How to create a project in C++

ADTs

We use classes and separate compilation to create an Abstract Data Type (ADT).

An ADT is a user-defined data type, where other users are allowed to *use* the data type, i.e. to declare variables of this data type and to use the functions provided by the data type, but are not allowed to modify the functions provided by the ADT. Separate compilation allow placing the interface and implementation of an ADT in files separate from each other and separate from the programs that use the ADT.

An object is an instance of a particular class. A class hides information by restricting access to it. In effect the data members and member functions within an object are hidden from users to prevent inadvertent changes. Users of the class simply need to understand how to use the interface or interaction between the object and its methods. Typically if we use a class, we need a class definition (in the .h file), an implementation of the class (in the .cpp file with the same name as the .h file for the class) and an application file (main.cpp in the project) where we use the class.

Separate Compilation

C++ allows you to divide a program into parts. Each part can be stored in a separate file and can be compiled separately and then linked together when (or just before) the program is run. Any software project which is slightly more than trivial is generally divided into separate files.

Programs that use user-defined classes usually also use multiple source files. Typically the definition (specification) of the class will be placed in a **.h** file, called the interface or specification file, while the implementation will be placed in a **.cpp** file. This offers, among others, benefits such as compiling the files separately and software reuse. For example, the interface may be required (with `#include`) on multiple occasions within one or more projects' code, but the implementation need only be compiled once.

To use separate compilation, you create a ***project*** to which the files to be included can be added.

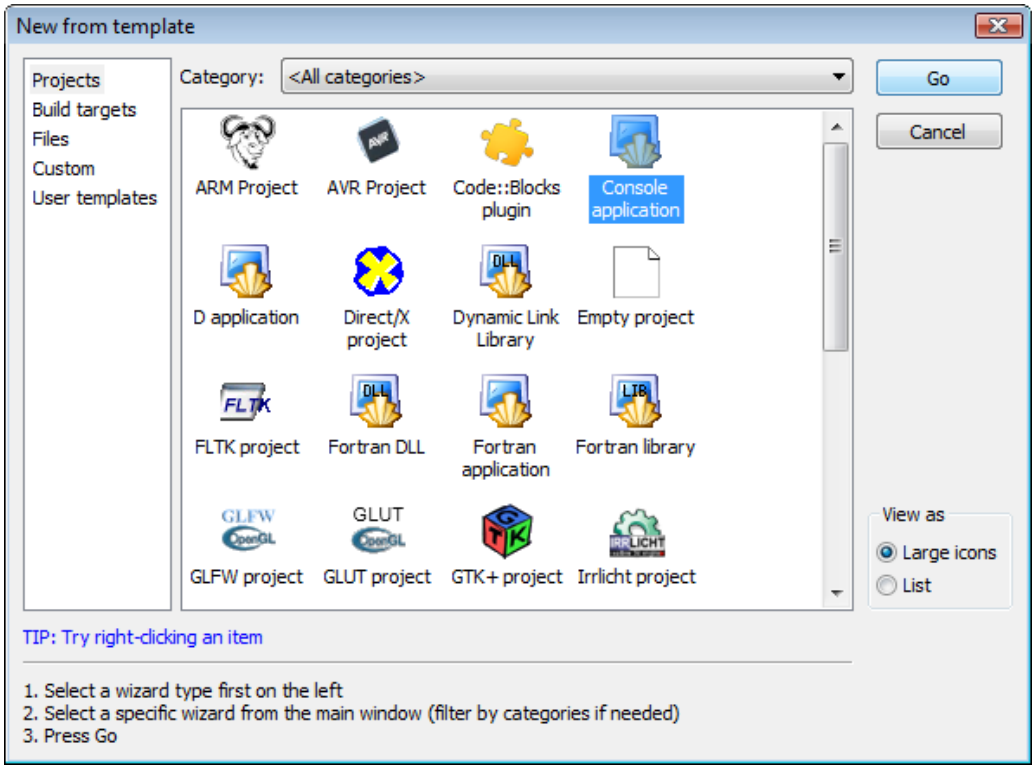
In C++ we use projects to implement separate compilation.

Creating a project

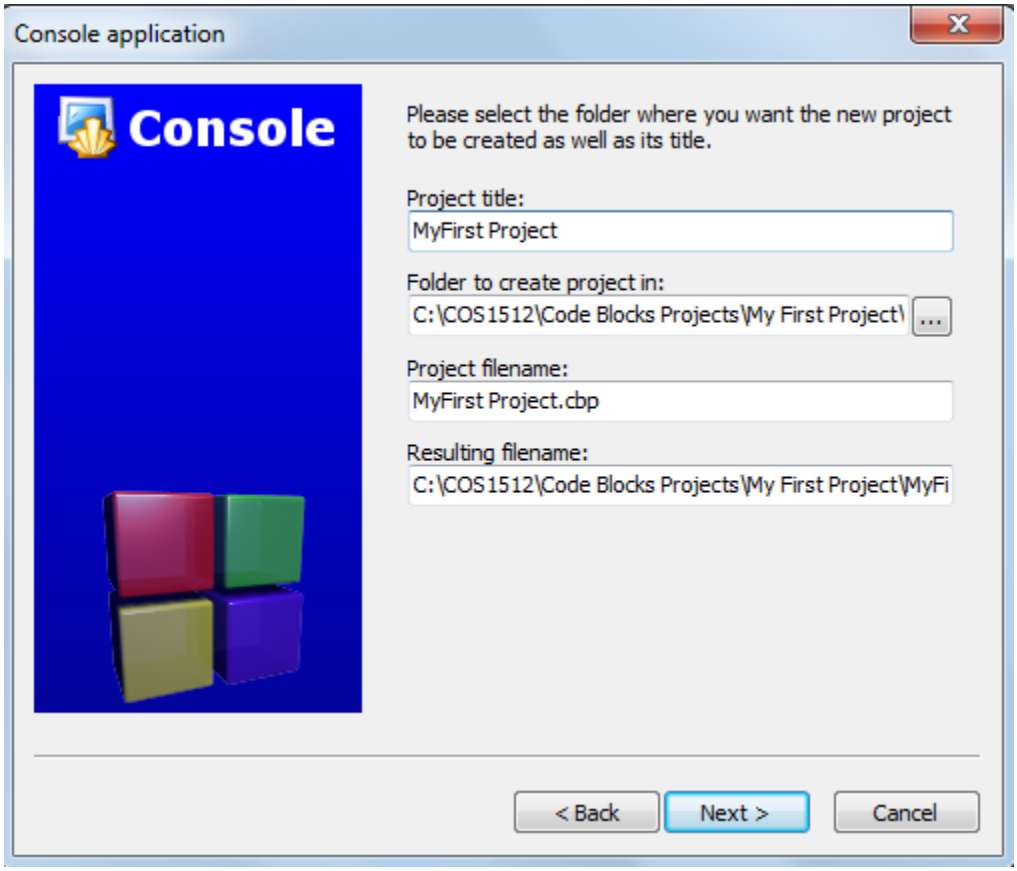
This notes show how to create a project, with a very small header or interface file (.h file, and a corresponding implementation file. Note that the example we use to illustrate how to create a project, does not define a class. However, the principles to define and implement a class by creating a project, is the same as this example.

Starting a new project

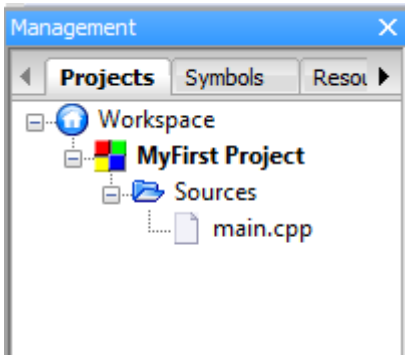
Launch the Project Wizard through *File->New->Project.* From the pre-configured templates for various types of projects, select **Console application** and click **Go**.



The console application wizard will appear next. Continue through the menus, selecting **C++** when prompted for a language. In the next screen, give the project a **name** and type or select a destination folder. As seen below, Code::Blocks will generate the remaining entries from these two.



Finally, the wizard will ask if this project should use the default compiler (normally GCC) and the two default builds: **Debug** and **Release**. All of these settings are fine. Press finish and the project will be generated. The main window will turn gray, but that is not a problem, the source file needs only to be opened. In the **Projects** tab of the **Management** panel on the left expand the folders and double click on the source file **main.cpp** to open it in the editor.



This file contains the following standard code.

main.cpp

```
1. #include <iostream>
2.
3. usingnamespacestd;
4.
5. int main()
6. {
7. cout<<"Hello world!"<<endl;
```

```
8. return0;
9. }
```

The header file

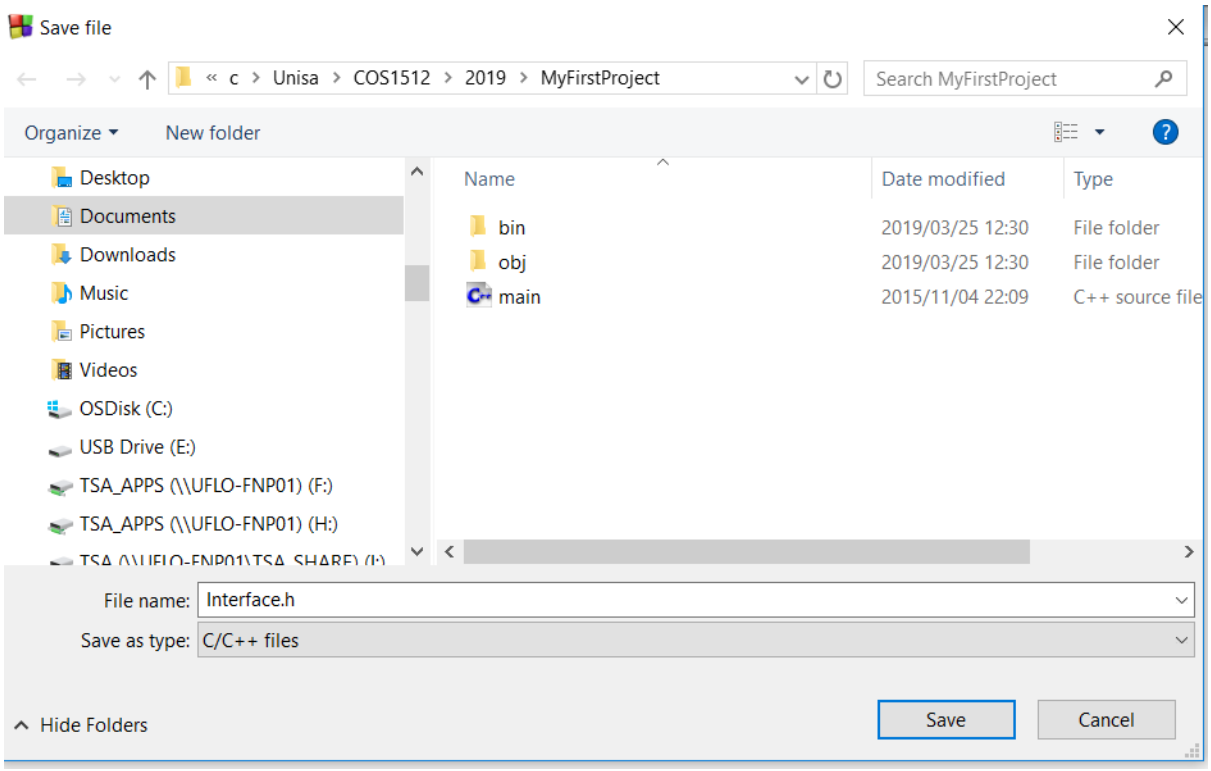
The header file or interface, the .h file, are typically not added to the project, but must be saved in the same folder as the project.

Copy an existing file to your project folder or launch a plain text editor (for example Notepad), and as an example, add the following code.

Interface.h

```
1. #ifndef INTERFACE_H
2. #define INTERFACE_H
3.
4. void hello();
5.
6. #endif // INTERFACE_H
```

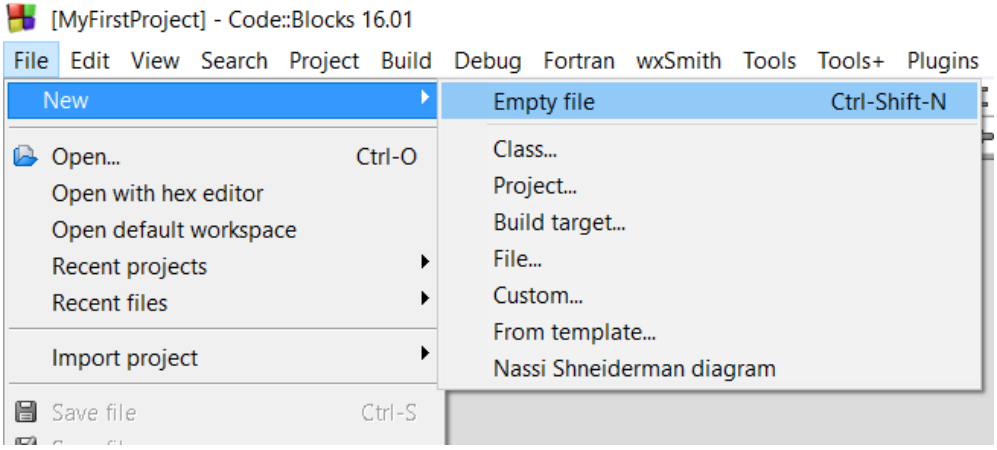
Save this file as a header (**Interface.h**) in the same directory as the other source files in this project.



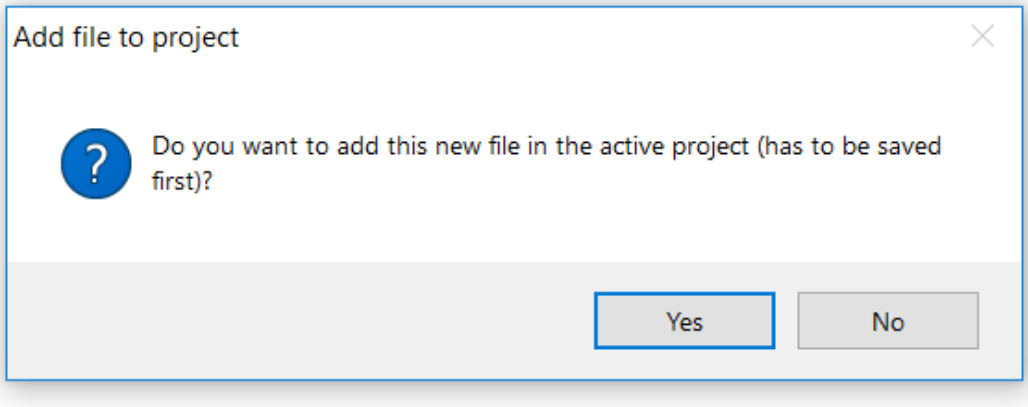
Note that the header file, when defining a class, containing the class definition (the.h file) and the implementation file for the class (a .cpp file) have the same names but different extensions.

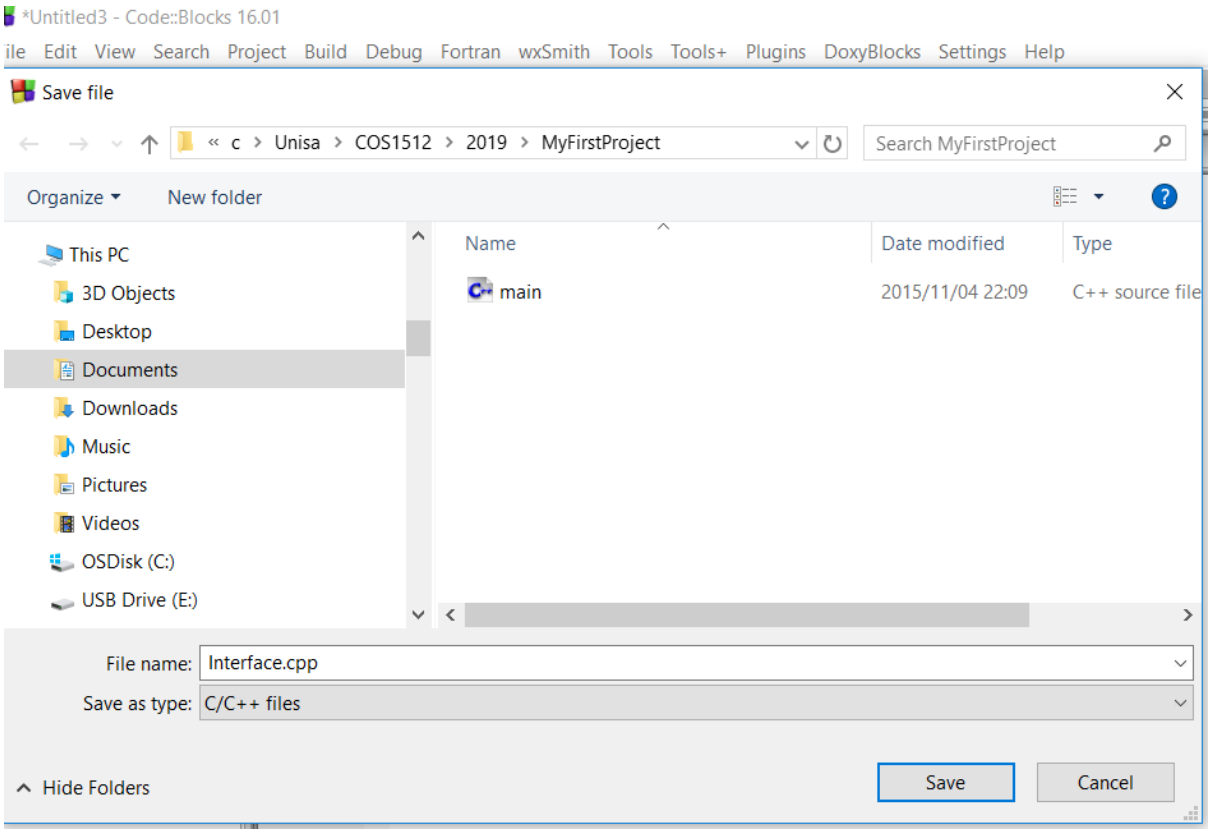
Adding a file to your project (e.g. the implementation file)

To add the new file to the project, bring up the file template wizard through either *File->New->File...* or *Main Toolbar->New file (button)->File...*

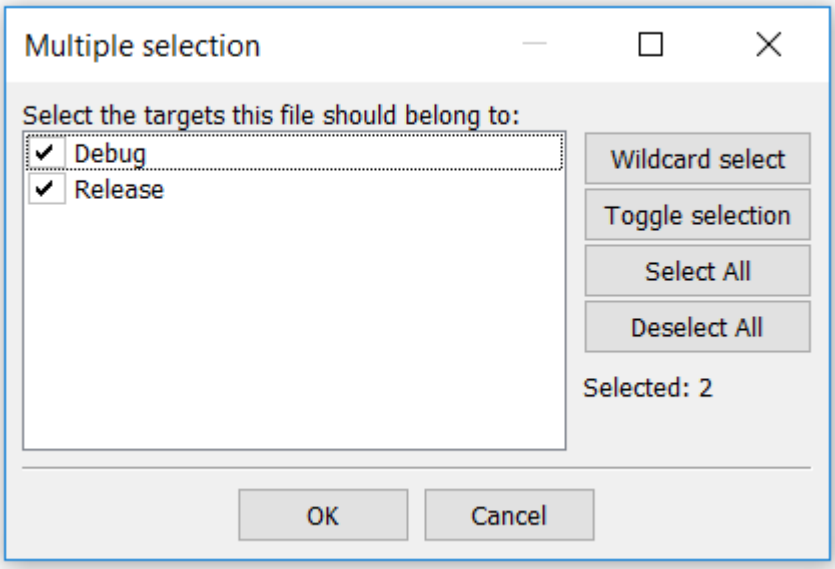


Select the 'Yes' option to add the file to the project.





Enter the new filename and location (as noted, the full path is required). You can browse for the file by clicking the browse button (see below) to display the file browser window to save the file's location. Checking **Save** will return the following screen:

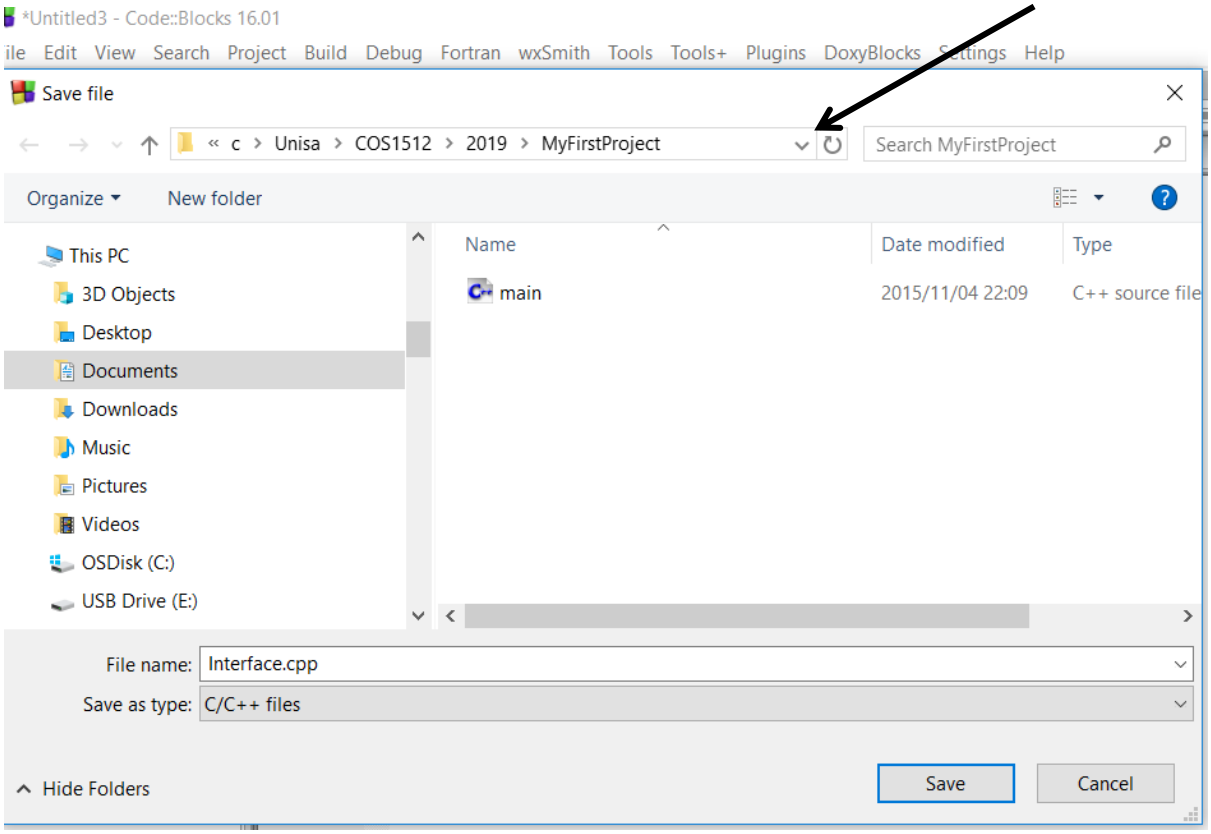


The wizard will ask if this project should use the two default builds: **Debug** and **Release**. All of these settings are fine.

When you check OK, the filename will be stored in the **Sources** folder of the **Projects** tab of the **Management** panel. Checking any of the build targets will alert

Code::Blocks that the file should be compiled and linked into the selected target(s). click **Finish** to generate the file.

Browse button



The newly created file should open automatically; if it does not, open it by double clicking on its file in the **Projects** tab of the **Management** panel. You can now add code to the new file. Be careful not to save your files with **`.c`** extension (this is not a C++ extension).

Add the following code to `Interface.cpp`:

```
#include "Interface.h"
#include <iostream>
using namespace std;

void hello()
{
    cout << "Hello" << endl;
}
```

Returning to the main source (**main.cpp**) include the header file and replace the `cout` function to match the new setup of the project.

main.cpp

```
1. #include "Interface.h" //include header file
2.
```

```
3. int main()
4. {
5.     hello();//calling the function defined in the header
                //file will cause the implementation of the
                //function in the implementation file
                // (Interface.cpp) to be executed
6. return0;
7. }
```

Press *Ctrl-F9* or *File->Build*, or*Compiler Toolbar->Build (button - the gear)* to compile the project. If the following output is generated in the build log (in the bottom panel) then all steps were followed correctly.

```
-------------- Build: Debug in MyFirst Project ---------------

Compiling: main.cpp
Linking console executable: bin\Debug\MyFirst Project.exe
Output size is 913.10 KB
Process terminated with status 0 (0 minutes, 1 seconds)
0 errors, 0 warnings
```

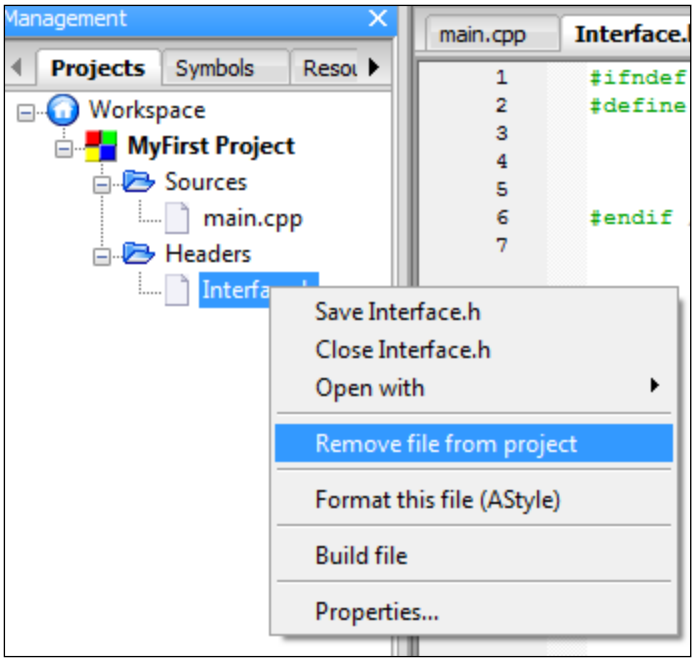
You can now “run” the project by either clicking the *Run* button or hitting *Ctrl-F10*.

Note: the option F9 *(for build and run) combines these commands, and may be more useful in some situations.*

When this program is executed, the call to the function `hello()` in the main application (`main.cpp`) which was defined in `Interface.h`, will cause the implementation of `hello()` in the implementation file (`Interface.cpp`), to be executed.

Removing a file

You can remove a file by simply right-clicking on the file name in the **Projects** tab of the **Management** panel and selecting **Remove file from project**.



*Note: removing a file from a project does **not** physically delete it.*