

## ЛАБОРАТОРНАЯ РАБОТА № 2

### ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ

**Цель работы:** формирование знаний и умений по работе с интегрированной средой разработки программ на языке программирования Си, приобретение практических навыков работы с редактором, контекстной помощью.

#### Краткие теоретические сведения

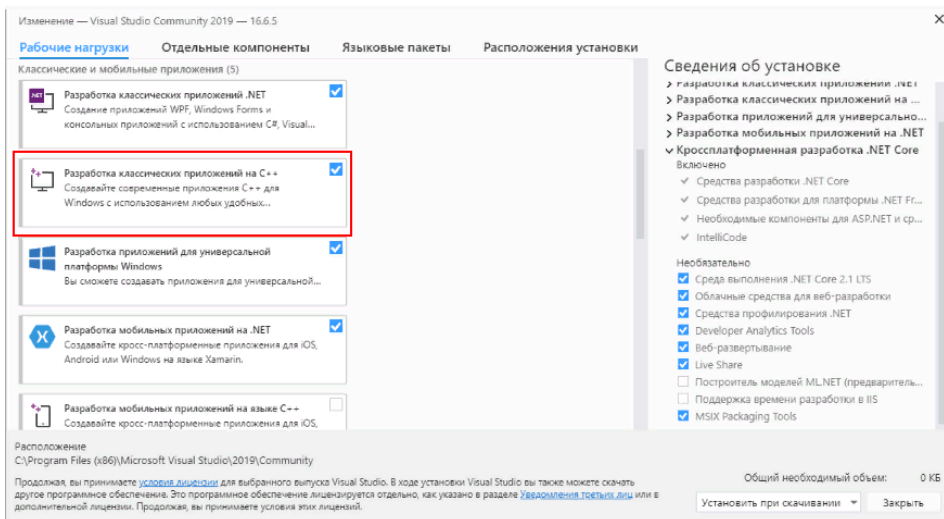
#### *Основные компоненты интегрированной среды Borland C*

##### **Первая программа в Visual Studio**

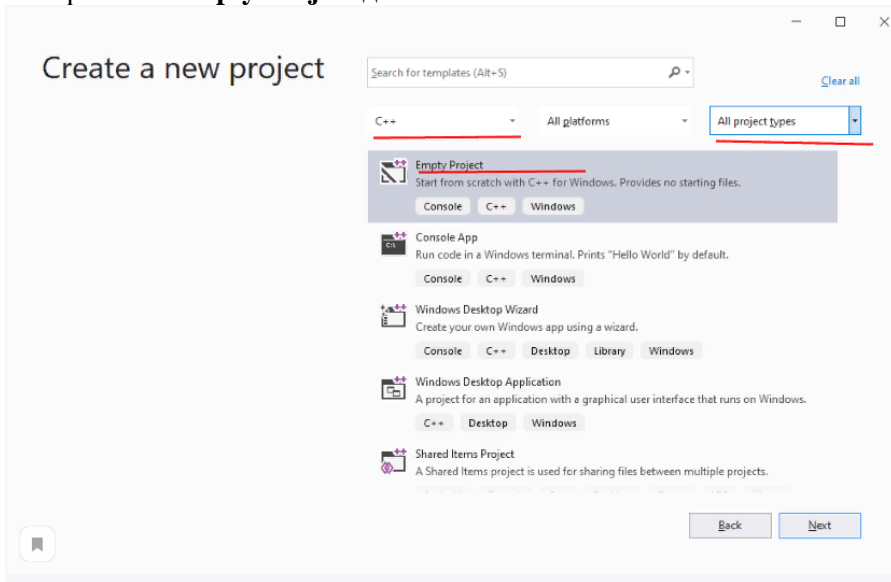
Мы можем по отдельности использовать текстовый редактор и компилятор, вручную компилировать и запускать программу в консоли или терминале, однако более удобный способ представляет использование различных сред разработки или IDE. Они, как правило, содержат встроенный текстовый редактор, имеет связь с компилятором, позволяя скомпилировать и запустить программу по одному клику мыши, а также еще множество разных вспомогательных возможностей.

В данном случае для программирования под Windows в качестве среды разработки мы будем использовать бесплатную и полнофункциональную среду Visual Studio 2019 Community, которую можно найти по адресу <https://www.visualstudio.com/ru/vs/>.

После загрузки и запуска установщика Visual Studio в нем необходимо отметить пункт "Разработка классических приложений на C++":



Выбрав все необходимые пункты, нажмем ОК для запуска установки. После установки Visual Studio создадим первый проект. Для этого откроем Visual Studio. На стартовом экране выберем тип **Empty Project** для языка C++:




На следующем экране в поле для имени проекта дадим проекту имя HelloApp и также можно указать расположение проекта. И затем нажмем на Create для создания проекта.

## Configure your new project

Empty Project   Console   C++   Windows

Project name

Location

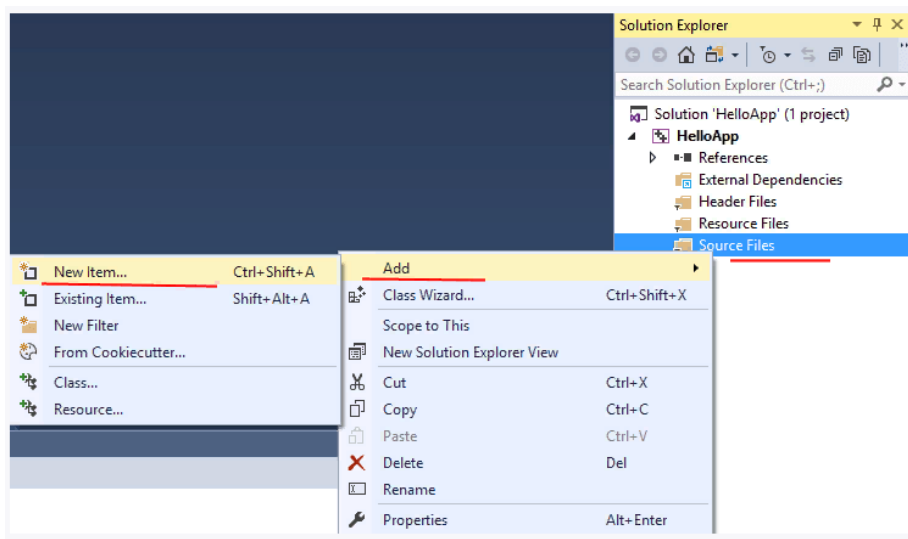
Solution name ⓘ

☐ Place solution and project in the same directory

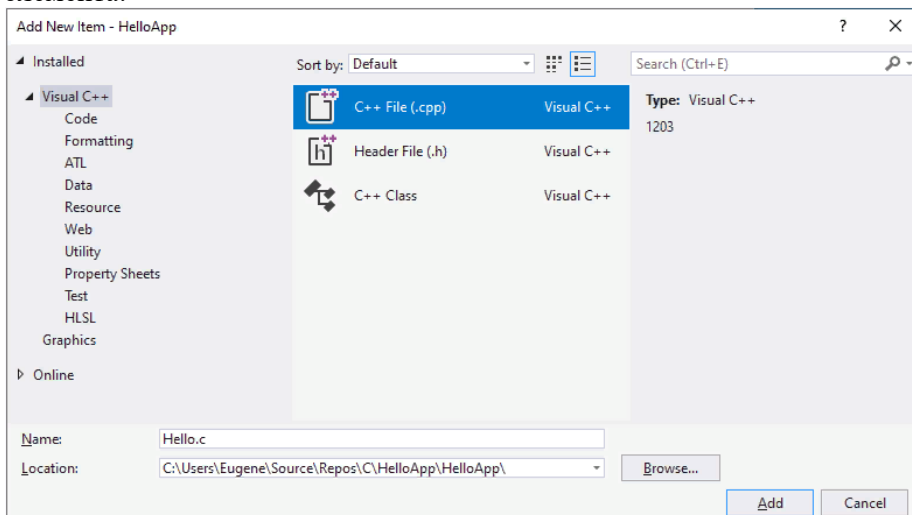
Back   Create

Если в VS уже открыт какой-нибудь проект, то можно создать новый проект для C через меню **File (Файл) -> New (Создать) -> Project... (Проект)** и дальше повторить те же действия.

После этого Visual Studio создаст пустой проект. Добавим в него текстовый файл для набора исходного кода. Для этого в окне Solution Explorer (Обозреватель решений) нажмем правой кнопкой мыши на узел **Source Files** и в контекстном меню выберем **Add -> New Item...**:



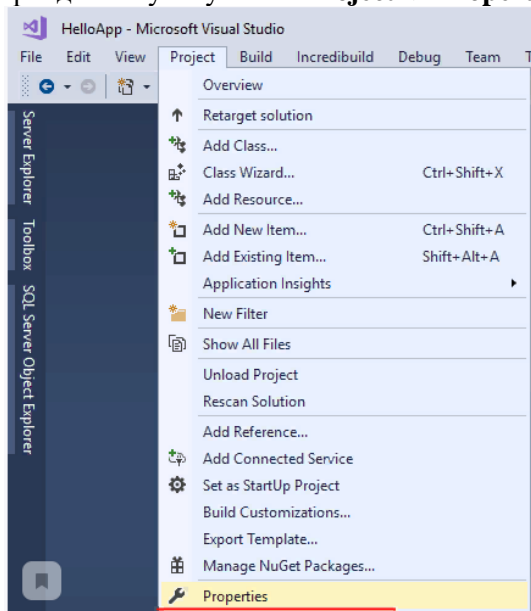
Затем нам откроется окно для добавления нового элемента:



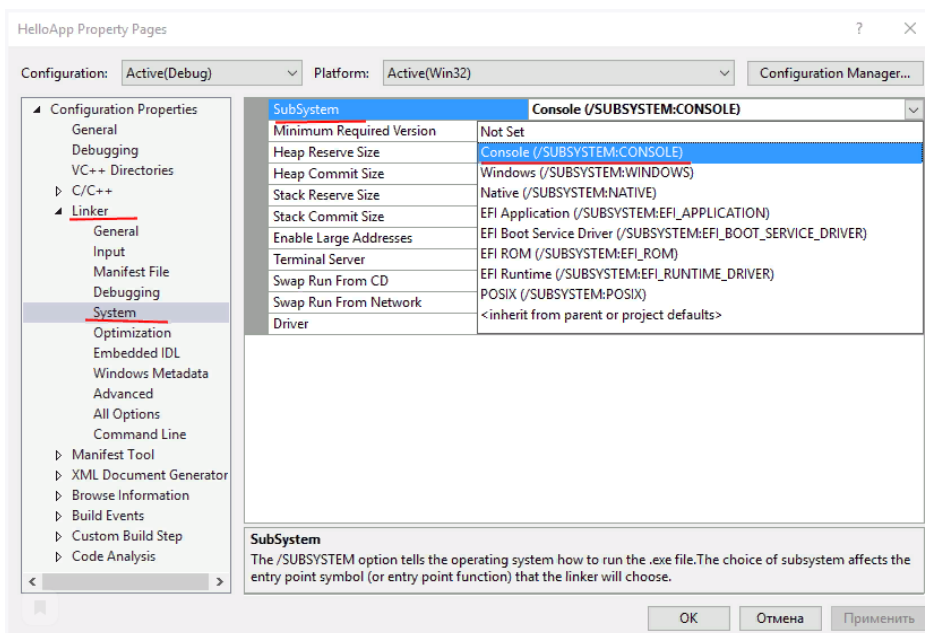
Здесь нам надо выбрать пункт **C++ File(.cpp)**, а внизу окна укажем для файла имя **Hello.c**. Как правило, исходные файлы на Си имеют расширение **.c**. Оно указывает, что этот файл содержит

исходный код на языке C, и он будет обрабатываться соответствующим компилятором.

После добавления файла изменим опции проекта. Для этого перейдем к пункту меню **Project -> Properties**

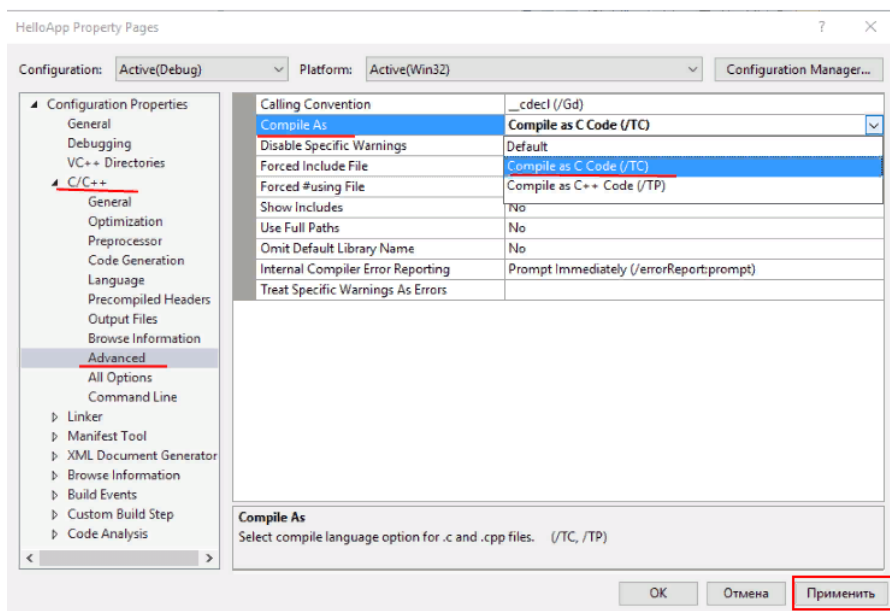


Сначала в свойствах проекта установим, что это будет консольное приложение. Для этого перейдем к пункту **Linker ->System** и далее для поля **SubSystem** установим значение **Console(/SUBSYSTEM:CONSOLE)**, выбрав нужный элемент в списке:



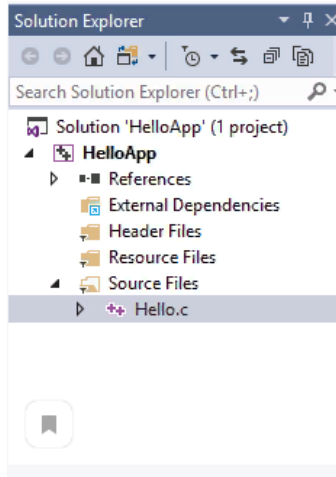
После установки этого значения нажмем на кнопку "Применить", чтобы новые настройки конфигурации вступили в силу.

Также в окне свойств проекта в левой части перейдем к секции C/C++ и далее к пункту Advanced:



В правой части окна для поля **Compile As** установим значение **Compile as C Code (/TC)**. Тем самым мы говорим, чтобы по умолчанию исходный код компилировался именно как код C, а не C++. После установки этой опции нажмем на кнопку "Применить".

После добавления файла проект будет иметь следующую структуру:



Вкратце пробежимся по этой структуре. Окно Solution Explorer содержит решение. В данном случае оно называется HelloApp. Решение может содержать несколько проектов. По умолчанию у нас один проект, который имеет то же имя - HelloApp. В проекте есть ряд узлов:

- **External Dependencies:** отображает файлы, которые используются в файлах исходного кода, но не являются частью проекта
- **Header Files:** предназначена для хранения заголовочных файлов с расширением .h
- **Resource Files:** предназначена для хранения файлов ресурсов, например, изображений
- **Source Files:** хранит файлы с исходным кодом

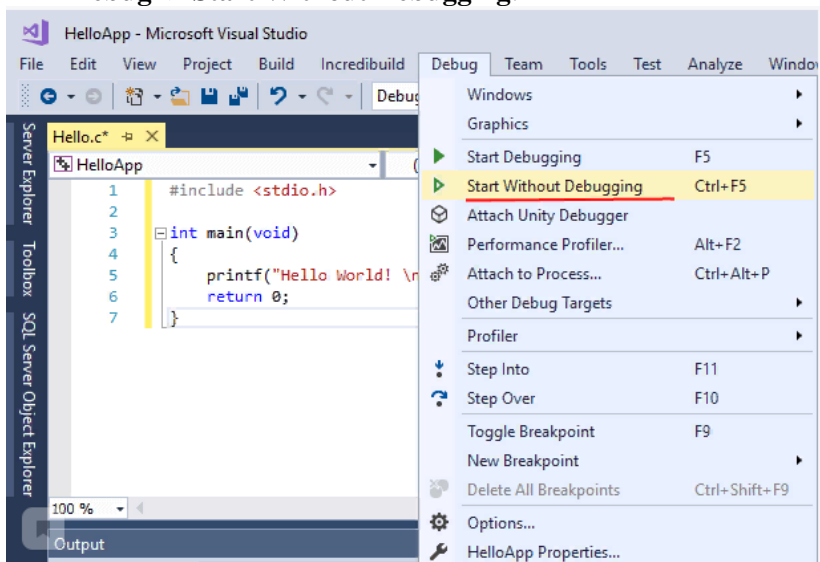
Теперь определим в файле Hello.c простейший код, который будет выводить строку на консоль:

```
1 #include <stdio.h>           // подключаем заголовочный файл stdio.h
2 int main(void)               // определяем функцию main
3 {                             // начало функции
4     printf("Hello world! \n"); // выводим строку на консоль
5     return 0;                // выходим из функции
6 }
```

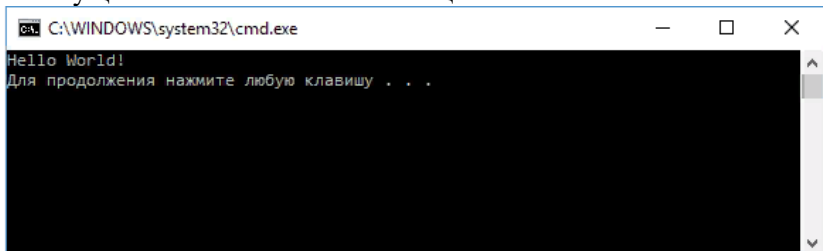
Здесь использован весь тот код, который был рассмотрен в предыдущих темах про компиляцию с помощью GCC.



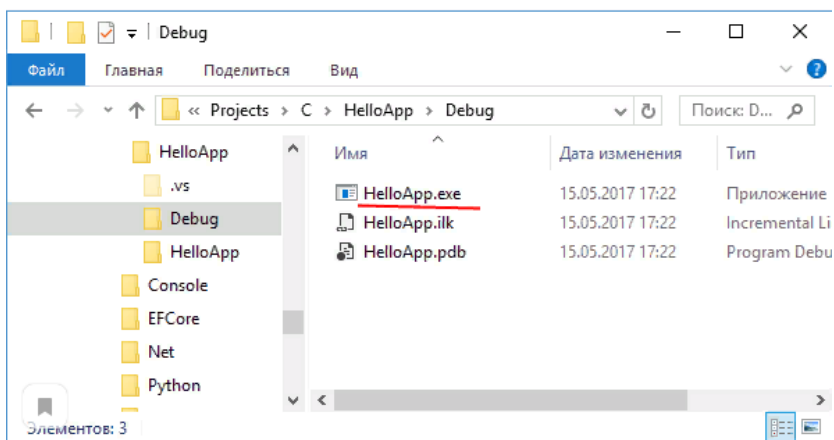
Теперь запустим программу. Для этого в Visual Studio нажмем на сочетание клавиш **Ctrl+F5** или выберем пункт меню **Debug -> Start Without Debugging**:



И в итоге Visual Studio передаст исходный код компилятору, который скомпилирует из кода исполняемый файл `exe`, который потом будет запущен на выполнение. И мы увидим на запущенной консоли наше сообщение:



Затем в папке **Debug** в проекте мы можем увидеть скомпилированный файл `exe`, который мы можем запускать независимо от Visual Studio:



В данном случае файл **HelloApp.exe** как раз и представляет скомпилированный исполняемый файл. Кроме этого файла в той же папке автоматически генерируются два вспомогательных файла:

- **HelloApp.ilink**: файл "incremental linker", который используется компоновщиком для ускорения компоновки
- **HelloApp.pdb**: файл, который содержит отладочную информацию

### Локализация и кириллица в консоли

При использовании кириллических символов мы можем столкнуться с ситуацией, когда вместо кириллических символов отображаются непонятные знаки. Особенно это актуально для ОС Windows. И в этом случае необходимо явным образом задать текущую локаль (культуру) для вывода символов. В Си это делается с помощью функции `setlocale()`, определение которой имеется в заголовочном файле **locale.h**.

Итак, изменим код, который использовался в прошлых темах следующим образом:

```

1  #include <stdio.h>
2
3  int main(void)
4  {
5      printf("Привет мир! \n");
6      return 0;
7  }

```

Компиляция и запуск в ОС Windows может выглядеть следующим образом:

The screenshot shows a Windows Command Prompt window titled "Администратор: Командная строка". The text inside the window is as follows:

```

Microsoft Windows [Version 10.0.15063]
(c) Корпорация Майкрософт (Microsoft Corporation), 2017. Все права защищены.

C:\WINDOWS\system32>cd C:\c

C:\c>gcc hello.c

C:\c>a.exe
Привет мир!
C:\c>

```

Вместо русских слов я получаю непонятные символы, и это не то, что ожидалось. Теперь изменим код, применив функцию `setlocale`:

```

1  #include <stdio.h>
2  #include <locale.h>
3
4  int main(void)
5  {
6      char *locale = setlocale(LC_ALL, "");
7
8      printf("Привет мир! \n");
9      return 0;
10 }

```

Поскольку функция `setlocale` определена в файле `locale.h`, то он подключается с помощью директивы `#include <locale.h>`.

Повторно компилируем и запустим приложение:

Стоит отметить, что в качестве кодировки текстового файла в этом случае должна использоваться кодировка ANSI или Windows-1251, но не UTF-8.

На некоторых платформах, например, Ubuntu 16.04, мы можем не столкнуться с подобной проблемой. И в этом случае вызов функции `setlocale` просто не окажет никакого влияния.

### **Порядок выполнения работы**

1. Изучить теоретические сведения по теме «Интегрированная среда разработки».
2. Ответить на контрольные вопросы.
3. Выполнить задание.

### **Контрольные вопросы**

1. Как осуществляется загрузка интегрированной среды?
2. С помощью какого пункта меню можно создать новый файл?
3. Какое сочетание клавиш позволяет выполнить программу?

### **Задание для выполнения**

Написать программу «Hello world» всеми способами, описанными в примере