## Лабораторная работа 1.

### Установка и настройка программного обеспечения.

### Знакомство с правилами написания программного кода.

1. Необходимо скачать и установить компилятор MinGW по ссылке:

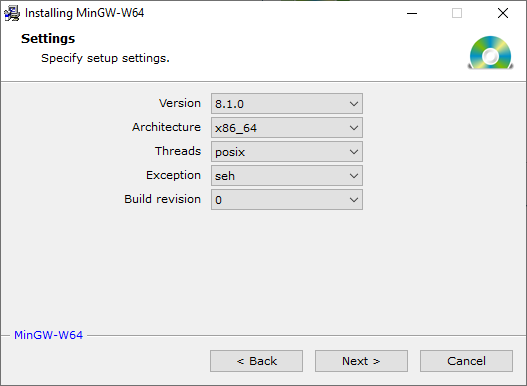
[https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win32/Personal%20Builds/mingw-builds/installer/mingw-w64-install.exe/download](https://sourceforge.net/projects/mingw-w64/files/Toolchains targetting Win32/Personal Builds/mingw-builds/installer/mingw-w64-install.exe/download)

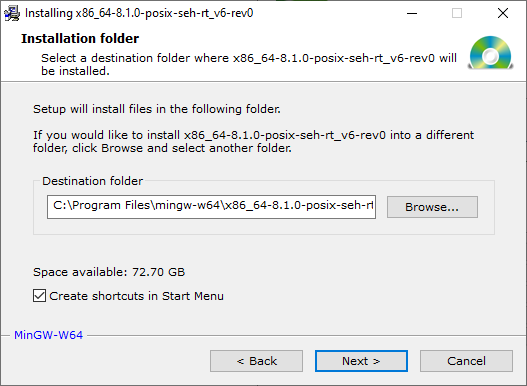
**MinGW** (Minimalist GNU for Windows) — набор инструментов разработки программного обеспечения для создания приложений под Windows. Включает в себя компилятор, родной программный порт GNU Compiler Collection (GCC) под Windows вместе с набором свободно распространяемых библиотек импорта и заголовочных файлов для Windows API.

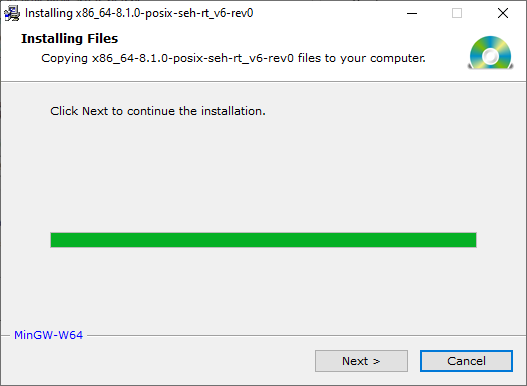
**GNU Compiler Collection** (обычно используется сокращение GCC) — набор компиляторов для различных языков программирования, разработанный в рамках проекта GNU. GCC является свободным программным обеспечением, распространяется фондом свободного программного обеспечения (FSF) на условиях GNU GPL и GNU LGPL и является ключевым компонентом GNU toolchain. Он используется как стандартный компилятор для свободных UNIX-подобных операционных систем.

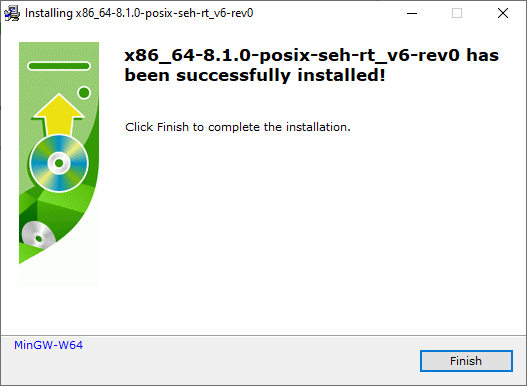


Необходимо выбрать архитектуру **x86\_64**



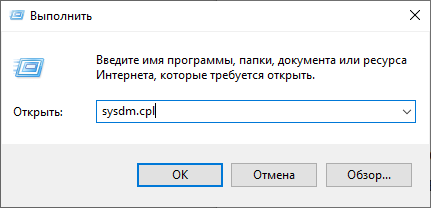


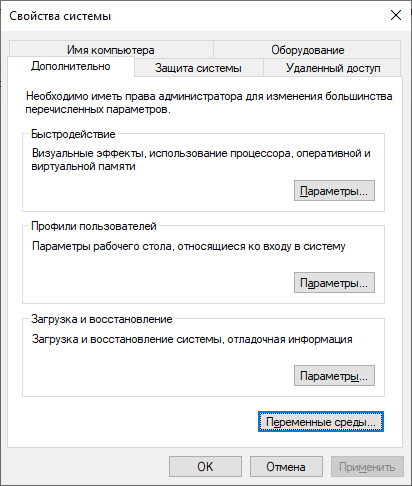




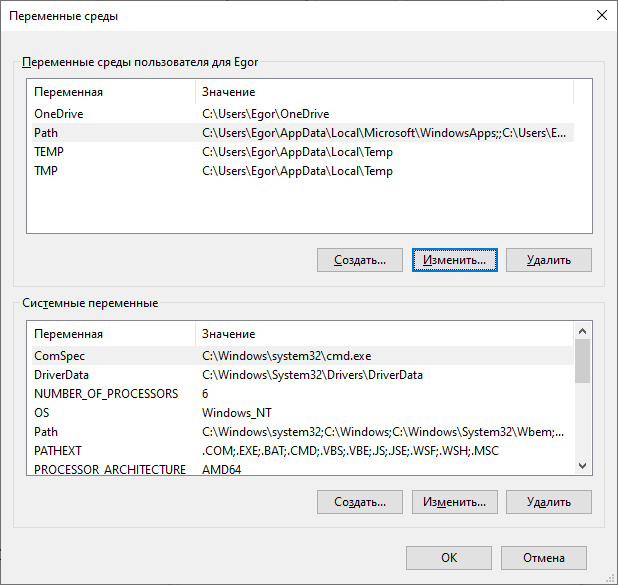
Добавьте путь к папке bin Mingw-w64 в переменную среды Windows PATH, выполнив следующие действия:

1) Нажмите **Win+R** и введите **sysdm.cpl**, чтобы быстро открыть свойства системы. Перейдите во вкладку "Дополнительно" и снизу нажмите на "Переменные среды".

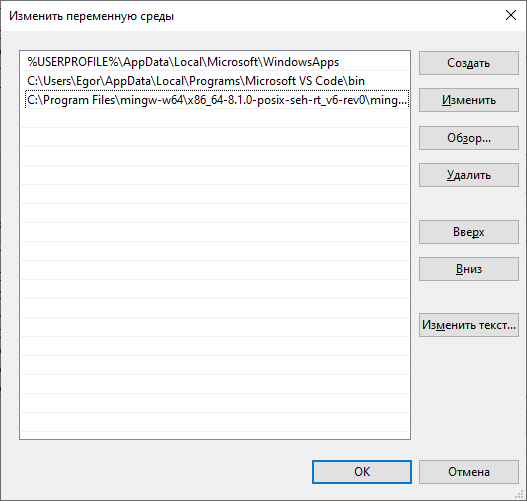




2) Выберите переменную Path и затем нажмите "Изменить".



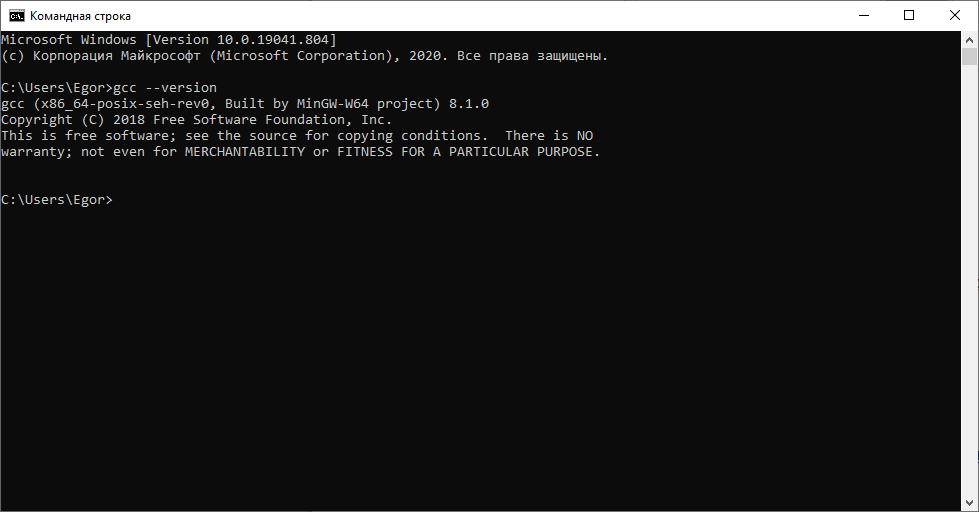
3) Выберите «Создать» и добавьте путь к папке назначения Mingw-w64. Точный путь зависит от того, какую версию Mingw-w64 вы установили и где вы ее установили. Если вы использовали указанные выше настройки для установки Mingw-w64, добавьте этот путь: **C:\Program Files\mingw-w64\x86\_64-8.1.0-posix-seh-rt\_v6-rev0\mingw64\bin**



4) Нажмите OK, чтобы сохранить обновленный Path.

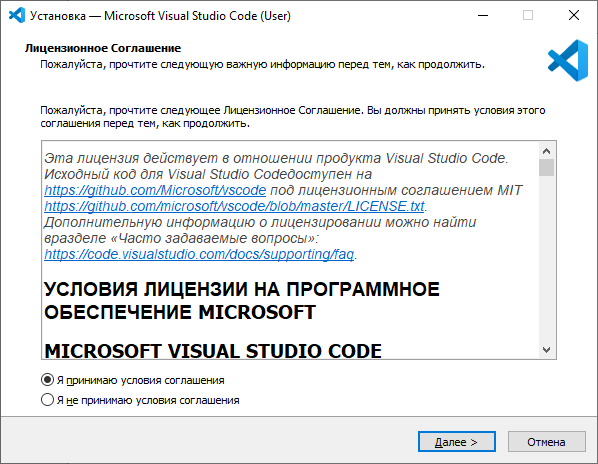
5) Чтобы убедиться, что ваши инструменты Mingw-w64 правильно установлены и доступны, откройте новую командную строку (нажмите Win + R введите слово cmd) и введите:

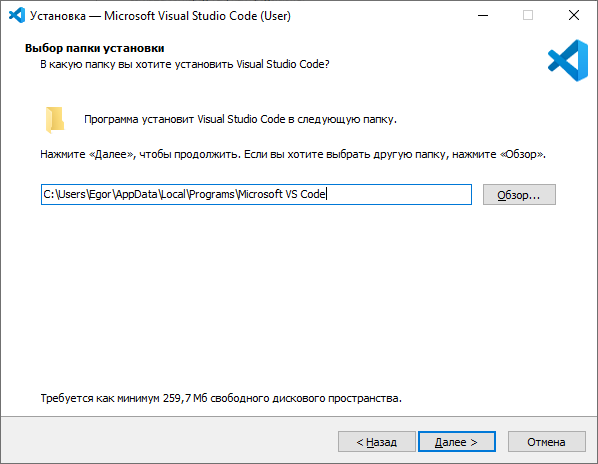
gcc --version

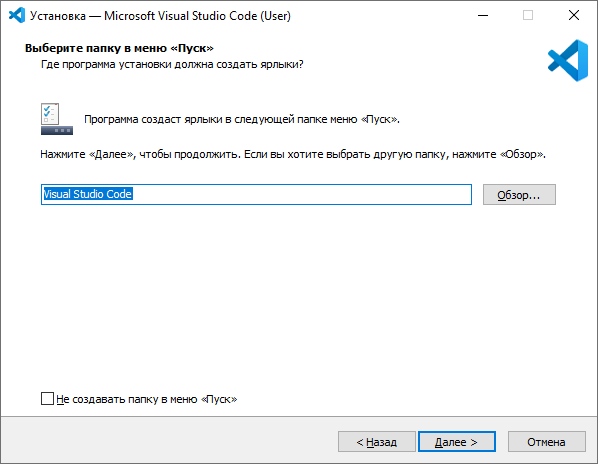


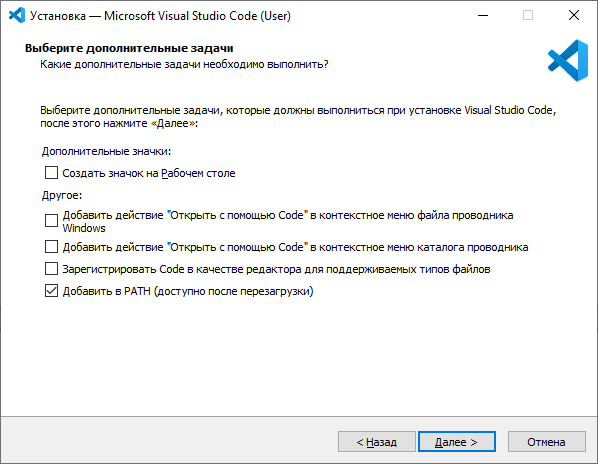
2. Скачайте и установите Visual Studio Code по ссылке <https://code.visualstudio.com/>

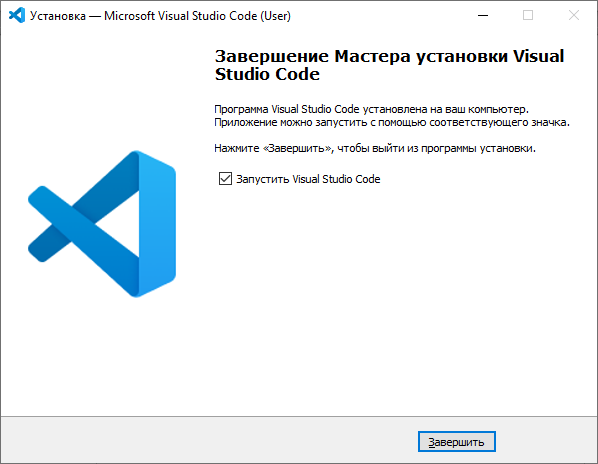
**Visual Studio Code** — редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.







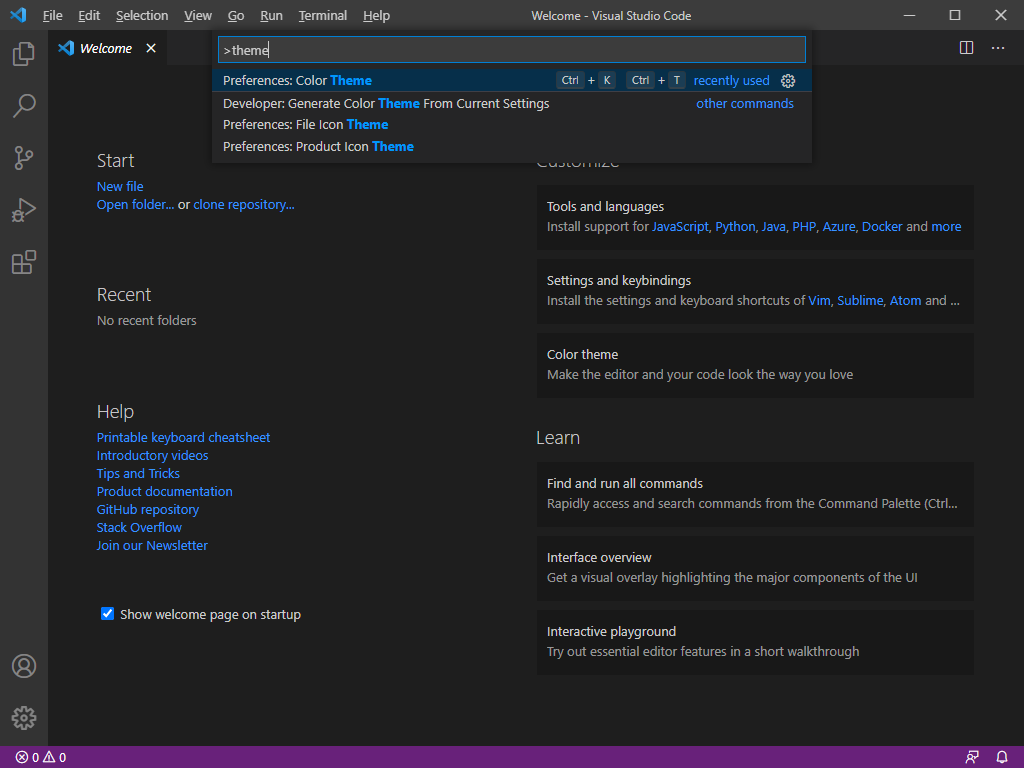




3. Запустите Visual Studio Code.

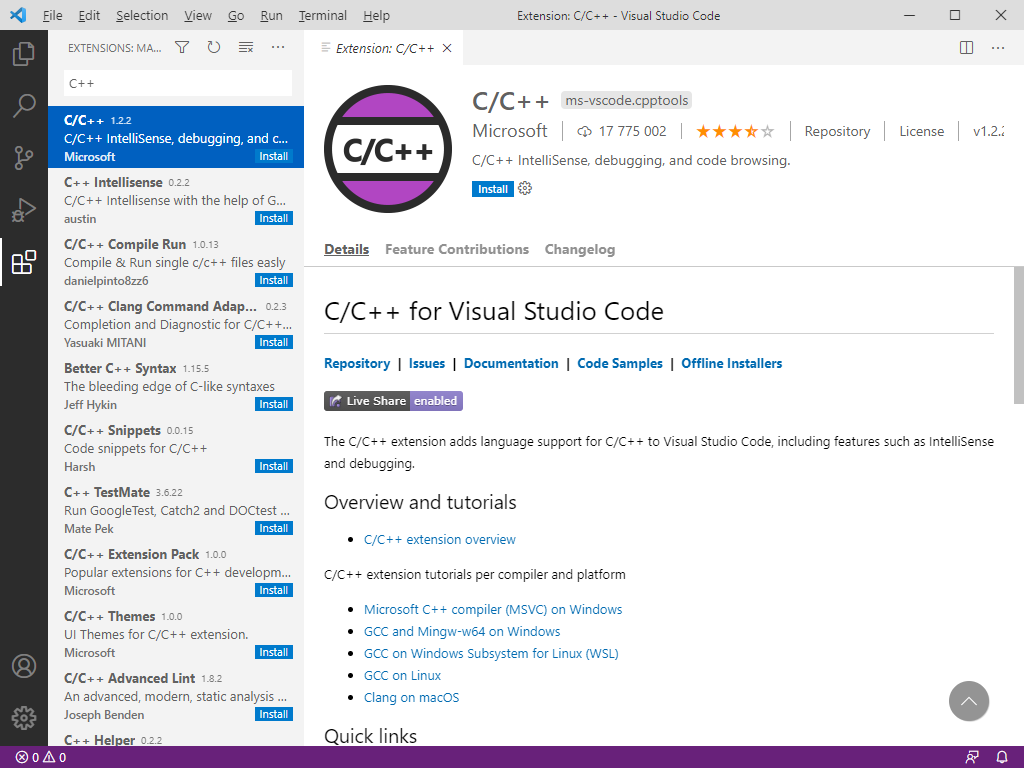
По умолчанию в Visual Studio Code используется тёмная тема оформления, которая может не всем подойти для комфорта глаз.

Чтобы поменять тему оформления, нажмите Ctrl + Shift + P. Откроется палитра команд. Осуществите поиск по слову “theme” и выберите опцию Color Theme. Чтобы настроить иконки, можете выбрать опцию File Icon Theme.



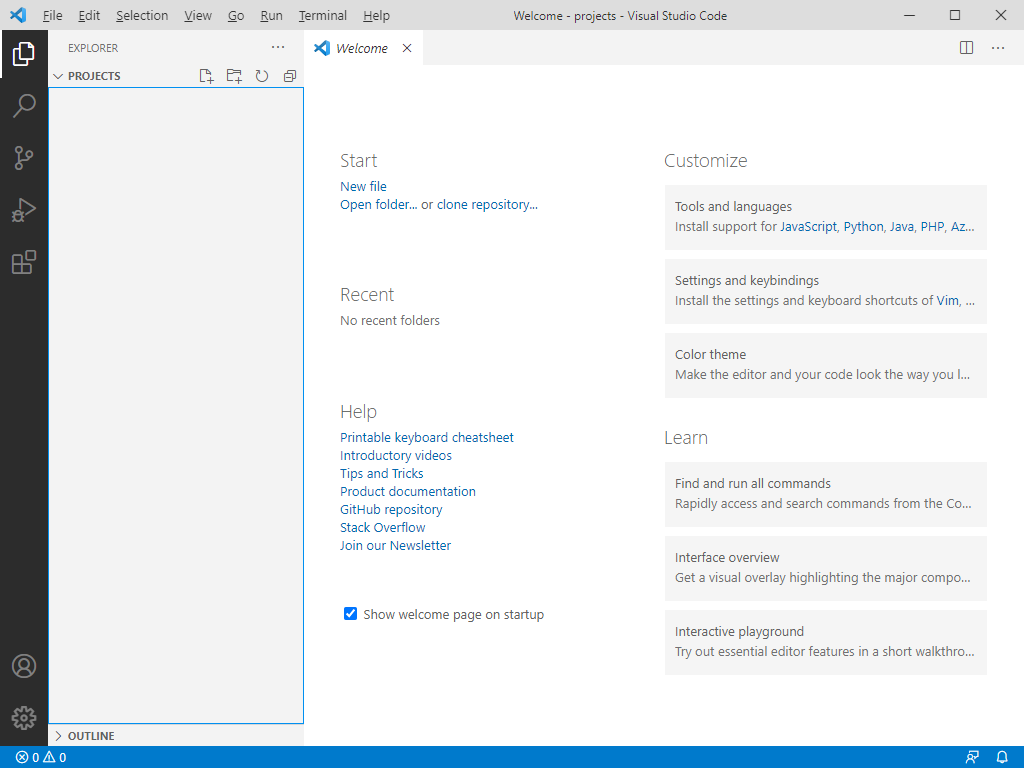
4. Далее необходимо установить расширение [C/C++ extension for VS Code](https://marketplace.visualstudio.com/items?itemName=ms-vscode.cpptools)

Вы можете установить расширение C / C ++, выполнив поиск «c++» в меню «Extensions» (Ctrl + Shift + X). Для установки расширения необходимо нажать кнопку «Install»



5. Через проводник создайте на жёстком диске папку, в которой будете хранить проекты лабораторных работ. Например, папку projects.

Далее откройте созданную папку в Visual Studio Code. **File→Open folder**



Нажмите по иконке  и создайте каталог для первого проекта, например project\_1

Перейдите в созданный каталог и создайте новый файл test.c

6. Напишите первую программу на языке С, которая будет выводить на консоль текст «Hello world»

#include <stdio.h>

int main() {

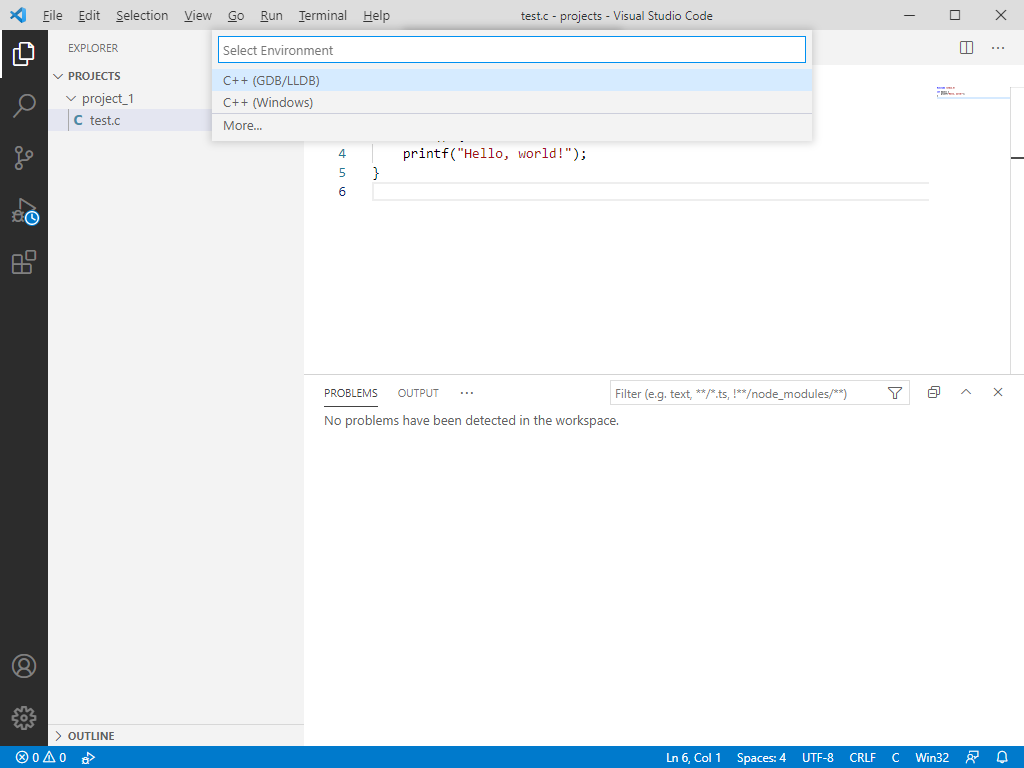
    printf("Hello, world!");

}

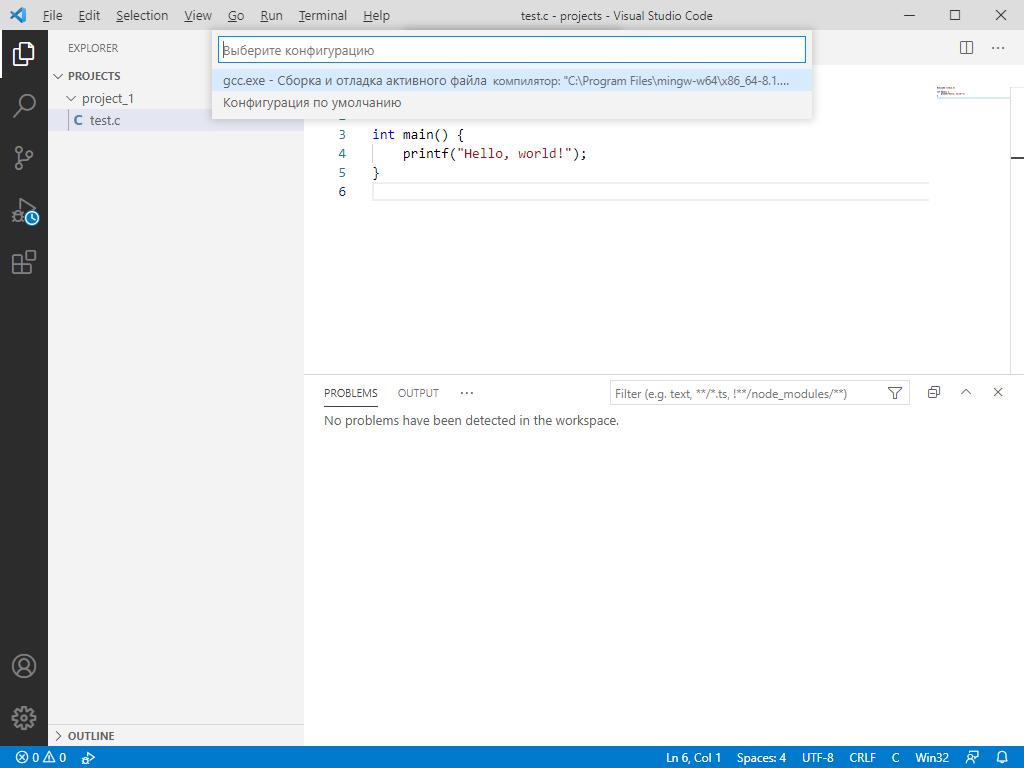
Сохраните созданный файл.

Далее выберите пункт меню **Run > Run without debugging** (или нажмите **Ctrl + F5**).

Выберите C++ (GDB/LLDB)



Далее выберите gcc.exe

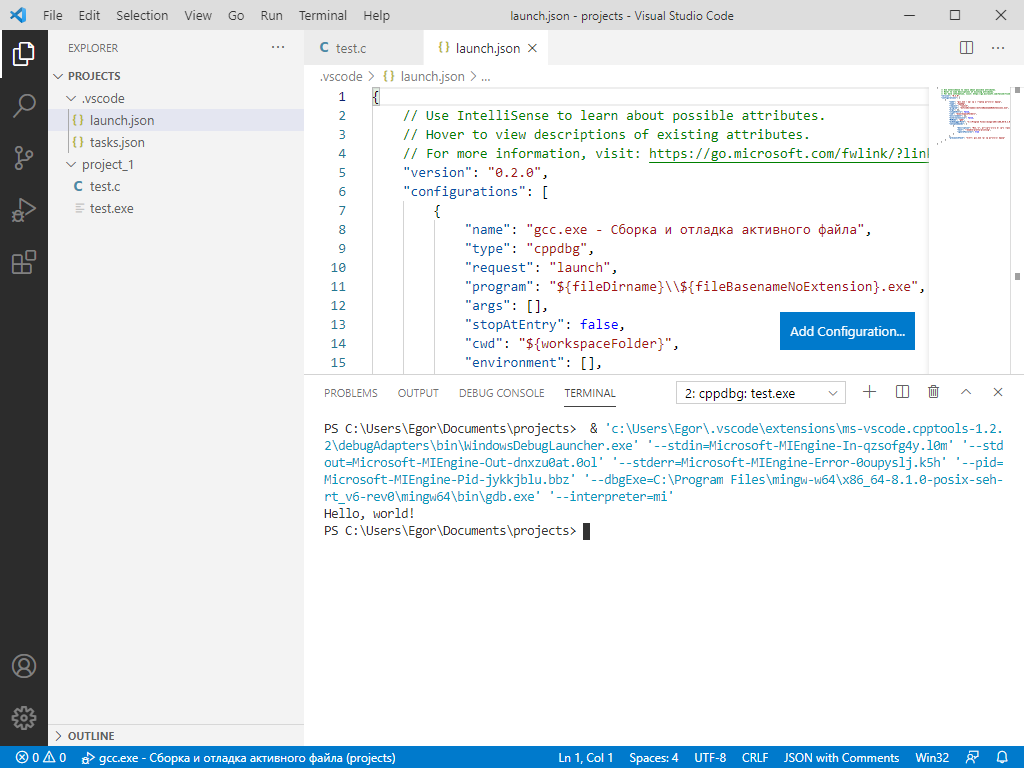


Автоматически сгенерируется файл **launch.json**

Сохраните данный файл.

На вкладке **Terminal** появится результат выполнения, написанной вами программы. Также создастся исполняемый файл **test.exe**

С вкладкой **Terminal** можно работать, как с обычной командной строкой Windows.



7. Измените программный код, чтобы он выводил «Hello, my name is *ВАШЕ\_ИМЯ*»

8. Ознакомьтесь с **обязательными** требованиями к оформлению программного кода в лабораторных работах:

# 1. Соглашения по идентификаторам

## 1.1 Подбор идентификаторов

1.1.1 Все идентификаторы должны выбираться из соображений читаемости и максимальной семантической нагрузки. Например:

**#define** EPS (0.0001) // точность

**int** sum; // сумма

**char** \*message; // сообщение

Неудачными можно считать идентификаторы:

**#define** uU (0.0001) // точность

**int** uu; // сумма

**char** \*zz; // сообщение

1.1.2 Идентификаторы рекомендуется подбирать из слов английского языка. Например:

/\* выдает звуковой сигнал заданной частоты и длительности \*/

**void** beep(**int** hertz, **int** msec)

/\* выдает 1, если файл с именем fname существует \*/

**int** exist\_file(**char** \*fname)

**int** done; // признак окончания работы с программой

**double** width, height; // размеры изделия (ширина, высота)

Не очень удачными можно считать идентификаторы:

/\* выдает звуковой сигнал заданной частоты и длительности \*/

**void** zvuk(**int** chast, **int** dlit)

/\* выдает 1, если файл с именем Im существует \*/

**int** est\_file(**char** \*Im)

**int** konec; // признак окончания работы с программой

**double** shirina, vysota; // размеры изделия (ширина, высота)

## 1.2 Написание идентификаторов

1.2.1 Идентификаторы констант и макроопределений рекомендуется писать заглавными буквами. Например:

**#define** PI (3.14) // значение числа пи

**#define** MAX(x,y) ((x)>(y))?(x):(y) // максимум двух чисел

1.2.2 Существуют разные подходы к написанию остальных идентификаторов. Например:

а) все буквы идентификатора пишутся маленькими, для разделения слов в идентификаторе используется символ "\_"

**int** cnt\_node; // количество звеньев

/\* удалить звено c номером i из списка node \*/

**void** delete\_node(list \*node, **int** i)

б) в идентификаторах каждое слово, входящее в идентификатор, писать, начиная с большой буквы, остальные буквы - маленькие.

**int** CntNode; // количество звеньев

/\* удалить звено c номером I из списка Node \*/

**void** DeleteNode(List \*Node, **int** I)

в) аналогично б) за исключением того, что первая буква идентификатора пишется маленькой.

**int** cntNode; // количество звеньев

/\* удалить звено c номером i из списка node \*/

**void** deleteNode(list \*node, **int** i)

Принятого подхода нужно придерживаться во всем тексте программы.

# 2. Соглашения по самодокументируемости программ

## 2.1 Комментарии

2.1.1 Комментарии в теле программы следует писать на русском языке и по существу так, чтобы программист, не участвовавший в разработке программы (но имеющий опыт работы на языке Си), мог без особого труда разобраться в логике программы, и, при необходимости, сопровождать данный программный продукт.

## 2.2 Спецификация функции и прототипа

Для каждой пользовательской ***функции*** должна быть описана в виде комментария спецификация, содержащая следующую информацию:  
а) назначение функции;  
б) описание семантики параметров-значений (параметров, передаваемых по значению), если она неочевидна;  
в) описание семантики параметров-переменных (параметров, передаваемых по ссылке), если она неочевидна.  
г) описание семантики возвращаемого значения, если она неочевидна.  
Например:

1) семантика параметров и возвращаемого значения очевидна:

/\* возвращает 1, если год year -- високосный \*/

**int** is\_leap\_year(**int** year)

2) семантика параметров очевидна, семантика возвращаемого значения неочевидна

/\* Возвращает день недели даты d/m/y;

год y должен быть в отрезке 1582..4902;

результат: ВСК = 0, ПНД = 1, ВТР = 2, ... СБТ = 6 \*/

**int** day\_of\_week(**int** d, **int** m, **int** y)

3) семантика параметров и возвращаемого значения неочевидна

**#define** MAXN (10)

**typedef** **double** matrix\_t[MAXN, MAXN];

**typedef** **double** vector[MAXN];

/\* Решение системы линейных алгебраических уравнений

методом Гаусса.

Входные данные:

a -- матрица коэффициентов системы;

b -- столбец свободных членов системы;

eps -- точность вычислений.

Выходные данные:

x -- вектор решения;

has\_solution -- флаг, устанавливаемый в 1, если решение

системы существует, и в 0 во всех

остальных случаях;

num\_of\_roots -- число корней в решении системы, может

принимать значения:

0 -- если решение системы не существует,

MAXN -- если решение системы существует и

единственно,

MAXINT -- если существует бесконечное

множество решений;

det -- значение определителя матрицы a;

afor\_reverse -- нижняя треугольная матрица, полученная из a в

в результате выполнения прямого хода алгоритма

Гаусса;

bfor\_reverse -- столбец свободных членов, полученный из b в

в результате выполнения прямого хода алгоритма

Гаусса. \*/

**double** gauss(matrix\_t a, vector b, **double** eps,

vector \* x,

**int** \* has\_solution,

**int** \* num\_of\_roots,

**double** \* det,

matrix\_t \* afor\_reverse,

vector \* bfor\_reverse)

Замечание:  
Если функция реализует какой-либо вычислительный метод (например: нахождение площади фигуры методом трапеций, поиск минимума функции методом Ньютона и т.п.), рекомендуется в теле функции поместить комментарий с кратким описанием метода, либо ссылку на источник, где описан метод.

***Прототипы функций*** достаточно снабдить кратким комментарием назначения функции. Например:

/\* решение системы линейных алгебраических уравнений

методом Гаусса \*/

**double** gauss(matrix\_t, vector, **double**, vector \*, **int** \*, **int** \*, **double** \*, matrix\_t \*, vector \*);

## 2.3 Спецификация программного файла

Программный файл должен начинаться со спецификации в виде комментария, содержащего следующую информацию:  
а) идентификация (имя) файла;  
б) фамилия и копирайт автора;  
в) дата написания файла;  
г) версия языка программирования и замечания по компиляции программного файла в других версиях языка (если требуется);  
д) назначение программного файла;  
Например:

/\* primes.c

--------------------------

(c)оздал: Иванов И.И.

группа : ММ-216

дата : 01/09/07

для : Borland C++ Builder 6.0

-------------------------------------------------------

Подсчет количества простых чисел в промежутке [1..200]. \*/

# 3. Соглашения по читаемости программ

## 3.1 Лесенка

"Лесенка" должна отражать структурную вложенность языковых конструкций. Рекомендуется ***отступ*** не менее 2-х и не более 8-и пробелов. Принятого отступа нужно придерживаться во всем тексте программы. Правила написания ***конструкций*** (K&R стиль):

а) **if - else**

**if** (<условие>) {

<операторы>

} **else** {

<операторы>

}

**if** (<условие>)

<оператор>;

**else**

<оператор>;

б) **while**

**while** (<условие>) {

<операторы>

}

**while** (<условие>)

<оператор>;

в) **do - while**

**do** {

<операторы>

} **while** (<условие>);

**do**

<оператор>;

**while** (<условие>);

г) **for**

**for** (<выражение1>; <выражение2>; <выражение3>) {

<операторы>

}

**for** (<выражение1>; <выражение2>; <выражение3>)

<оператор>;

д) **switch**

**switch** (<выражение>) {

**case** <выражение>:

<операторы>

**break**;

.......

**default:**

<операторы>

}

е) **определение функции**

<тип> <имя\_функции>(<список\_параметров>)

{

<операторы>

}

Например:

**int** sign(**double** x)

/\* выдает знак числа x \*/

{

**int** result;

**if** (x > 0)

result = 1;

**else**

**if** (x < 0)

result = -1;

**else**

result = 0;

**return** result;

}

/\* нахождение действительных корней квадратного уравнения;

a, b, c -- коэффициенты

x1, x2 -- корни (если действительного решения нет,

то полагаются равными 0);

num -- число корней (0, 1, или 2) \*/

**void** equation(**double** a, **double** b, **double** c, **double** \*x1, **double** \*x2, **int** \*num)

{

**double** d;

d = pow(b, 2) - 4 \* a \* c;

**if** (d < 0) {

\*num = 0;

\*x1 = 0.0;

\*x2 = 0.0;

} **else** {

\*x1 = (- b + sqrt(d)) / (2 \* a);

\*x2 = (- b - sqrt(d)) / (2 \* a);

**if** (\*x1 == \*x2)

\*num = 1;

**else**

\*num = 2;

}

}

## 3.2 Длина строк программного текста

Длина строк программы не должна превышать ширины экрана (80 символов). Инструкции длиннее 80 символов разбиваются на логические части, которые всегда значительно короче, чем изначальная строка, и распологаются со сдвигом вправо. То же самое относится к заголовкам функций с длинным списком аргументов и к длинным строковым константам. Например:

**double** function(**double** param1, **double** param2,

**double** param3, **double** param4)

{

**double** result;

result = param1 + sqrt(param2) - param3 +

pow(param3, param4) \* cos(param4);

printf("Внимание! Это длинный printf с 5-ью параметрами: "

"1-й параметр = %f, 2-й параметр = %f, 3-й параметр = %f, "

"4-й параметр = %f, результат: %f\n", param1, param2, param3,

param4, result);

**return** result;

}

## 3.3 Прочие рекомендации

3.3.1 Рекомендуется при перечислении идентификаторов после запятой "," ставить один пробел " ". Например:

**double** a, b;

printf("Сумма: %f. Разность: %f.", a + b, a - b);

3.3.2 Рекомендуется всегда писать символ-разделитель операторов ";" (непосредственно после оператора). Например:

**switch** (num) {

**case** 1:

printf("один...");

**break**;

**case** 2:

printf("два...");

**break**;

**case** 3:

printf("три..."); // <-- здесь

**break**;

**default:**

printf("много!"); // <-- и здесь

}

**if** (n < 0) {

printf("Введено неверное значение n, прерываем работу!");

exit(0); // <-- здесь

}

9. Оформите программный код из прошлого задания по указанным правилам.

10. Загрузите отчёт по лабораторной работе в систему Moodle. Отчёт должен содержать.

- исходный код конечной программы (файл с расширением **.с**)

- скриншот работающей программы

Файлы заархивировать в **zip** архив.