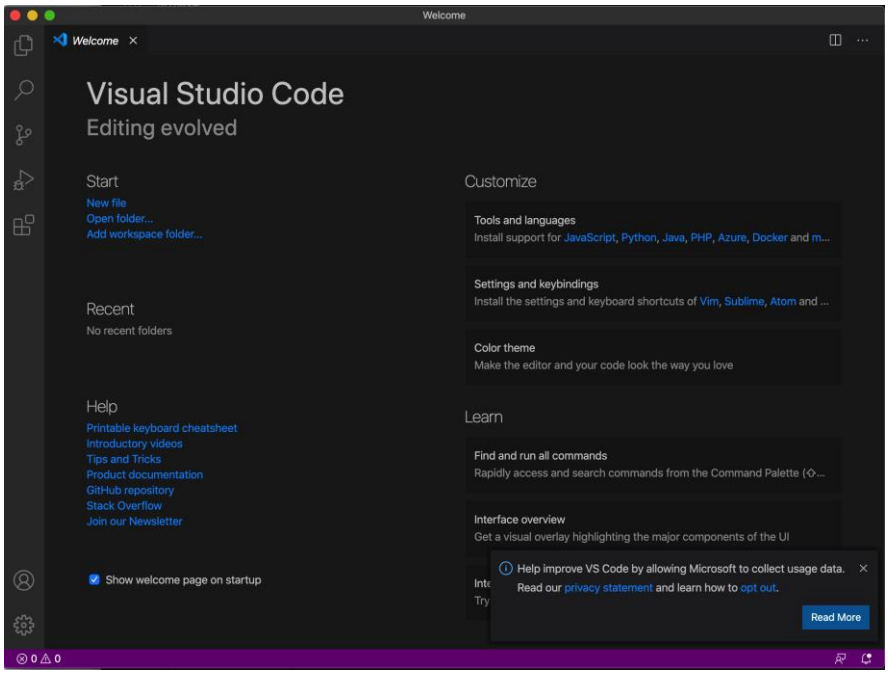
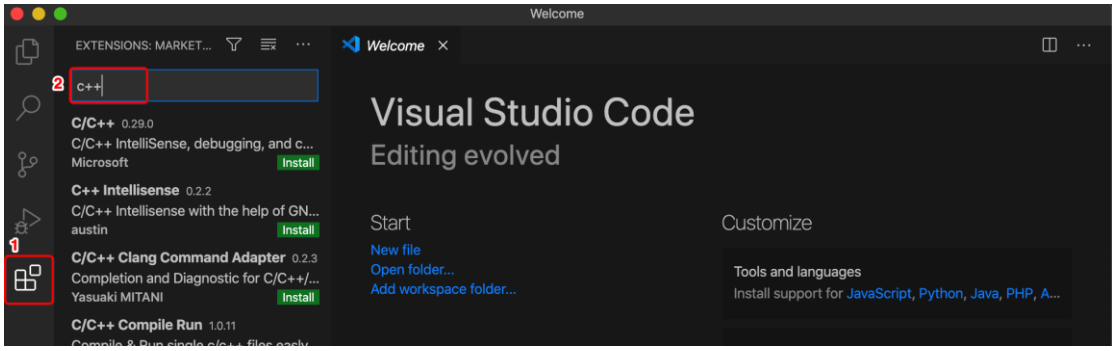
VS Code Installation and Usage for MacOS

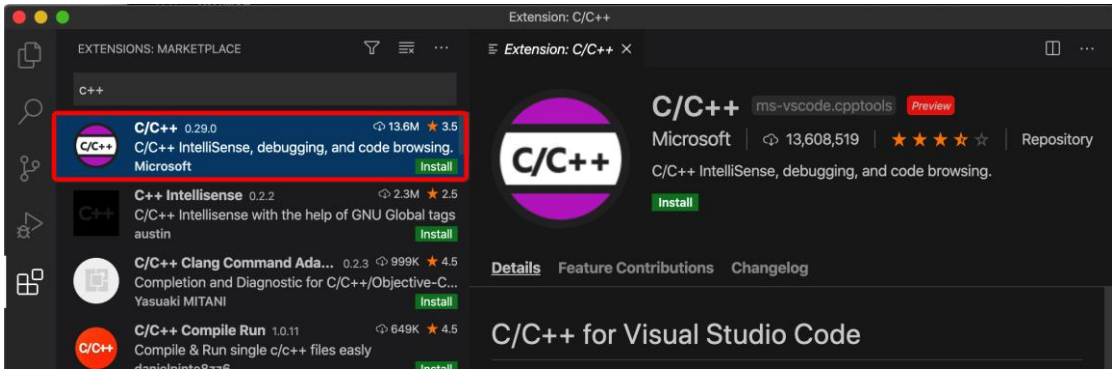
- **Prerequisites**

1. Install Visual Studio Code on macOS.



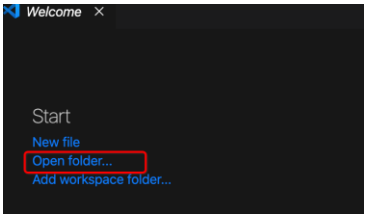
2. Install the C++ extension for VS Code.





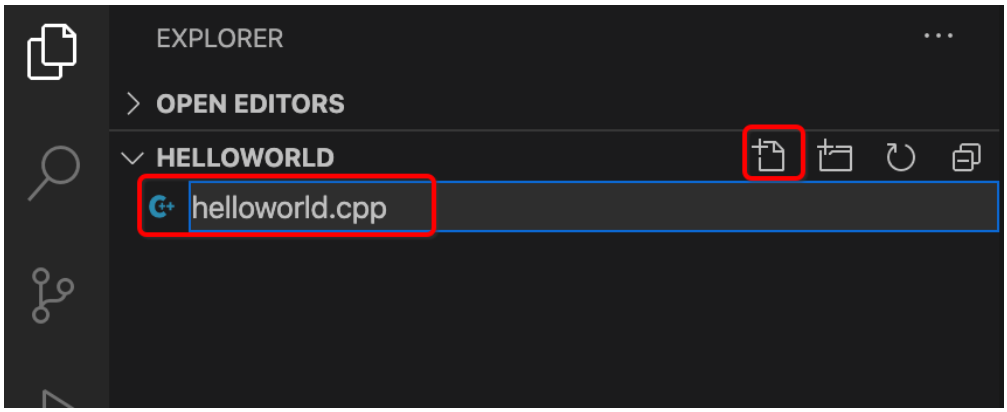
- **Create Hello World**

Create a new folder “projects” and click “File -> Open Folder…” to select the folder just created



- **Add hello world source code file**

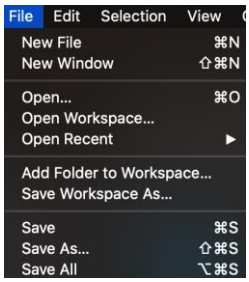
In the File Explorer title bar, select **New File** and name the file `helloworld.cpp`.



Paste in the following source code:

```
#include <iostream>
using namespace std;
int main( )
{
    cout << "Hello world! :)" << endl;
}
```

Remember save the file, do “File -> Save”.



- **Set g++ as the default compiler**

Clang is the default compiler for Mac OS X. Clang may already be installed on your Mac. To verify that it is, open a macOS Terminal window and enter the following command:

clang --version

```
(base) quanyuqing@MacBook-Pro ~ % clang --version
Apple clang version 11.0.3 (clang-1103.0.32.62)
Target: x86_64-apple-darwin19.6.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
```

```
(base) quanyuqing@MacBook-Pro ~ % g++ -v
Configured with: --prefix=/Library/Developer/CommandLineTools/usr --with-gxx-include-dir=/Library/Developer/
CommandLineTools/SDKs/MacOSX.sdk/usr/include/c++/4.2.1
Apple clang version 11.0.3 (clang-1103.0.32.62)
Target: x86_64-apple-darwin19.6.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
```

In COMP2011, COMP2012, and COMP2012H officially we use the g++ compiler.
Here's how to use g++ instead of clang++:

1. Use homebrew to help

Homebrew

Homebrew "installs the stuff that you need that Apple don't". It's like Ubuntu's apt-get, where one can install packages easily from repositories. Instead of having to download, configure, and install something yourself, all you need to do is run one command, and Homebrew will take care of the rest for you.

Pre-requisites

Homebrew requires that you have either Xcode or the Xcode command line tools installed on your Mac. Xcode is a free integrated development environment similar to Eclipse designed by Apple and mainly intended for iOS development or targeting the compiler. In this class, we will focus on .

Refer to https://brew.sh/ to install Homebrew

2. Installing GCC

brew search gcc

```
(base) quanyuqing@MacBook-Pro ~ % brew search gcc
==> Formulae
gcc                     gcc@5                   gcc@7                   gcc@9                   x86_64-elf-gcc
gcc@4.9                 gcc@6                   gcc@8                   i686-elf-gcc
==> Casks
gcc-arm-embedded
```

brew install gcc@9

```
(base) quanyuqing@MacBook-Pro ~ % brew install gcc@9
Updating Homebrew...
==> Downloading https://homebrew.bintray.com/bottles/gmp-6.2.0.catalina.bottle.tar.gz
==> Downloading from https://d29vzk4ow07wi7.cloudfront.net/2e6acd6e62d1b8ef0800061e113aea30a63f56b32b99c010234c0420fd6d3
ecf?
==> Downloading https://homebrew.bintray.com/bottles/isl-0.22.1.catalina.bottle.tar.gz
==> Downloading from https://d29vzk4ow07wi7.cloudfront.net/b5319e3bbbb36ef3536d841999b7497b3dce4bf9e07fb04f6b0db716e0878
96d?
==> Downloading https://homebrew.bintray.com/bottles/mpfr-4.1.0.catalina.bottle.tar.gz
==> Downloading from https://d29vzk4ow07wi7.cloudfront.net/5fcf57834f58c18761c6c7b0eb961eb7f9fc54325b5361bf3a17c4dee6ebc
08a?
```

```
gcc-9 -v
```

```
(base) quanyuqing@MacBook-Pro ~ % gcc-9 -v
Using built-in specs.
COLLECT_GCC=gcc-9
COLLECT_LTO_WRAPPER=/usr/local/Cellar/gcc@9/9.3.0/libexec/gcc/x86_64-apple-darwin19/9.3.0/lto-wrapper
Target: x86_64-apple-darwin19
Configured with: ../configure --build=x86_64-apple-darwin19 --prefix=/usr/local/Cellar/gcc@9/9.3.0 --libdir=/usr/local/Cellar/gcc@9/9.3.0/lib/gcc/9 --disable-nls --enable-checking=release --enable-languages=c,c++,objc,obj-c++,fortran --program-suffix=-9 --with-gmp=/usr/local/opt/gmp --with-mpfr=/usr/local/opt/mpfr --with-mpc=/usr/local/opt/libmpc --with-isl=/usr/local/opt/isl --with-system-zlib --with-pkgversion='Homebrew GCC 9.3.0' --with-bugurl=https://github.com/Homebrew/homebrew-core/issues --disable-multilib --with-native-system-header-dir=/usr/include --with-sysroot=/Library/Developer/CommandLineTools/SDKs/MacOSX10.15.sdk SED=/usr/bin/sed
Thread model: posix
gcc version 9.3.0 (Homebrew GCC 9.3.0)
```

3. Change configuration parameters and environment variable

```
nano ~/.bash_profile
```

Type the following path:

```
alias gcc='gcc-9'
alias cc='gcc-9'
alias g++='g++-9'
alias c++='c++-9'
```

```
quanyuqing — nano ~/.bash_profile — 119×37
  GNU nano 2.0.6                File: /Users/quanyuqing/.bash_profile

alias gcc='gcc-9'
alias cc='gcc-9'
alias g++='g++-9'
alias c++='c++-9'
```

Exit and Save

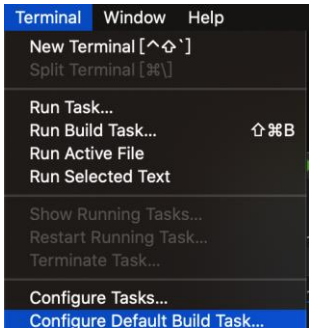
4. Refresh environment variables

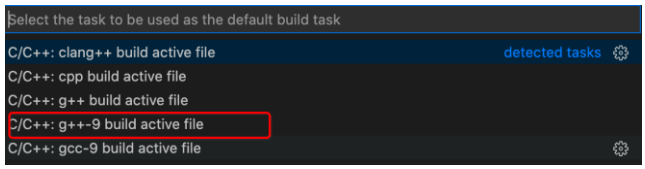
```
source ~/.bash_profile
g++ -v
```

```
(base) quanyuqing@MacBook-Pro ~ % sudo vi ~/.bash_profile
Password:
(base) quanyuqing@MacBook-Pro ~ % source ~/.bash_profile
(base) quanyuqing@MacBook-Pro ~ % g++ -v
Using built-in specs.
COLLECT_GCC=g++-9
COLLECT_LTO_WRAPPER=/usr/local/Cellar/gcc@9/9.3.0/libexec/gcc/x86_64-apple-darwin19/9.3.0/lto-wrapper
Target: x86_64-apple-darwin19
Configured with: ../configure --build=x86_64-apple-darwin19 --prefix=/usr/local/Cellar/gcc@9/9.3.0 --libdir=/usr/local/Cellar/gcc@9/9.3.0/lib/gcc/9 --disable-nls --enable-checking=release --enable-languages=c,c++,objc,obj-c++,fortran --program-suffix=-9 --with-gmp=/usr/local/opt/gmp --with-mpfr=/usr/local/opt/mpfr --with-mpc=/usr/local/opt/libmpc --with-isl=/usr/local/opt/isl --with-system-zlib --with-pkgversion='Homebrew GCC 9.3.0' --with-bugurl=https://github.com/Homebrew/homebrew-core/issues --disable-multilib --with-native-system-header-dir=/usr/include --with-sysroot=/Library/Developer/CommandLineTools/SDKs/MacOSX10.15.sdk SED=/usr/bin/sed
Thread model: posix
gcc version 9.3.0 (Homebrew GCC 9.3.0)
(base) quanyuqing@MacBook-Pro ~ %
```

- **Build helloworld.cpp**

From the main menu, choose **Terminal** > **Configure Default Build Task**. Choose **C/C++ g++ build active file** to build the file that is currently displayed (active) in the editor.

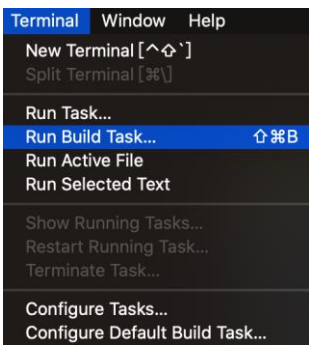




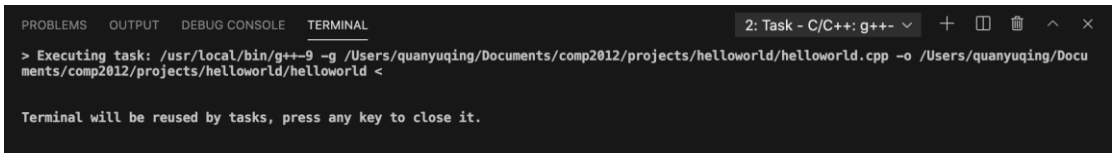
This will create a `tasks.json` file in the `.vscode` folder and open it in the editor.

- **Running the build**

1. Go back to `helloworld.cpp`.
2. To run the build task that you defined in tasks.json, click **Terminal** main menu and choose **Run Build Task**.



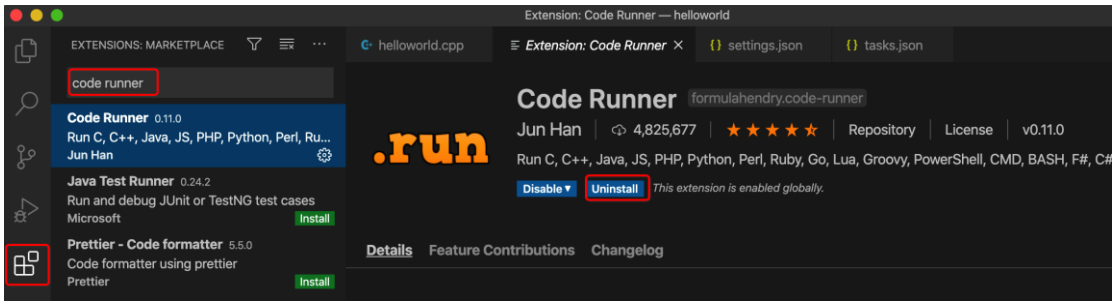
3. For a successful build, the output looks something like this:



4. Create a new terminal using the + button and you'll have a new terminal with the `helloworld` folder as the working directory. Run `ls` and you should now see the executable `helloworld` along with the debugging file (`helloworld.dSYM`).

5. You can run `helloworld` in the terminal by typing `./helloworld`.

- **Add "Code Runner"**



Now open the command palette, type and run "Preferences: Open Settings (JSON)". Put in the following and save the file:

```
{
    "code-runner.customCommand": "make",
    "code-runner.runInTerminal": true,
    "code-runner.executorMap": { "cpp" : "cd $dir && /usr/local/bin/g++-9 -std=c++11 $fileName -o $fileNameWithoutExt &&
$dir$fileNameWithoutExt" }
}
```

Go to “helloworld.cpp”, click “Run Code”