Компиляция в командной строке с использованием утилиты make

Здесь я покажу минимальный набор действий для успешной компиляции программы с использованием командной строки и утилиты make. Этот текст написан в качестве примера при обсуждении поста http://www.cyberforum.ru/cpp-beginners/thread1368779.html на сайте http://www.cyberforum.ru.

Используемая операционная система — Windows 8.1, но все, что здесь представлено, работает и в других версиях. Еще раз повторю, что здесь показан минимум, который демонстрирует суть работы с компилятором и утилитой make в командной строке. В качестве подопытного компилятора был выбран компилятор TDM-GCC-64. Это компилятор из семейства GNU GCC, адаптированный под Windows (по сути, это MinGW, но более свежие сборки). Скачать его можно по ссылке: http://tdm-gcc.tdragon.net/download. Если Вы скачаете установочную программу, то после его установки, прописывать пути к нему в переменную окружения не нужно (это делается автоматически). Здесь я просто покажу последовательность действий для этого.

В моей системе, компилятор был установлен на диск C в папку TDM-GCC-64. На рисунке 1 представлен путь к бинарным файлам компилятора.

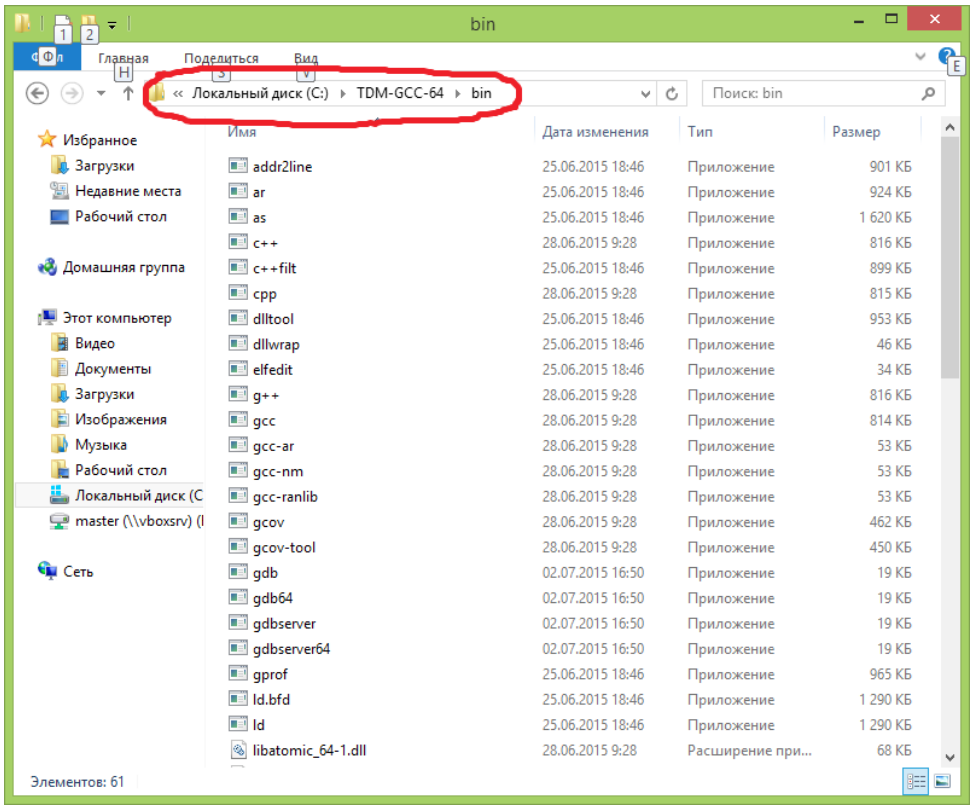

Рисунок 1 — Путь к бинарным файлам компилятора и утилитам.

Именно этот путь и должен быть прописан в системную переменную PATH. В этом случае, при загрузке командного интерпретатора, автоматически будет обеспечиваться доступ к любым файлам, находящимся в этой папке из любой точки рабочего пространства (из любой директории).

Напомню, что при установки компилятора TDM-GCC при помощи инсталлятора, путь в переменную PATH прописывается автоматически, но желательно проследить в процессе установки, чтобы была поставлена соответствующая «галочка» в меню выбора. Если Вы производите установку компилятора другим способом, то делать запись в системную переменную придется самостоятельно. Для этого потребуется выполнить несколько шагов. Далее я буду показывать процесс изменения переменной окружения только одним из

способов (используя графический интерфейс). Способ изменения переменной окружения с помощью командной строки здесь не рассматривается.

Итак, первый шаг заключается в вызове панели управления (рисунок 2).

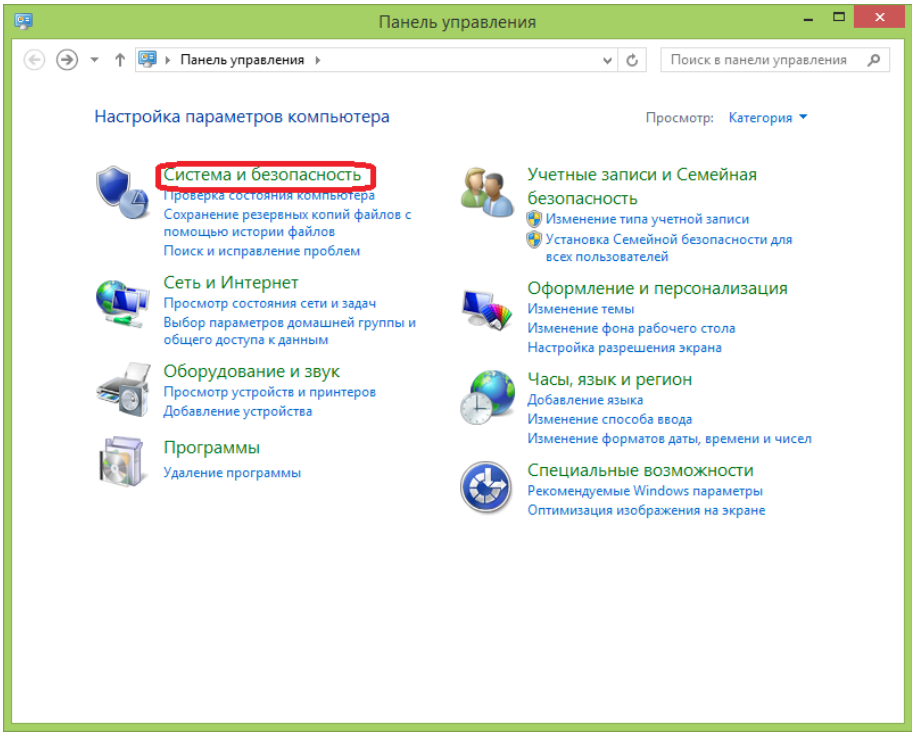

Рисунок 2 — Панель управления.

Далее, выбираем пункт «Система и безопасность» (выделено красным цветом) и попадаем в соответствующее окно (рисунок 3).

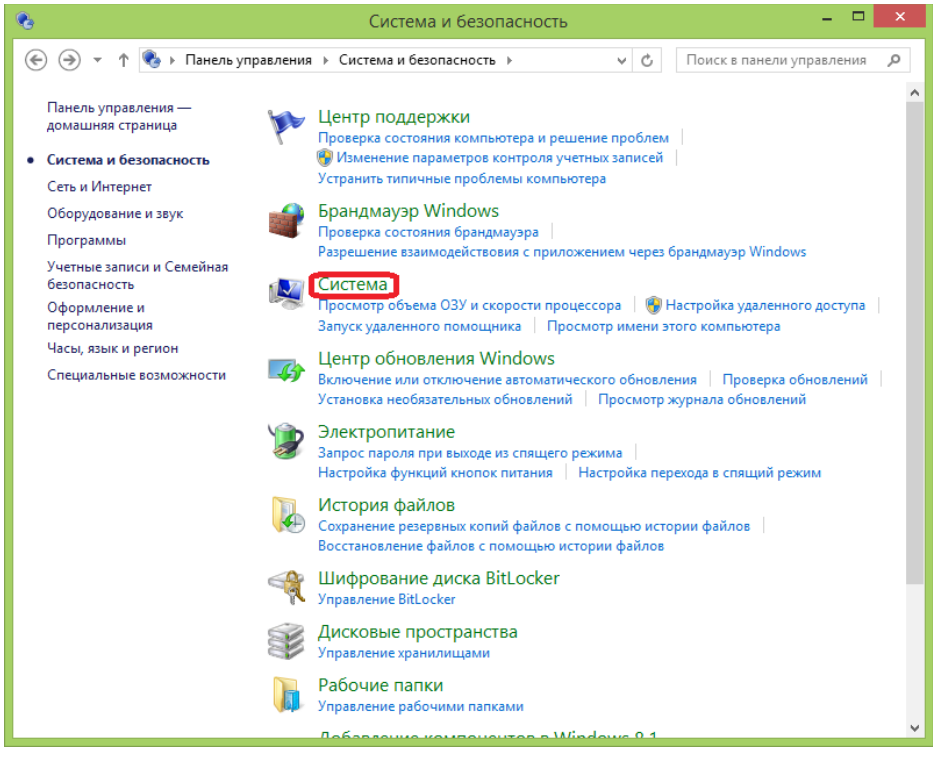

Рисунок 3 — Окно «Система и безопасность»

Следующий шаг заключается в выборе пункта меню «Система», после чего в одноименном окне нужно выбрать пункт «Дополнительные параметры системы» (рисунок 4).

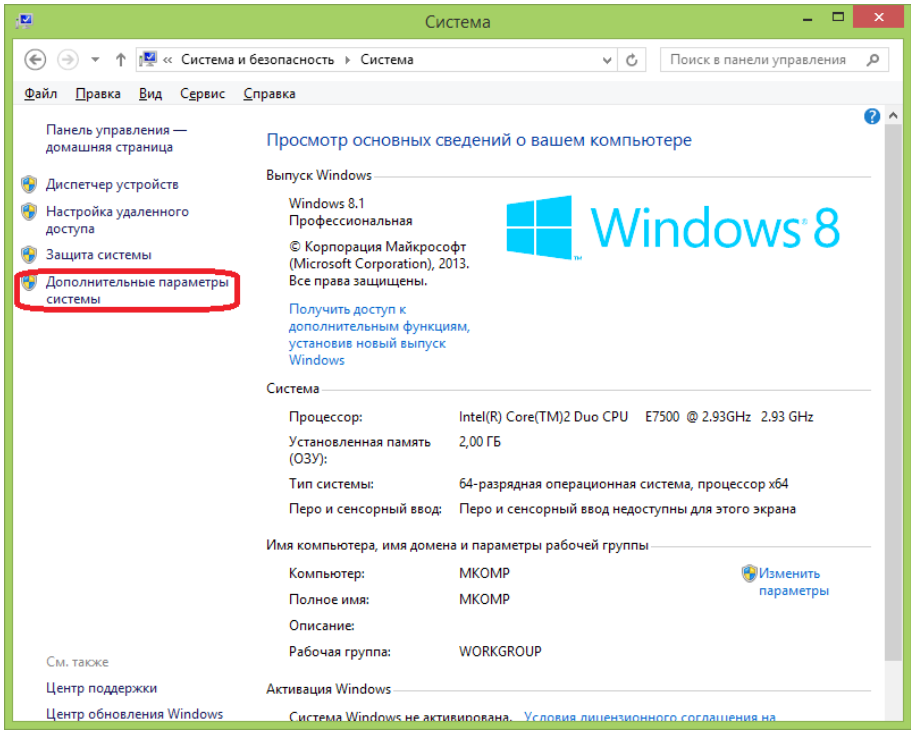

Рисунок 4 — Выбор пункта «Дополнительные параметры системы»

В открывшемся окне «Свойства системы» на вкладке «Дополнительно» нужно нажать кнопку «Переменные среды» (рисунок 5).

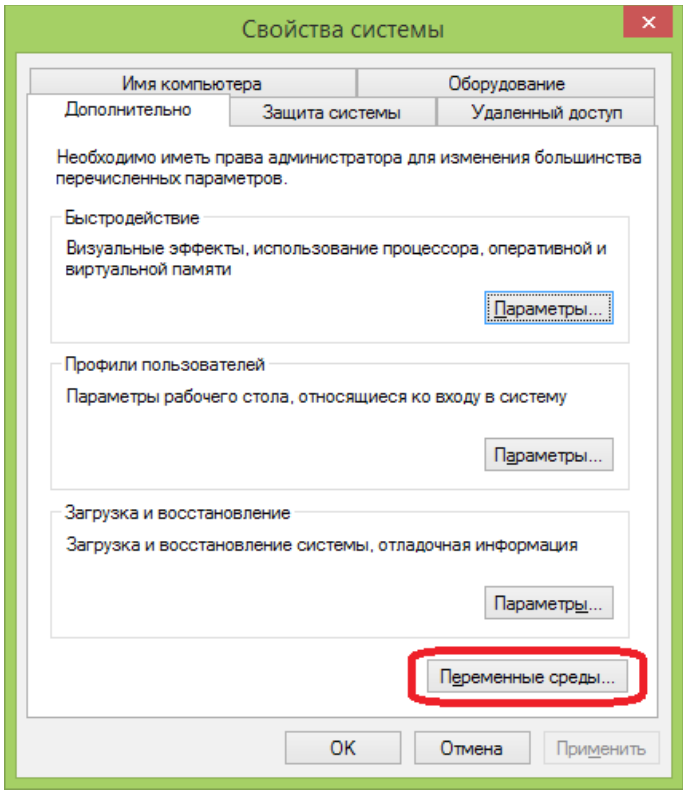

Рисунок 5 — Доступ к переменным среды.

В результате этих действий появится окно, содержащее два списка переменных: переменные среды для конкретного пользователя и системные переменные (рисунок 6).

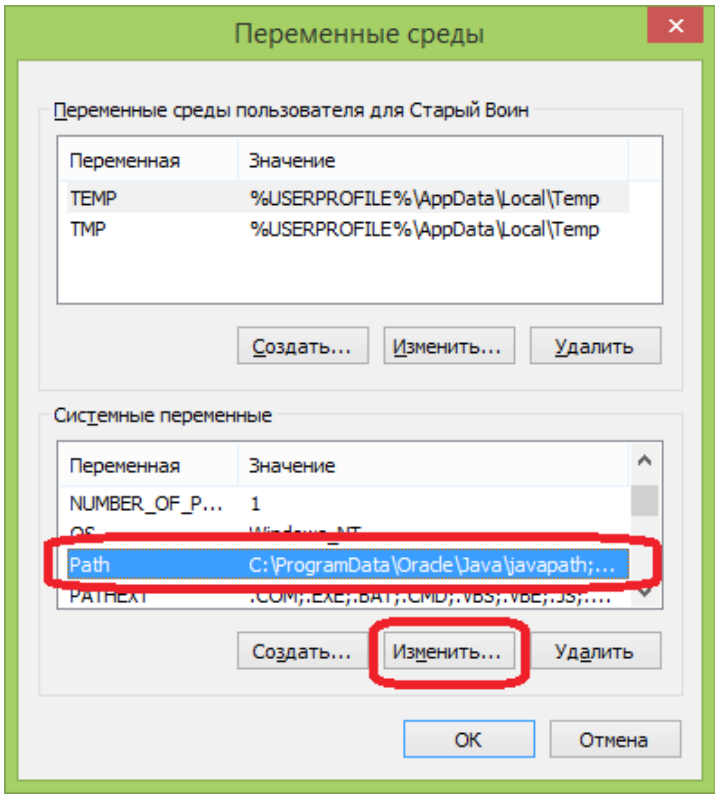

Рисунок 6 — Переменные среды.

Различия заключаются в том, что если использовать переменные среды пользователя, то все внесенные туда изменения будут доступны только для одного (конкретного) пользователя. Если Вы хотите дать возможность работать с компилятором всем пользователям данной системы, то необходимо вносить изменения в системные переменные. Допустим, что Вы решили предоставить возможность работать с компилятором всем пользователям. В таком случае выбираем системную переменную Path и нажимаем кнопку «Изменить...». Появляется диалоговое окно «Изменение системной переменной» (рисунок 7).

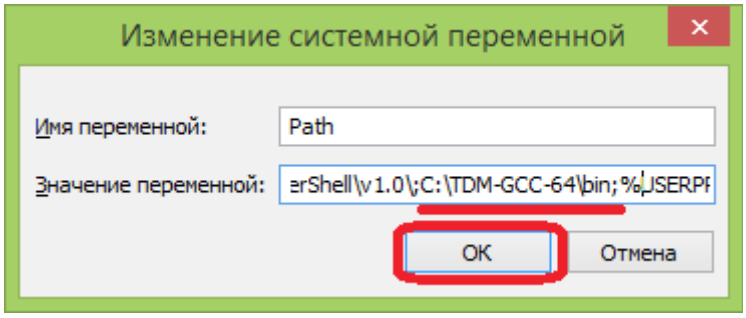

Рисунок 7 — Изменение системной переменной.

Так как в моей системе путь к компилятору был прописан автоматически при установке, то на рисунке видна соответствующая запись (подчеркнуто красным). В случае, когда запись необходимо внести вручную, следует в конце строки поставить точку с запятой (это разделитель), а затем прописать (или вставить из буфера обмена) путь к папке с компилятором. Точку с запятой в конце ставить не обязательно (главное, чтобы она отделяла внесенную запись от предыдущей.

Далее, нажимаем кнопку «ОК» во всех открытых окнах и можем проверить результат в командной строке. Для этого вызываем командный интерпретатор «cmd» и выполняем команду: g++ --version (проверяем версию компилятора, хотя можем просто выполнить g++ и компилятор выдаст ошибку, что не назначен файл с исходным кодом). Результат проверки представлен на рисунке 8.

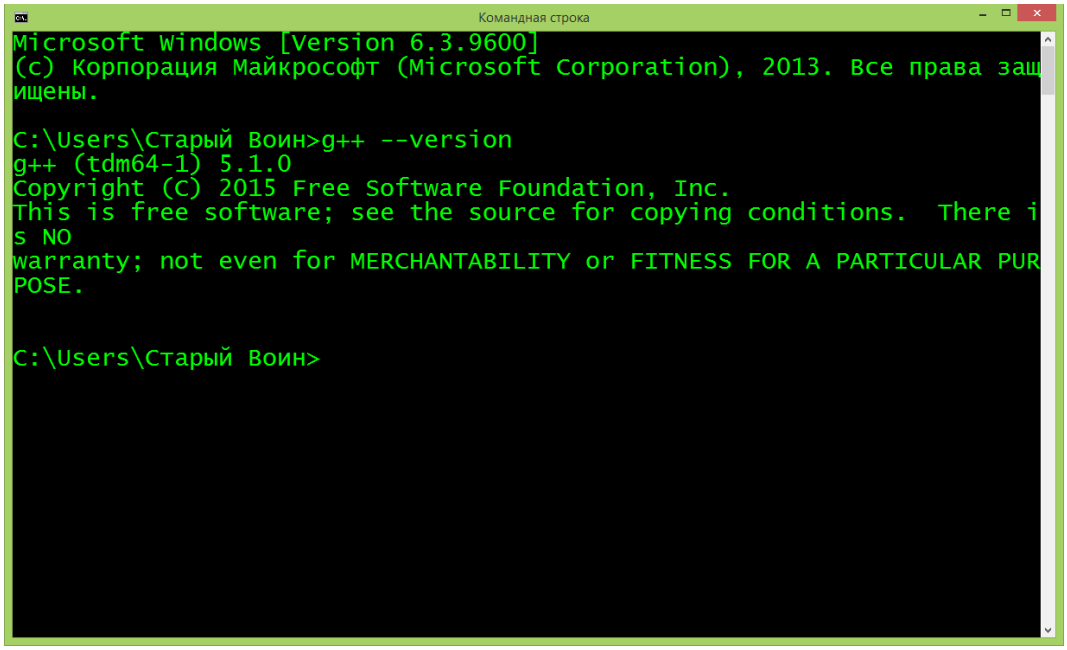

Рисунок 8 — Запуск компилятора в командной строке.

Далее, в любом текстовом редакторе создаем минимальную программу для проверки работы компилятора и утилиты make. Я для этого использовал Notepad++ (рисунок 9).

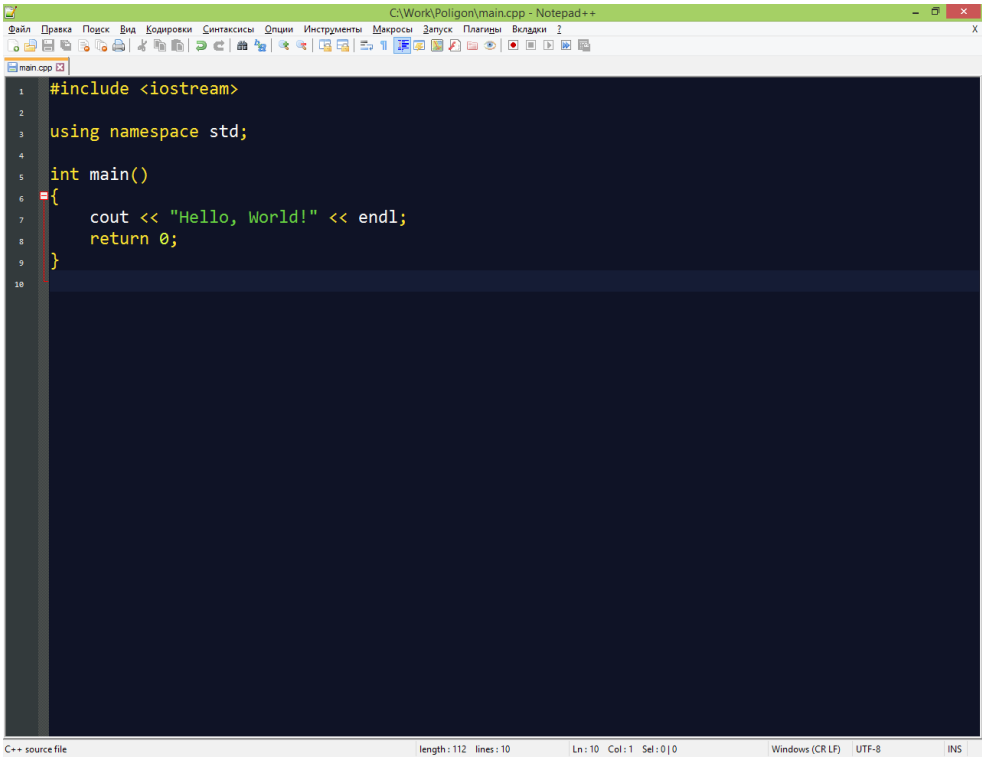

Рисунок 9 — Традиционный «Hello, World!».

Я сохранил файл программы в папку Poligon (мы же тут испытания проводим…), расположенную в C:\Work. Таким образом, путь к исходному файлу получился следующим: «C:\Work|Poligon». Переходим в командной строке в эту папку (рисунок 10).

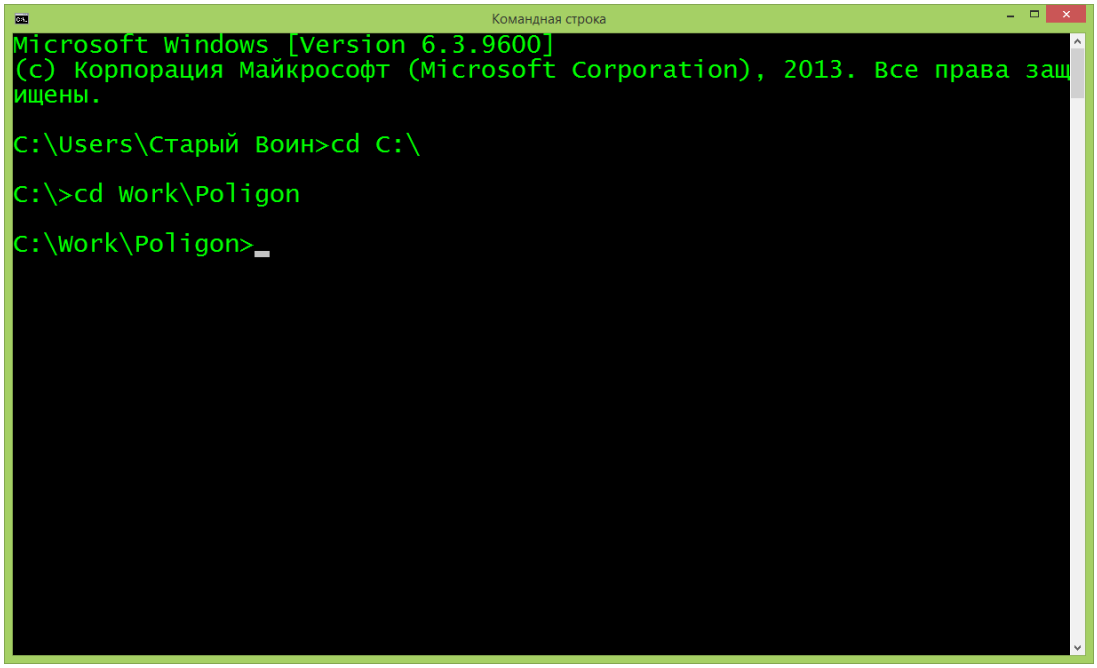

Рисунок 10 — Переход в папку с исходным файлом.

Это даст возможность не указывать компилятору путь к файлу (упрощаем себе работу по набору команды). Затем компилируем и выполняем программу (рисунок 11).

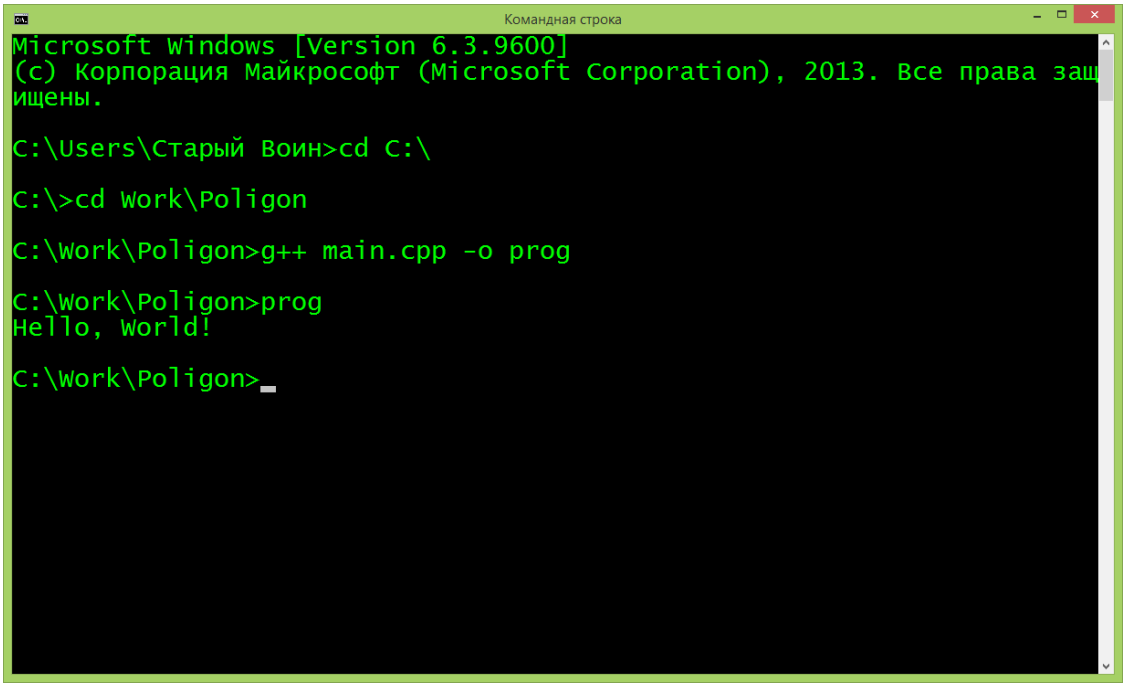

Рисунок 11 — Компиляция и выполнение программы.

Следующий этап — создание файла для утилиты make. Напомню, что утилита make ищет в текущем каталоге файл с именем «makefile» (без всяких расширений). При сохранении первый раз созданного файла контролируйте, чтобы текстовый редактор не

добавил расширение «.txt» автоматически. На самом деле, вместо имени «makefile» можно использовать любое (в том числе и с расширением), только в командной строке при вызове утилиты нужно использовать ключ «-f». В нашем случае поступим как проще. Еще одна особенность работы утилиты «make». Иногда она требует файл с прописной первой буквой «Makefile». Точно не разобрался в этих тонкостях, но могу сказать, что такое случалось в операционной системе Linux при попытке компиляции модуля ядра. В Windows вполне себе работает вариант «makefile».

На рисунке 12 представлен простейший makefile. К сожалению, цветовая схема, используемая мною в Notepad++ отобразила имя цели темным цветом, поэтому ниже продублирую содержимое файла.

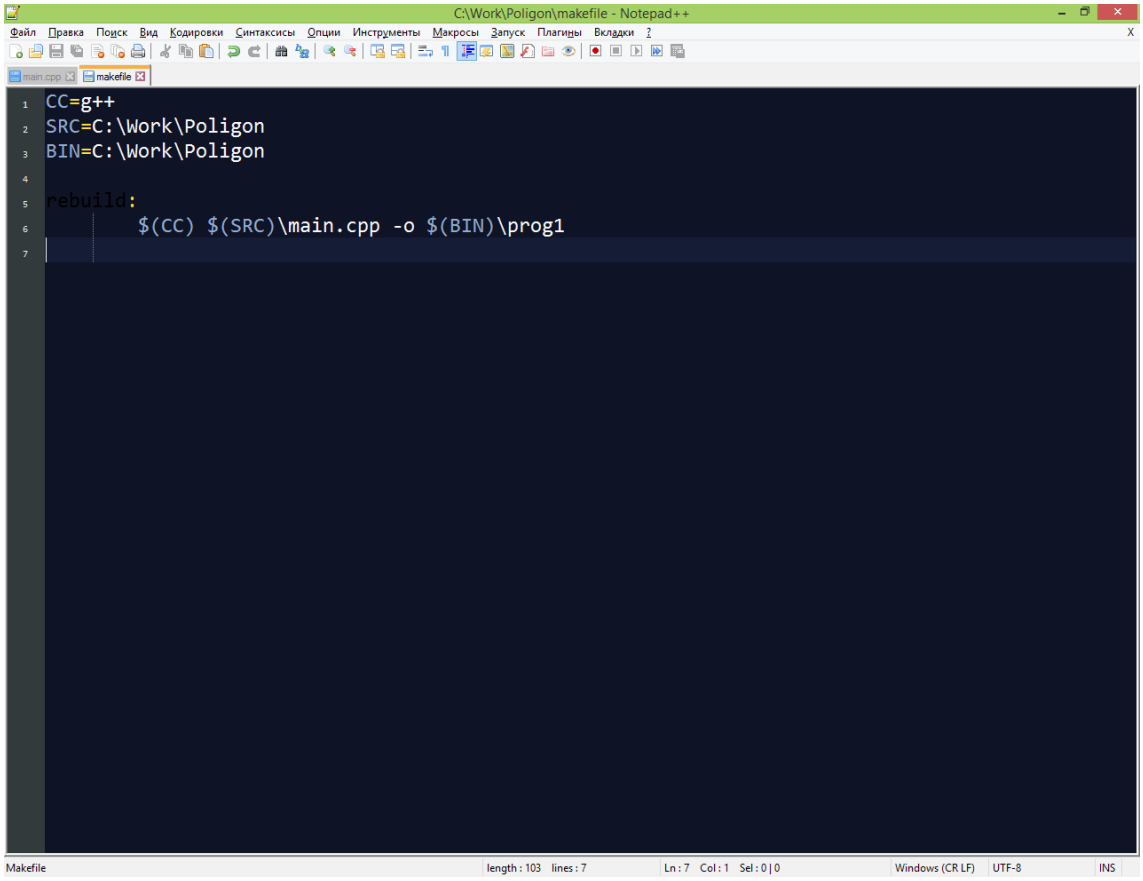

Рисунок 12 — Содержимое файла makefile.

```
CC=g++
SRC=C:\Work\Poligon
BIN=C:\Work\Poligon

rebuild:
	$(CC) $(SRC)\main.cpp -o $(BIN)\prog1
```

Несколько слов по поводу составления makefile. В этом файле указываются цели, которые необходимо выполнить утилите. У нас сейчас указана одна цель — rebuild. После имени цели ставиться двоеточие и могут указываться файлы, необходимые для выполнения этой цели. В нашем случае цель очень простая. Она заключается в создании компилятором исполняемого файла с именем «prog1.exe».

Кроме того, здесь показан способ передачи параметров для созданной цели через переменные. Первые три строки представляют собой переменные «CC», «SRC» и «BIN», в которых указывается используемый компилятор и путь к исходному файлу и папке, куда

должен попасть исполняемый файл. Использование переменных сокращает время на запись файла (иначе пришлось бы писать все пути вручную) и позволяет при необходимости достаточно просто вносить изменения.

Еще одной особенностью записи в файл является то, что строка цели должна содержать в начале табуляцию (кажется не менее 8 символов, но могу и ошибаться). Несколько слов про вызов утилиты «make». Если Вы попробуете вызвать эту утилиту командой «make» (этот способ является традиционным в операционной системе Linux), то результатом выполнения команды будет ошибка (рисунок 13).

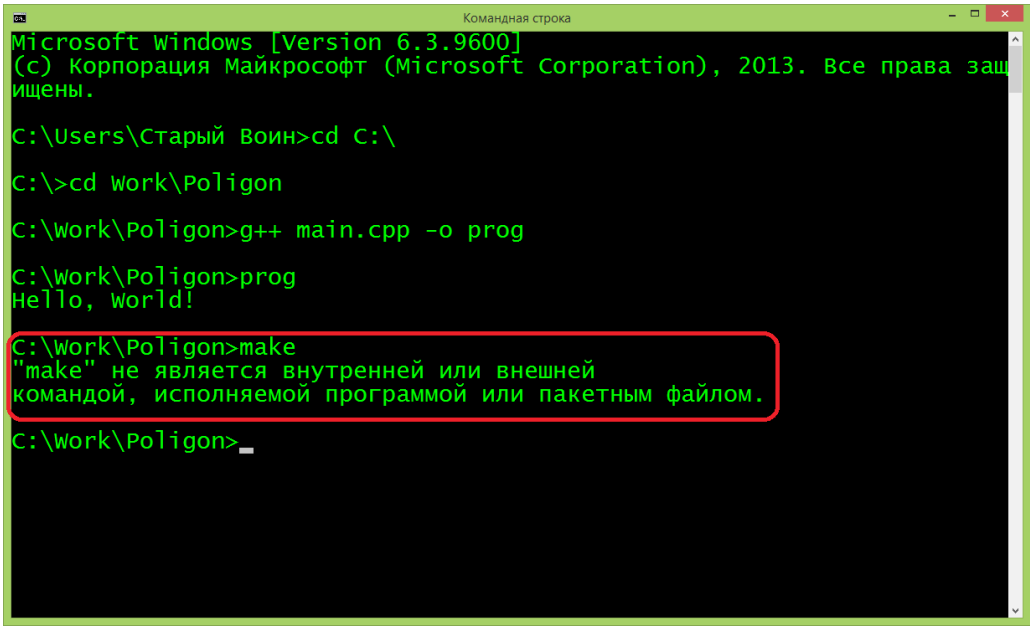

Рисунок 13 — Ошибка вызова утилиты «make».

Это происходит потому, что в компиляторе используется другое имя для этой утилиты. Имя файла утилиты — mingw32-make.exe. Использование этого имени даст положительный результат (рисунок 14).

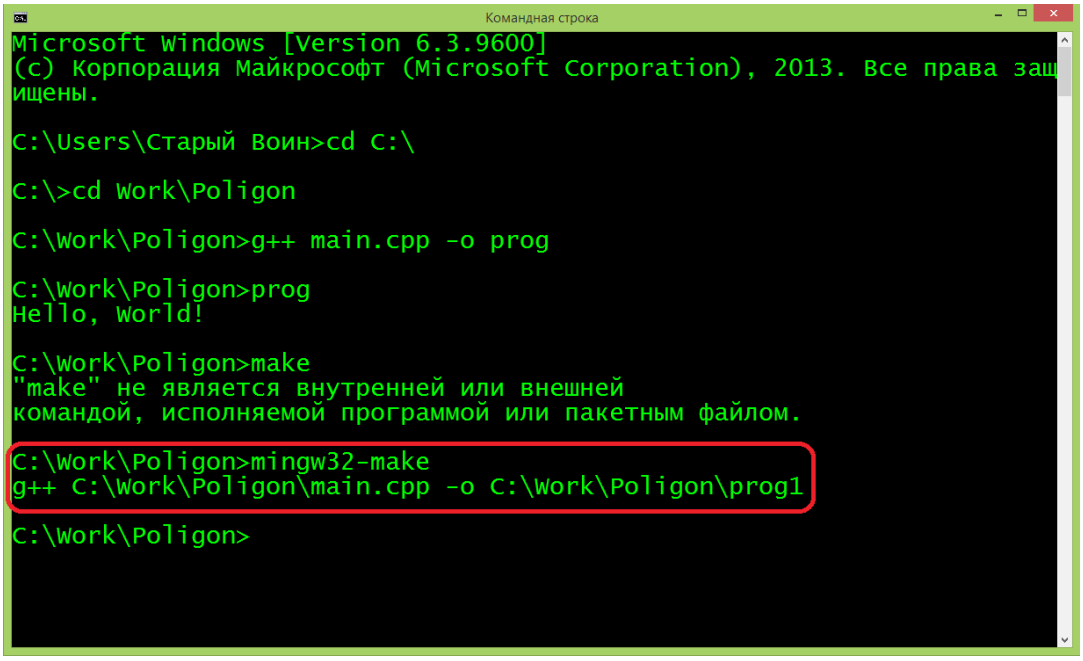

Рисунок 14 — Результат компиляции с использованием утилиты «make».

Проверяем содержимое папки «Poligon» и убеждаемся, что исполняемый файл «prog1.exe» присутствует (рисунок 15).

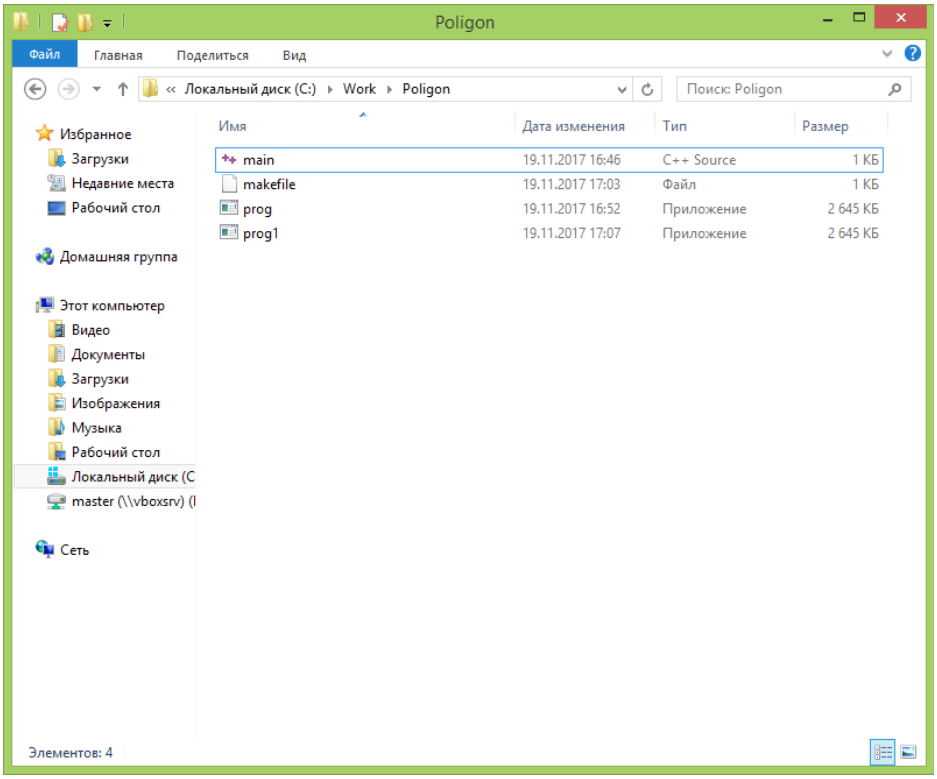

Рисунок 15 — Содержимое папки «Poligon».

Проверяем работу программы (рисунок 16).

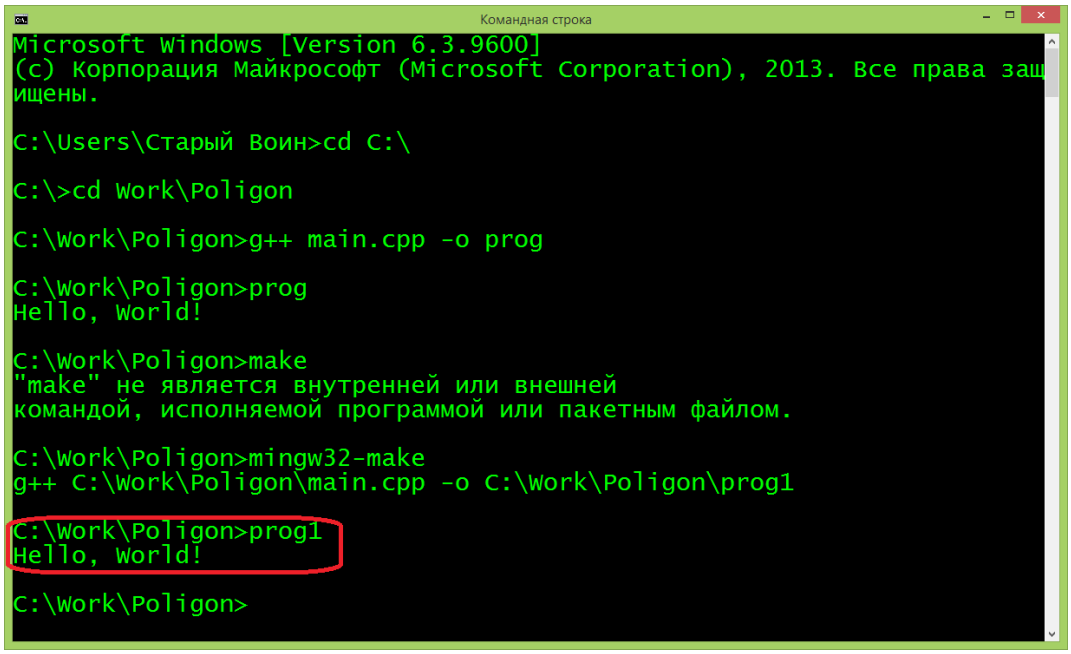

Рисунок 16 — Результат работы программы.

Для того, чтобы упростить работу с этой утилитой, можно создать bat файл, в котором прописать вызов утилиты «mingw32-make.exe».