.cpp и наберем минимальный код программы:

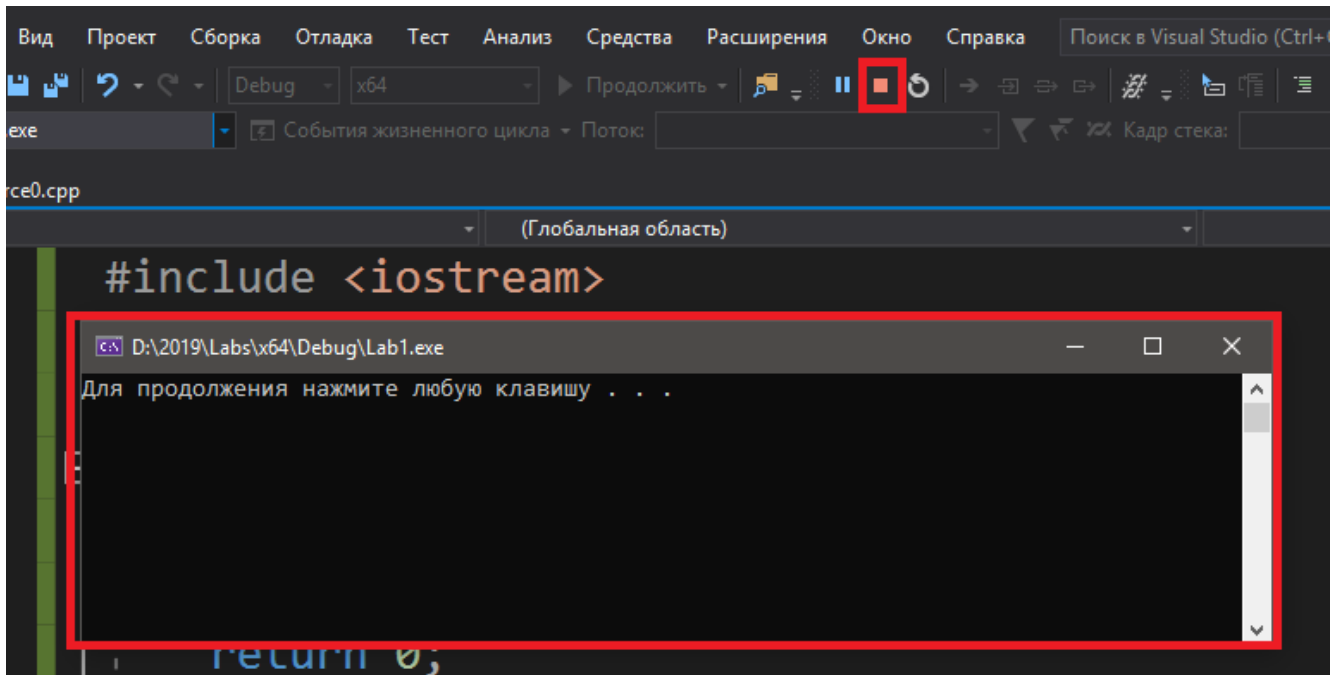


```
Вид  Проект  Сборка  Отладка  Тест  Анализ  Средства  Расширения  Окно
[Иконки] [Отмена] [Отмена] [Отмена] Debug x64 [Локальный отладчик Windows]
Source0.cpp [X]
(Глобальная область)

#include <iostream>
using namespace std;

int main()
{
    system("pause");
    return 0;
}
```

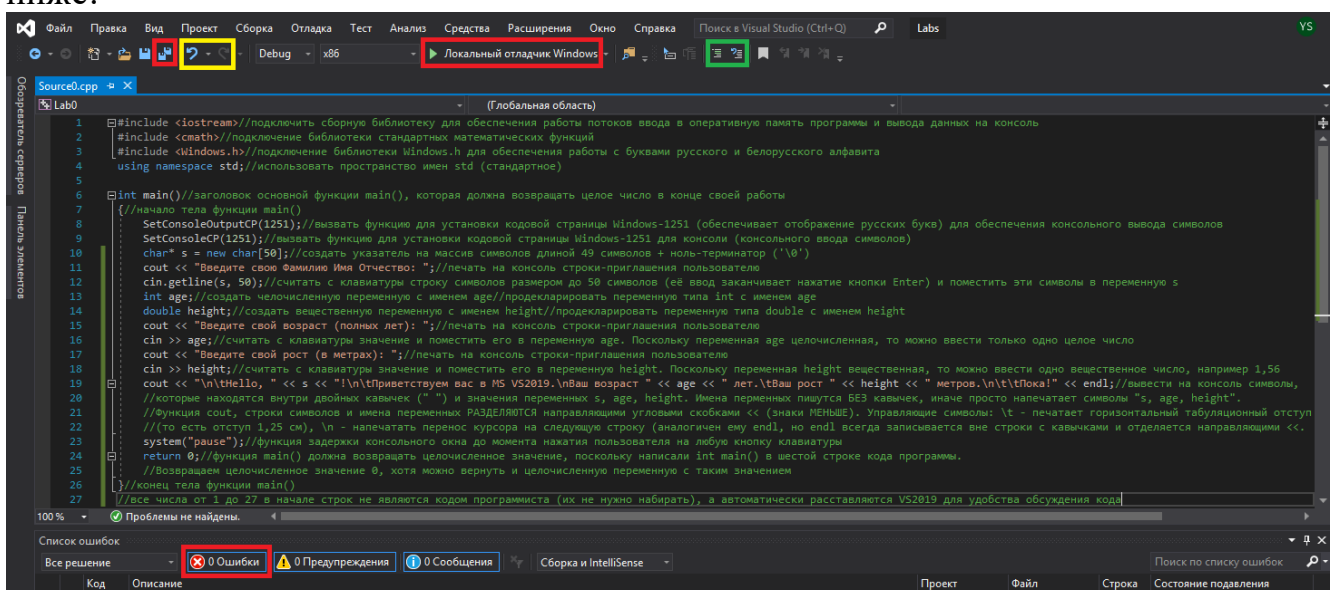
Этот код нужно набирать в .cpp-файле любого нового пустого проекта и запускать на выполнение кнопкой «зеленый треугольник» или в верхнем меню выбрать пункты Отладка / Начать отладку или кнопкой на клавиатуре F5. Так вы проверяете, что проект действительно создан, он автозагружаемый и написан на нужном языке. При выполнении вы должны увидеть следующее окно:



Вверху выделена кнопка принудительного завершения вашей программы, которой вы можете воспользоваться при зависании вашей программы. Старайтесь не закрывать эту кнопку окном консоли.

Нажмите, как указано в окне консоли, на любую кнопку на клавиатуре (например, Enter) и, если потребуется, еще раз. Консольное окно должно закрыться и не должно появиться дополнительных окон или фраз, сообщающих об ошибках. Запуск окна консоли и есть доказательство, что ваш проект создан успешно и работает. После этого можно ставить курсор ПЕРЕД функцией `system("pause")`, нажать кнопку Enter и писать нужную вам программу. Если далее вам VS2019 сообщит об ошибках, то вы уже знаете, что они связаны с вашим «новым» кодом, но сам проект создан правильно и работает, а ошибки могут находиться в дописанных новых строках кода.

Расширим функционал вашей программы. Допишите ваш код как показано ниже:



Первые три строки представляют собой код подключения библиотек (файлов, содержащих готовые функции для использования). Обычно одна библиотека содержит функции одной направленности, например, стандартные математические функции. Код, начинающийся с символа # означает директивы препроцессору (указания программе, которая проходит по коду перед компиляцией и выполняет код, начинающийся с символа #). В 4-й строке подключается пространство имен `std`, чтобы далее не писать каждый раз `std::cin`, `std::cout`, `std::endl` и так далее, но в реальной разработке может потребоваться именно последний вариант, если создаваемая программная система будет иметь свое пространство имен. Следующую строку оставим пустой, чтобы отделить код подготовки программы от кода собственно выполняемой программы – функции `main()`. Программа будет выполняться с первой строки функции `main()` до ее последней строки (в нашем коде – с 7-й до 26-й строк включительно). Все функции в большинстве случаев пишутся с парой круглых скобок после своего имени: `имяФункции()`.

Весь код зеленого цвета представляет собой комментарии, которые компилятором не будут обрабатываться, но видны программистам. Компилятор – это программа, которая проходит по коду `cpp`-файла и переводит его в код, понятный текущей операционной системе и конкретному процессору. Таким образом компилятор создаст исполнимый файл – файл с расширением «.exe», который и будет являться итоговой программой, которую можно исполнять на компьютерах. Строки 8 и 9 содержат код вызова функций, обеспечивающих поддержку русского языка.

Код `cout << ...` служит для вывода на экран консоли символов и значений переменных (то есть они напечатаются внутри окошка консоли (черного окна) как будто их напечатал принтер на длинный лист бумаги). Код `cin >>...` служит для считывания значения, которое вводится пользователем посредством клавиатуры, и сохранения его в соответствующую переменную, которая будет упомянута после направляющих угловых скобок `>>`. Перед кодом по считыванию значения пользователя нужно писать код по печати на консоль фразы, предлагающей пользователю ввести конкретные значения определенного типа, чтобы пользователь знал, когда и какие значения вводить.

Функция `system("pause")` ставит консоль на паузу до тех пор, пока пользователь не нажмет на любую клавишу на клавиатуре. Она нужна в реальных программах, которые будут запускаться не из среды VS2019, а по клику мышью по соответствующему .exe-файлу.

Функция `main()` записана в виде `int main()` в 6-й строке, что означает, что мы используем версию функции `main()`, которая должна возвращать значение типа `int` в конце своей работы. Поэтому в конце кода функции `main()` – то есть перед 26-й строкой пишем код `return 0` – то есть «вернуть значение 0». Функция `main()` возвращает (посылает, сообщает) значение 0, которое означает «я завершаюсь с кодом «ноль» – то есть что наша программа проработала штатно, успешно. Неуспешно – код `-1` (минус 1), но в таких случаях программы обычно завершаются аварийно самой операционной системой, о чем она и так будет знать.

В окне выше выделены прямоугольниками кнопки Сохранить (изменения в данный файл .cpp) и Сохранить все (все изменения во всех отредактированных файлах). Если вы в .cpp-файле изменили хотя бы один символ, то название этого .cpp-

файла станет с символом «звездочка» (*) в конце сверху. Нужно сохранять изменения как можно чаще.

В желтой рамке находятся кнопки Вернуться на один шаг назад и Вернуться на один шаг вперед. Проще нажимать Ctrl+Z для отмены последнего действия.

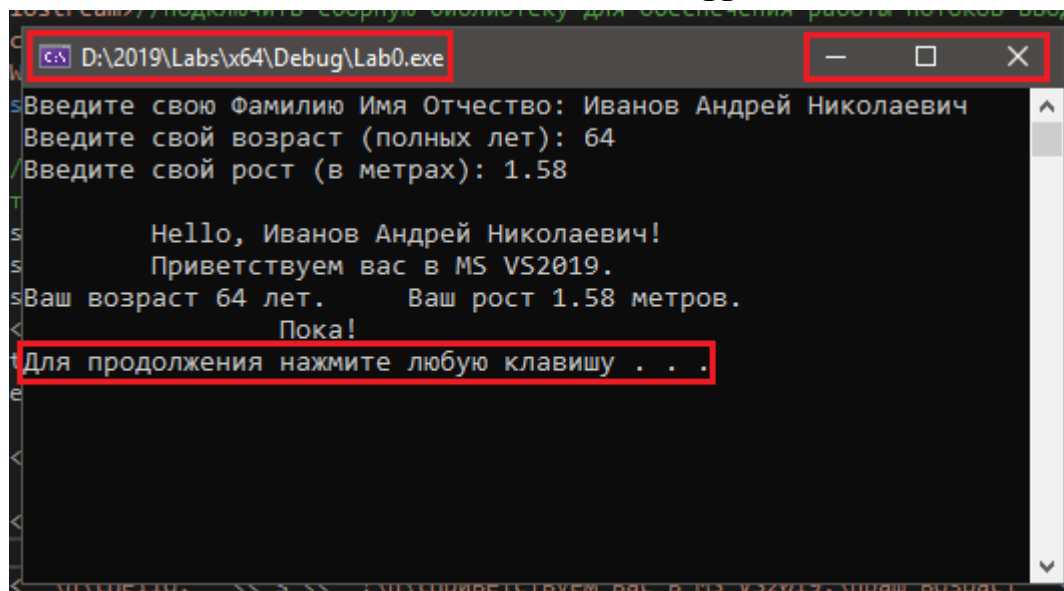
В зеленом прямоугольнике выделены кнопки Закомментировать и Раскомментировать, которые действуют на выделенный участок кода.

Внизу всегда смотрим в окно Список ошибок. Если ошибки есть, то они находятся в списке и обычно указывают на номер строки и номер символа строки, где компилятор предполагает вашу ошибку. Синтаксические ошибки – неправильно набран код, зарезервированные слова, операторы, имена переменных и функций, неправильно расставлены скобки (они должны быть парными), отсутствуют нужные пробелы и прочее. Если нажать по описанию ошибки, то обычно компилятор ставит курсор в место, где предполагает ошибку. Если ошибка общая, фатальная, то номер строки для нее не определяется. Ошибки нужно исправлять с самой ранней (первой) по коду и далее по порядку, поскольку первая ошибка может вызвать неправильную интерпретацию компилятором следующего за ней кода, а исправление первой ошибки вызовет ликвидацию всех следующих за ней ошибок.

Для запуска программы на компиляцию и исполнение нужно нажать на кнопку «зеленый треугольник» или в меню сверху выбрать Отладка / Начать отладку или нажать кнопку F5. Запустите написанную вами программу на выполнение, должно отобразиться окно консоли и строка-приглашение, в ответ на которое вы вводите свои данные. Вверху слева рамки окна консоли отображается заголовок программы с полным путем к .exe-файлу. По нему вы можете найти настоящий исполнимый файл вашей программы, скопировать его, например, на рабочий стол и запустить двойным нажатием мыши. Эту программу можно копировать и запускать на других компьютерах с аналогичным программным обеспечением (ПО) – версией ОС MS Windows и ее разрядностью, аналогичными библиотеками.

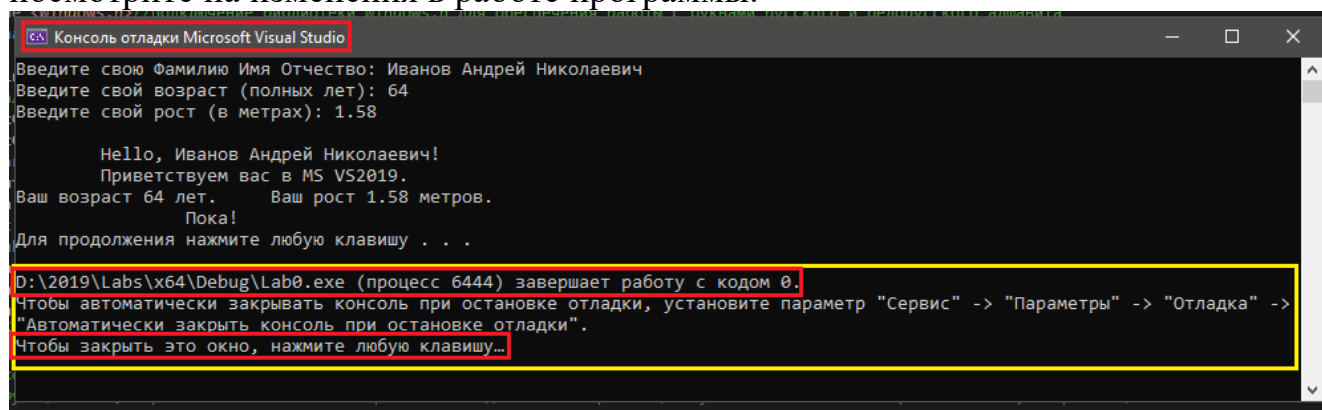
Также вверху справа рамки окна есть кнопки Свернуть, Развернуть и Закрыть окно вашей программы. Вы их не программировали – они добавлены вашему окну автоматически, поскольку окно вашей программы по умолчанию является дочерним окном общего класса Окна.

Внизу печатает фразу «Для продолжения нажмите любую клавишу...» - это работает функция system(“pause”), она поставила ваш сеанс консоли на паузу до тех пор, пока вы не нажмете на клавиатуре любую кнопку. Без этой функции процессор быстро выполнит вашу программу и окно консоли закроется за доли секунды.



Нажмите на любую кнопку клавиатуры и увидите отладочную информацию, которую дописывает в окно консоли вашей программы VS2019. Ваша выполняемая программа – это процесс, который получает уникальный номер (у меня 6444). Это позволяет скопировать в одну папку несколько экземпляров вашей программы и запустить их все – у каждого запущенного на выполнение процесса будет свой номер (идентификатор, дескриптор), поэтому они будут легко различимы операционной системой.

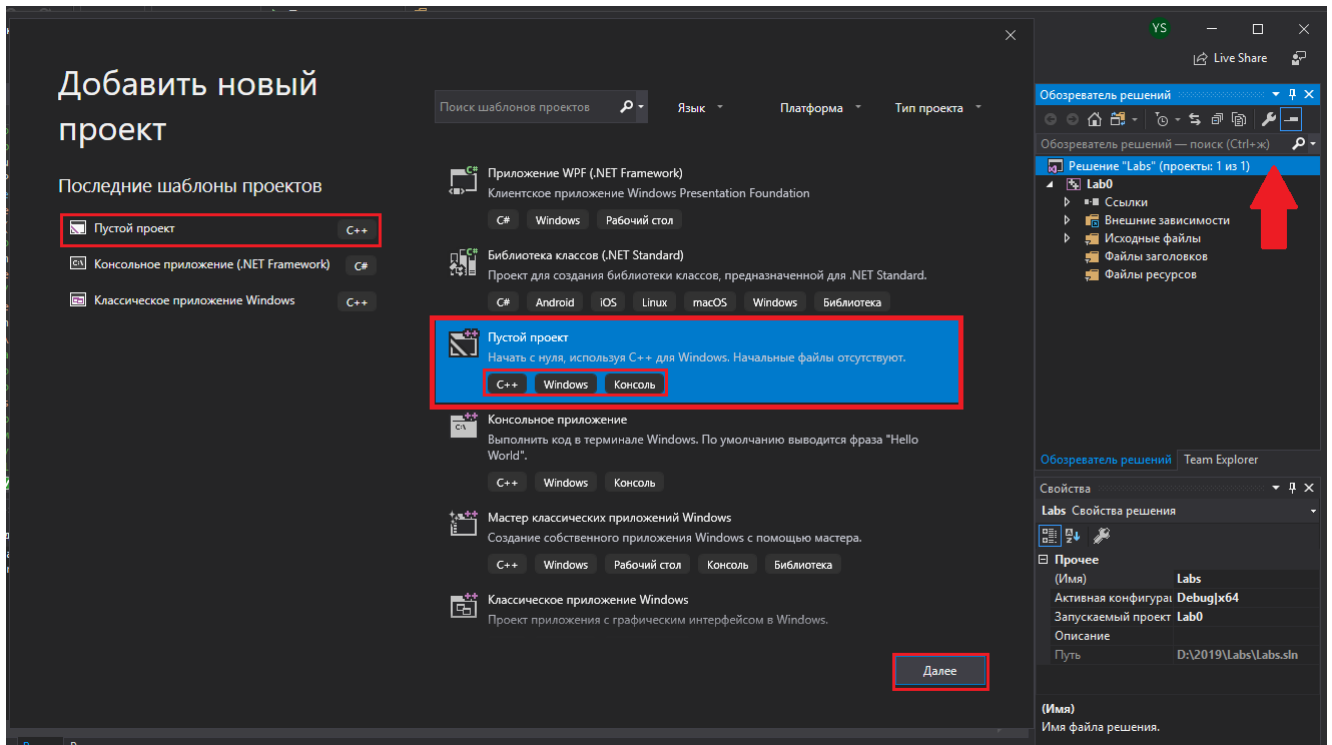
То, что ваш «процесс завершает работу с кодом 0» связано с кодом в конце вашей программы `return 0`. Замените 0 в коде 24-й строки на другое целое число и посмотрите на изменения в работе программы.



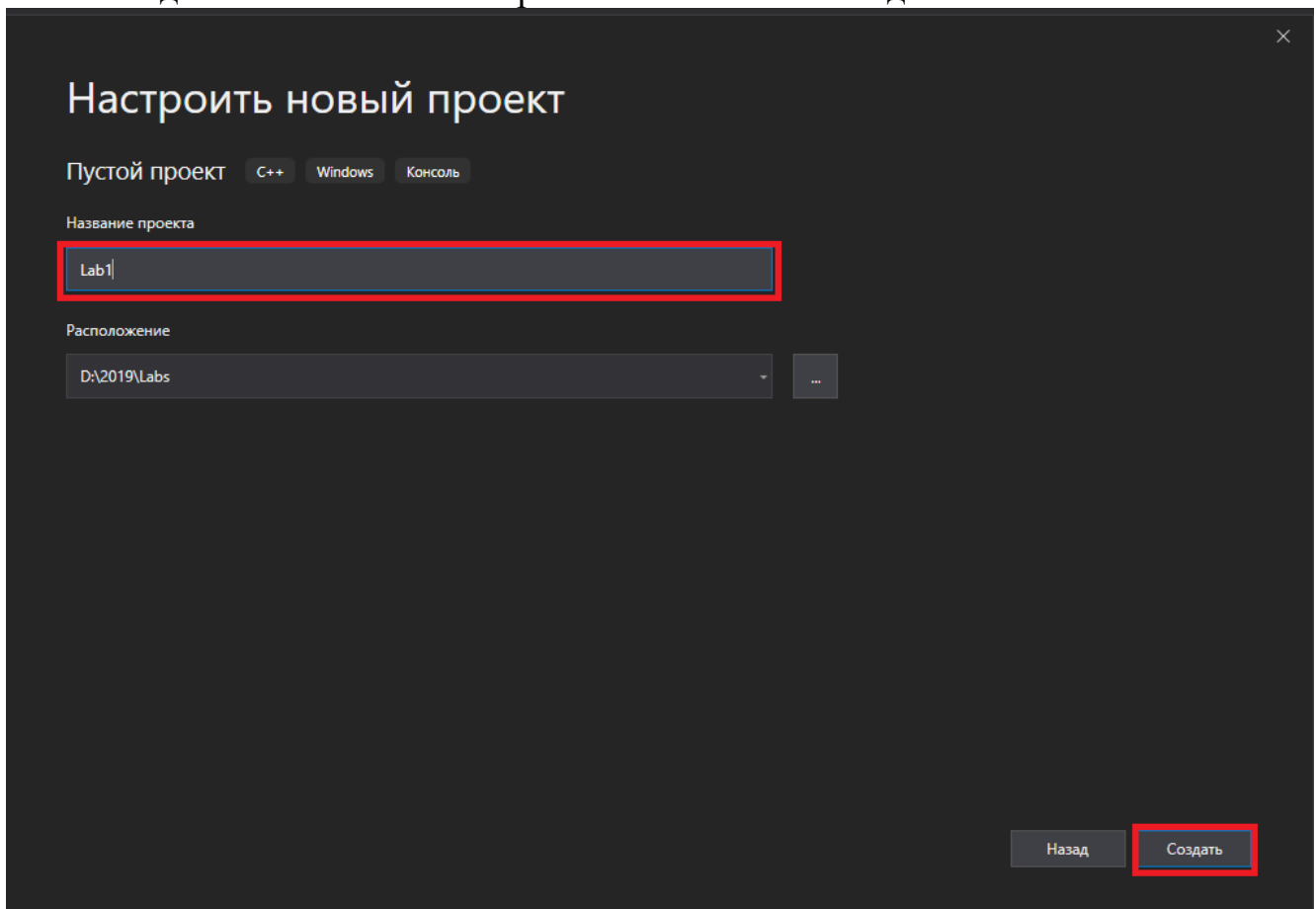
В одном решении можно создать много проектов (программ). В каждом проекте может быть только один .cpp-файл с функцией `main()`. Если в решении проектов больше одного, то нужно среди них назначать автозагружаемым проектом тот проект, программу которого вы хотите откомпилировать и запустить на выполнение.

Добавим в наше решение Labs еще один проект. Третий и последующий проекты добавляются аналогично.

Нажимаем правой кнопкой мыши по имени решения Labs, а НЕ имени проекта Lab0. Выбираем пункты контекстного меню Добавить / Создать проект... и в появившемся окне выбираем тип Пустой проект и нажимаем Далее.

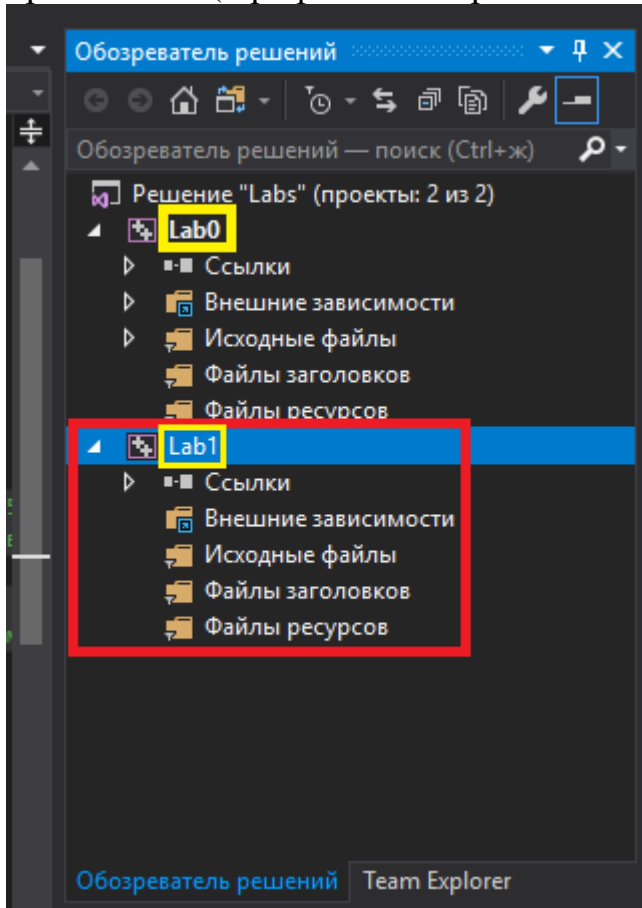


Вводим название нового проекта и нажимаем Создать.

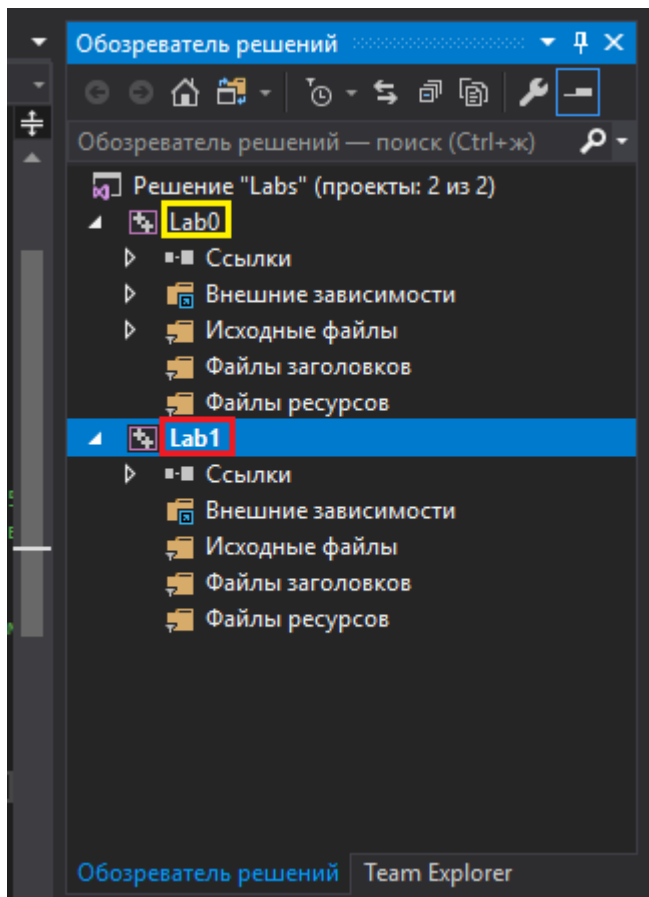


В решении Labs должен появиться проект с новым именем Lab1, причем его название отображается более «тонким» шрифтом, чем имя проекта Lab0, поскольку

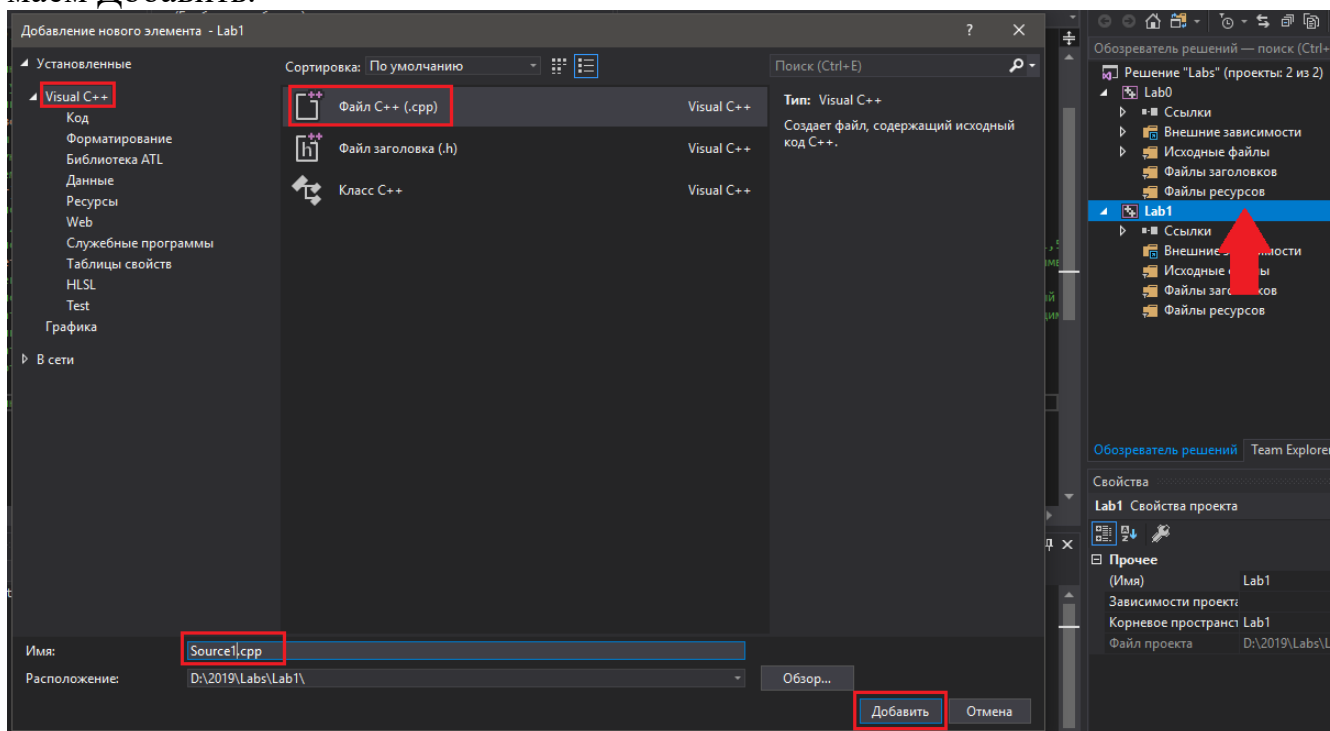
автозагружаемым проектом является пока именно первый созданный в решении проект Lab0 (шрифт его начертания более «жирный»).



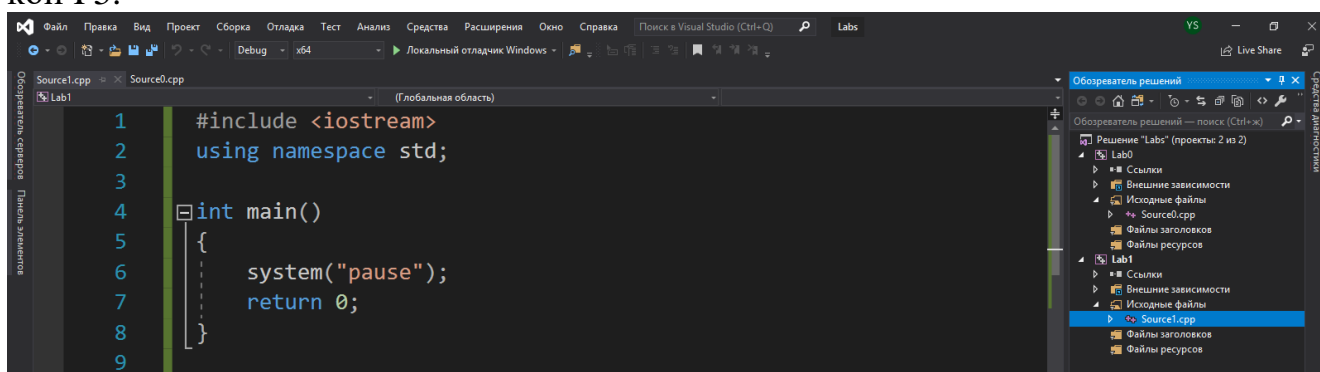
Сразу сделаем автозагружаемым проектом именно новый Lab1. Для этого нажимаем правой кнопкой мыши по его названию и выбираем пункт контекстного меню Назначить автозагружаемым проектом. Теперь его название должно стать жирного начертания, а всех остальных проектов данного решения – более тонкого начертания.



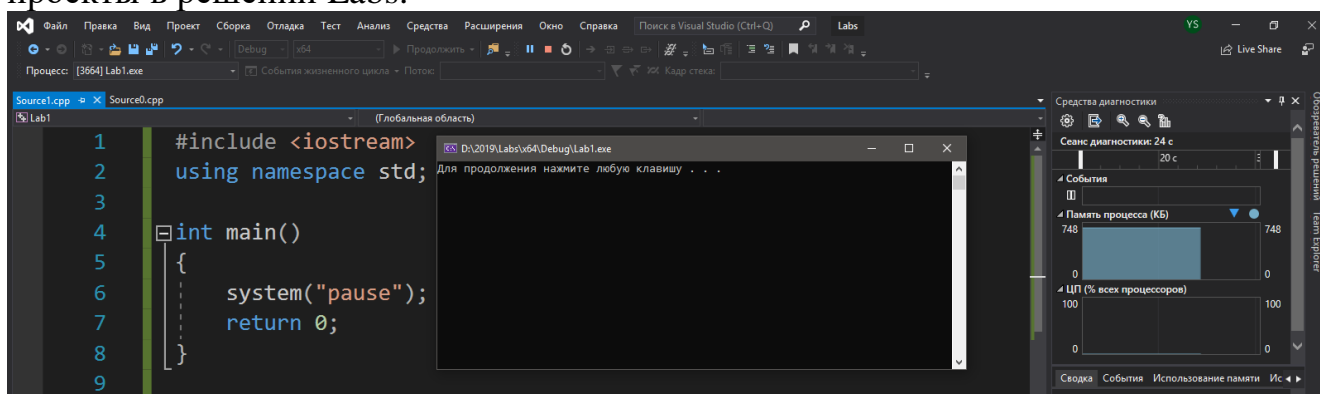
Новый проект Lab1 не содержит .cpp-файла с кодом программы, а потому нужно добавить в него этот файл. Нажимаем правой кнопкой мыши по имени Lab1 и выбираем пункты контекстного меню Добавить / Создать элемент... В появившемся окне выбираем тип файла .cpp, изменяем его имя, дописав единицу и нажимаем Добавить.



У нас создается пустой файл Source1.cpp, в который мы пишем проверочный код, в окне Список ошибок удостоверяемся, что ошибок ноль, и компилируем кнопкой F5.



Вторая программа (проект Lab1) работает. Аналогично можно создать другие проекты в решении Labs.



Типы данных C++

Любая программа в ходе работы обрабатывает данные. Хранение и обработка данных зависит от их типа. Каждая константа, переменная, результат вычисления выражения или функции должны иметь определенный тип.

Тип данных определяет:

- представление данных в памяти компьютера;
- множество значений, которые могут принимать величины данного типа;
- операции и функции, которые можно применять к величинам этого типа.

Таблица 1 – Типы данных C++

| Тип | байт | Диапазон принимаемых значений |
|---------------------------------------|------|--------------------------------|
| целочисленный (логический) тип данных | | |
| bool | 1 | 0 / 1 |
| целочисленный (символьный) тип данных | | |
| char | 1 | 0 / 255 |
| целочисленные типы данных | | |
| short int | 2 | -32 768 / 32 767 |
| unsigned short int | 2 | 0 / 65 535 |
| int | 4 | -2 147 483 648 / 2 147 483 647 |
| unsigned int | 4 | 0 / 4 294 967 295 |
| long int | 4 | -2 147 483 648 / 2 147 483 647 |

| Тип | байт | Диапазон принимаемых значений |
|--------------------------------|------|---|
| unsigned long int | 4 | 0 / 4 294 967 295 |
| типы данных с плавающей точкой | | |
| float | 4 | -2 147 483 648.0 / 2 147 483 647.0 |
| long float | 8 | -9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0 |
| double | 8 | -9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0 |

Декларация и инициализация переменной:

тип имяПеременной = значение;

int countStudents = 28;

В C++ для представления константы рекомендуется использовать объявление переменной с начальным значением и ключевым словом const:

const тип ИмяКонстанты = значение;

const int cMaxMark = 10;

Операции с переменными

Над объектами в языке Си могут выполняться различные операции:

- операции присваивания;
- операции отношения;
- арифметические;
- логические;
- сдвиговые операции.

Операция присваивания обозначается символом = и выполняется в 2 этапа: вычисляется выражение в правой части; результат присваивается операнду, стоящему в левой части: объект = выражение;

Пример:

int a = 4;

int b;

b = a + 2;

Операции отношения

Основные операции отношения:

== эквивалентно – проверка на равенство;

!= не равно – проверка на неравенство;

< меньше;

> больше;

<= меньше или равно;

>= больше или равно.

Операции отношения используются при организации условий и ветвлений. Результатом этих операций является 1 бит, значение которого равно 1, если результат выполнения операции – истина, и равно 0, если результат выполнения операции – ложь.

Арифметические операции

Основные бинарные операции, расположенные в порядке уменьшения приоритета:

* – умножение;

/ – деление;

+ – сложение;

- – вычитание;

% – остаток от целочисленного деления.

Основные унарные операции:

++ – инкрементирование (увеличение на 1);

-- – декрементирование (уменьшение на 1);

- – изменение знака.

Результат вычисления выражения, содержащего операции инкрементирования или декрементирования, зависит от того, где расположен знак операции (до объекта или после него). Если операция расположена до объекта, то сначала происходит изменение значения переменной на 1, а потом это значение используется для выполнения следующих операций. Если операция ++ или -- расположена после переменной, то сначала выполняется операция, а потом значение переменной изменяется на 1.

Бинарные арифметические операции могут быть объединены с операцией присваивания:

объект *= выражение; // объект = объект * выражение

объект /= выражение; // объект = объект / выражение

объект += выражение; // объект = объект + выражение

объект -= выражение; // объект = объект — выражение

объект %= выражение; // объект = объект % выражение

Логические операции

Логические операции делятся на две группы: условные; побитовые.

Условные логические операции чаще всего используются в операциях проверки условия if и могут выполняться над любыми объектами. Результат условной логической операции: 1 если выражение истинно; 0 если выражение ложно. Все значения, отличные от нуля, интерпретируются условными логическими операциями как истинные.

Основные условные логические операции:

&& – И (бинарная) – требуется одновременное выполнение всех операций отношения;

|| – ИЛИ (бинарная) – требуется выполнение хотя бы одной операции отношения;

! – НЕ (унарная) – требуется невыполнение операции отношения.

Побитовые логические операции оперируют с битами, каждый из которых может принимать только два значения: 0 или 1.

Основные побитовые логические операции в языке Си:

& конъюнкция (логическое И) – бинарная операция, результат которой равен 1 только когда оба операнда единичны (в общем случае – когда все операнды единичны);

| дизъюнкция (логическое ИЛИ) – бинарная операция, результат которой равен 1 когда хотя бы один из операндов равен 1;

! и ~ инверсия (логическое НЕ) – унарная операция, результат которой равен 0 если операнд единичный, и равен 1, если операнд нулевой;

^ исключающее ИЛИ – бинарная операция, результат которой равен 1, если только один из двух операндов равен 1 (в общем случае если во входном наборе операндов нечетное число единиц).

Для каждого бита результат выполнения операции будет получен в соответствии с таблицей 2.

Таблица 2 – Побитовые логические операции

| a | b | a & b | a b | !a | a ^ b |
|---|---|-------|-------|----|-------|
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

Пример: Мальчик купил в магазине n порций мороженого по цене 12 руб. и k плиток шоколада по цене 38 руб. Сколько всего потратил мальчик? Решение изображено на рисунке 1.

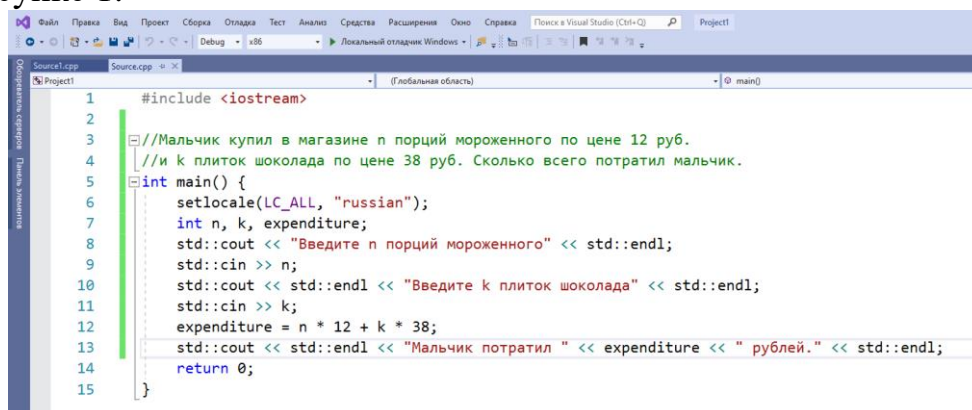


Рисунок 1 – Пример решения задачи

Математические функции C++

В C++ определены в заголовочном файле <cmath> функции, выполняющие некоторые часто используемые математические задачи. Например, нахождение корня, возведение в степень, sin(), cos() и многие другие. В таблице 3 показаны основные математические функций, прототипы которых содержатся в заголовочном файле <cmath>.

Таблица 3 – Математические функции в C++

| Функция | Описание | Пример |
|------------|--|---|
| abs(a) | модуль или абсолютное значение от a | abs(-3.0)= 3.0 abs(5.0)= 5.0 |
| sqrt(a) | корень квадратный из a, причём a не отрицательно | sqrt(9.0)=3.0 |
| pow(a, b) | возведение a в степень b | pow(2,3)=8 |
| ceil(a) | округление a до наименьшего целого, но не меньше чем a | ceil(2.3)=3.0 ceil(-2.3)=-2.0 |
| floor(a) | округление a до наибольшего целого, но не больше чем a | floor(12.4)=12 floor(-2.9)=-3 |
| fmod(a, b) | вычисление остатка от a/b | fmod(4.4, 7.5) = 4.4 fmod(7.5, 4.4) = 3.1 |

| | | |
|-----------------|--|----------------|
| exp(a) | вычисление экспоненты e^a | exp(0)=1 |
| sin(a) | a задаётся в радианах | |
| cos(a) | a задаётся в радианах | |
| log(a) | натуральный логарифм a(основанием является экспонента) | log(1.0)=0.0 |
| log10(a) | десятичный логарифм a | Log10(10)=1 |
| asin(a) | арксинус a, где $-1.0 < a < 1.0$ | asin(1)=1.5708 |

Пример:

```

1 #include <iostream>
2
3 using namespace std;
4 int main() {
5     cout << "log10(10) = " << log10(10.0) << endl; // логарифм десятичный
6     cout << "log10(1) = " << log10(1.0) << endl;
7     cout << "log(2.718281) = " << log(2.718281) << endl; // натуральный логарифм(по основанию экспоненты) exp = 2.718281
8     cout << "sqrt(9) = " << sqrt(9.0) << endl; // корень квадратный
9     cout << "pow(2,3) = " << pow(2.0, 3.0) << endl; // два в кубе
10    cout << "abs(0) = " << abs(0.0) << endl; // модуль от нуля
11    cout << "abs(-5) = " << abs(-5.0) << endl;
12    cout << "ceil(3.14) = " << ceil(3.14) << endl; // округление 3.14 до наименьшего целого, но не меньше чем 3.14
13    cout << "ceil(-2.4) = " << ceil(-2.4) << endl; // округление -2.4 до наименьшего целого, но не меньше чем -2.4
14    cout << "floor(3.14) = " << floor(3.14) << endl; // округление 3.14 до наибольшего целого, но не больше чем 3.14
15    cout << "floor(-2.4) = " << floor(-2.4) << endl; // округление -2.4 до наибольшего целого, но не больше чем -2.4
16    cout << "fmod(2.4/2.0) = " << fmod(2.4, 2.0) << endl; // остаток от деления 2.4/2
17    system("pause");
18    return 0;
19 }

```

Рисунок 2

Отладка приложения

Любой алгоритм выполняется последовательно, одна инструкция за другой. Точка остановки позволяет приостановить выполнение кода на определенной инструкции. Это необходимо, чтобы начать отладку с предположительно проблемного участка. Например, при модульной структуре проекта проблемы начались при подключении нового модуля. Незачем отлаживать весь проект, когда можно отладить только модуль. Точки остановки сильно облегчают процесс в этом случае. Для того, чтобы установить точку остановки необходимо кликнуть на небольшую область, выделенную как правило особым цветом, в левой части редактора кода. Также это можно сделать, установив каретку на нужную строку и нажав F9.

При установке точки остановки в окне редактора кода будет выведена иконка напротив строки кода.

Для того чтобы начать процесс отладки, необходимо поставить точку остановки в нужном месте и запустить модуль. Отладка станет доступна по достижению точки остановки. Также можно принудительно остановить выполнение модуля, в таком случае курсор автоматически переместится на код, который выполнялся до остановки. Но точка остановки при этом не установится.

Все кнопки в панели инструментов повторяют вкладку Debug, нас в данный момент интересует навигация по отлаживаемому коду.

Варианты работы пошаговой отладки

– шаг со входом. (F11) Если отладка остановилась на вызове функции, шаг со входом необходим для того, чтобы остановить выполнение модуля внутри тела функции. Т.е. если внутри функции нет точки остановки, можно продолжить пошаговое выполнение, выполнив шаг со входом. Обратите внимание, в данном примере шаг со входом остановит выполнение программы внутри функции get_message, а не hello, т.к. сначала выполняется она.

– шаг с обходом. (f10) Пошаговая отладка внутри функции. Не переходит внутрь функции при ее вызове, а выполняет как обычную инструкцию. При завершении функции останавливается на верхней в стеке вызовов. Вне зависимости от того, сколько функций вызывается на строчке, отладчик не прерывает внутри этих функций выполнение модуля.

– шаг с выходом. (Shift + f11) В этом случае отладчик остановит выполнение модуля после выхода из текущей функции, где происходит отладка.

3. Порядок выполнения работы

1. Изучить теоретические сведения к лабораторной работе.
2. Разработать на языке C++ программу вывода на экран решения задачи в соответствии с вариантом, указанным преподавателем.
3. Отлаженную, работающую программу сдать преподавателю. Работу программы показать с помощью самостоятельно разработанных тестов.
4. Ответить на контрольные вопросы.

Варианты индивидуальных заданий

Напишите код и нарисуйте графическое представление («блок-схемы») каждой вашей программы (по варианту):

Задание 1.

| Вариант | Создайте программу решения уравнения (x и y вводятся пользователем во время выполнения программы), учтите возможные условия арифметических операций |
|---------|---|
| 1 | $\sqrt{x - y} / 3x^2$ |
| 2 | $(x^2 - 4x + y) / 3$ |
| 3 | $\sqrt{3x + y} / 3x^2$ |
| 4 | $(x^2 - 4y + y) / 3$ |
| 5 | $\sqrt{ x - y } / 3x$ |
| 6 | $(x + 4y - 6) / 3$ |
| 7 | $\sqrt{10x - 5} / 3y$ |
| 8 | $(x^2 - 7x + y) / y$ |
| 9 | $(8x - 4y - 90) / x$ |
| 10 | $x + \frac{3y}{x^2}$ |
| 11 | $x^2 + \frac{3y}{x}$ |
| 12 | $(y^2 - 56)x \cdot y$ |
| 13 | $x \cdot y - y^2 + 78 / x$ |
| 14 | $x(y - 2) + \frac{3y}{x}$ |
| 15 | $y(y - 2) + \frac{3y}{x}$ |
| 16 | $x(x - 20) + \frac{3y}{x}$ |

| Вариант | Создайте программу решения уравнения (х и у вводятся пользователем во время выполнения программы), учтите возможные условия арифметических операций |
|---------|---|
| 17 | $x(y-2) + \frac{2y}{x}$ |
| 18 | $5x(y-2) + \frac{y}{x}$ |
| 19 | $\sqrt{ 2x-y }/3x$ |
| 20 | $\sqrt{ 6x-y }/3x$ |
| 21 | $\sqrt{ x-5y }/3x$ |
| 22 | $13(4y-2) + \frac{y}{x}$ |
| 23 | $5(x-2) + \frac{y}{x}$ |
| 24 | $5(y-2) + \frac{4y}{x}$ |
| 25 | $\sqrt{x-y}/3x^2$ |
| 26 | $(x^2 - 4x + y)/3$ |
| 27 | $\sqrt{3x+y}/3x^2$ |
| 28 | $(x^2 - 4y + y)/3$ |
| 29 | $\sqrt{ x-y }/3x$ |
| 30 | $(x + 4y - 6)/3$ |

Задание 2.

| Вариант | Создайте программу решения уравнения вида $x^2 - a \cdot x + b - c$, где |
|---------|---|
| 1 | A=5, b=7, c=7 – Это константы, х вводит пользователь с клавиатуры |
| 2 | A=5, b=7 – Это константы, с и х вводит пользователь с клавиатуры |
| 3 | A=5, c=7 – Это константы, b и х вводит пользователь с клавиатуры |
| 4 | b=7, c=7 – Это константы, а и х вводит пользователь с клавиатуры |
| 5 | A=15, b=7, c=27 – Это константы, х вводит пользователь с клавиатуры |
| 6 | A=3, b=6 – Это константы, с и х вводит пользователь с клавиатуры |
| 7 | A=1, c=2 – Это константы, b и х вводит пользователь с клавиатуры |
| 8 | b=3, c=3 – Это константы, а и х вводит пользователь с клавиатуры |
| 9 | A=11, b=22, c=33 – Это константы, х вводит пользователь с клавиатуры |
| 10 | A=7, b=8 – Это константы, с и х вводит пользователь с клавиатуры |
| 11 | A=8, c=6 – Это константы, b и х вводит пользователь с клавиатуры |
| 12 | b=4, c=4 – Это константы, а и х вводит пользователь с клавиатуры |
| 13 | A=2, b=2, c=4 – Это константы, х вводит пользователь с клавиатуры |
| 14 | A=4, b=3 – Это константы, с и х вводит пользователь с клавиатуры |
| 15 | A=2, c=2 – Это константы, b и х вводит пользователь с клавиатуры |
| 16 | b=1, c=4 – Это константы, а и х вводит пользователь с клавиатуры |
| 17 | A=8, b=8, c=8 – Это константы, х вводит пользователь с клавиатуры |
| 18 | A=8, b=1 – Это константы, с и х вводит пользователь с клавиатуры |
| 19 | A=2, c=0 – Это константы, b и х вводит пользователь с клавиатуры |
| 20 | b=13, c=13 – Это константы, а и х вводит пользователь с клавиатуры |

| Вариант | Создайте программу решения уравнения вида $x^2 - a \cdot x + b - c$, где |
|---------|---|
| 21 | A=78, b=2, c=91 – Это константы, x вводит пользователь с клавиатуры |
| 22 | A=19, b=3 – Это константы, c и x вводит пользователь с клавиатуры |
| 23 | A=3, c=7 – Это константы, b и x вводит пользователь с клавиатуры |
| 24 | b=21, c=9 – Это константы, a и x вводит пользователь с клавиатуры |
| 25 | A=3, b=6 – Это константы, c и x вводит пользователь с клавиатуры |
| 26 | A=1, c=2 – Это константы, b и x вводит пользователь с клавиатуры |
| 27 | b=3, c=3 – Это константы, a и x вводит пользователь с клавиатуры |
| 28 | A=11, b=22, c=33 – Это константы, x вводит пользователь с клавиатуры |
| 29 | A=7, b=8 – Это константы, c и x вводит пользователь с клавиатуры |
| 30 | A=3, b=6 – Это константы, c и x вводит пользователь с клавиатуры |

4. Контрольные вопросы

1. Как оформляется оператор вывода на экран?
2. Что можно указывать в качестве элементов списка вывода? Какой символ используется для разделения элементов списка вывода? Какой символ применяется для разделения целой и дробной частей вещественного числа?
3. Что будет выведено на экран, если в списке вывода записано:
 - а) число?
 - б) имя величины?
 - в) текст в кавычках?
 - г) арифметическое выражение?
4. Как должен быть оформлен оператор вывода, чтобы информация выводилась на экран с новой строки?
5. Как оформляется оператор ввода? Что можно указывать в качестве элементов списка ввода? Как работает оператор ввода (что происходит при его выполнении)?
6. Почему перед оператором ввода в программе целесообразно записывать оператор вывода?

Литература

Дейтел, Х. Как программировать на C++ / Х.Дейтел, П.Дейтел. – 1037 с.

Страуструп, Б. Язык программирования C++ / Б. Страуструп. – СПб. : БИНОМ, 2011.

Павловская, Т. А. C++. Объектно-ориентированное программирование : практикум / Павловская, Т. А., Щупак. – СПб. : Питер, 2011.

Преподаватель

Белокопыцкая Ю.А.

Рассмотрено на заседании

цикловой комиссии ПОИТ № 10

Протокол № _____ от «____» _____ 201__

Председатель ЦК

С.В.Банцевич