



# Laboratorio di Tecnologie dell'Informazione

Ing. Marco Bertini  
marco.bertini@unifi.it  
<http://www.micc.unifi.it/bertini/>

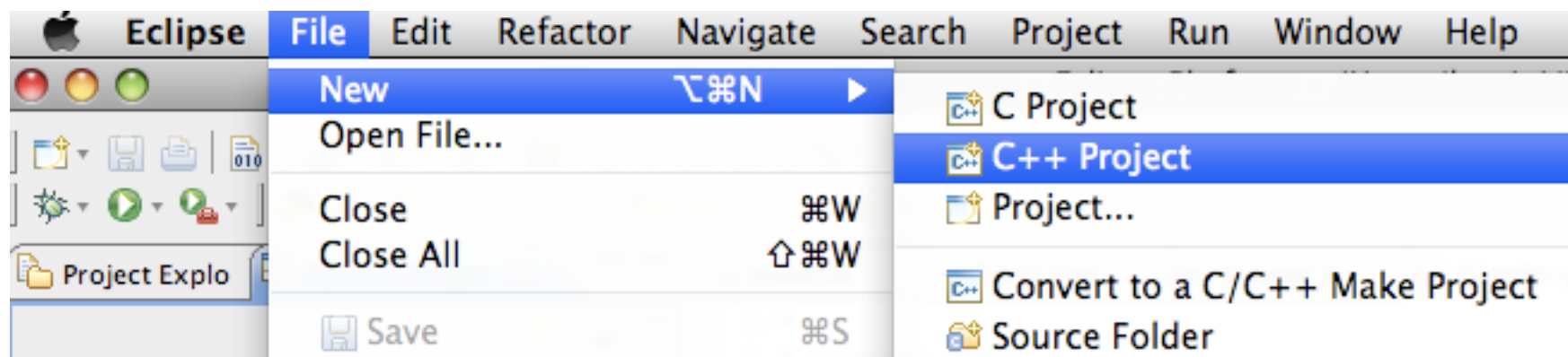


# Building a “Hello world” with Eclipse



# Use the project wizard

- File > New > C++ Project
- Select the “Executable” type: Eclipse will manage automatically the Makefile
- The Makefile project instead requires that the user manages the Makefile (though Eclipse can create a sample one)





# Add a .cpp and .h files

- Add, for example a .h file that contains a function to greet a user, given his name, and add the prototype in the include
- if the include is generated by Eclipse, it will provide automatically the #define guards

```
/*  
 * Greeter.h  
 *  
 * Created on: 26-feb-2009  
 * Author: bertini  
 */  
  
#ifndef GREETER_H_  
#define GREETER_H_  
  
#include <string>  
  
void greet(std::string name);  
  
#endif /* GREETER_H_ */
```



# Compile

- Let's say the code has been written in the .cpp (including all the includes required, e.g. iostream and the greeter.h): compile using Project > Build project
- Check the compile errors (shown in the console panel and in the problems panel)

A screenshot of an IDE's console window. The window title is "C-Build [Test1]". The toolbar shows "Problems", "Tasks", "Console", and "Properties". The console output shows the following text:

```
**** Build of configuration Debug for project Test1 ****  
  
make all  
Building file: ../src/Greeter.cpp  
Invoking: GCC C++ Compiler  
g++ -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/Greeter.d" -MT"src/Greeter.d" -o"src/  
Greeter.o" "../src/Greeter.cpp"  
../src/Greeter.cpp:12: error: variable or field 'greet' declared void  
../src/Greeter.cpp:12: error: 'int greet' redeclared as different kind of symbol  
../src/Greeter.h:13: error: previous declaration of 'void greet(std::string)'  
../src/Greeter.cpp:12: error: 'string' was not declared in this scope  
make: *** [src/Greeter.o] Error 1
```



# Debug

- Don't panic
- Start reading (carefully) the messages from the first to the last. Solve the first errors, perhaps they have an influence on the others.
- In the example the first error is in the .cpp



# Debug - cont.

```
1/*
2 * Greeter.cpp
3 *
4 * Created on: 26-feb-2009
5 * Author: bertini
6 */
7
8#include "Greeter.h"
9
10#include <iostream>
11
12void greet(string name) {
13    std::cout << name << std::endl;
14}
15
```

Problems Tasks Console Properties

C-Build [Test1]

```
**** Build of configuration Debug for project Test1 ****
make all
Building file: ../src/Greeter.cpp
Invoking: GCC C++ Compiler
g++ -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/Greeter.d" -MT"sr
Greeter.o" "../src/Greeter.cpp"
../src/Greeter.cpp:12: error: variable or field 'greet' declared void
../src/Greeter.cpp:12: error: 'int greet' redeclared as different kind of
../src/Greeter.h:13: error: previous declaration of 'void greet(std::strin
../src/Greeter.cpp:12: error: 'string' was not declared in this scope
make: *** [src/Greeter.o] Error 1
```

Eclipse shows where there's a problem

Read the message: the declaration does not match the prototype; the string was not declared



# Debug - cont.

- Correct the error: in this case it was necessary to add `std::` to `string` (we are not using “`using namespace std;`” in this file !)
- Build again to check the correction





The screenshot shows a code editor with four tabs: 'makefile', 'Test1.cpp', 'Greeter.h', and 'Greeter.cpp'. The 'Greeter.cpp' tab is active and shows the following code:

```
1/*
2 * Greeter.cpp
3 *
4 * Created on: 26-feb-2009
5 * Author: bertini
6 */
7
8#include "Greeter.h"
9
10#include <iostream>
11
12void greet(std::string name) {
13    std::cout << name << std::endl;
14}
15
```

A blue circle highlights the `#include <iostream>` line. The editor's status bar at the bottom shows 'C-Build [Test1]' and the following output:

```
Finished building: ../src/Greeter.cpp

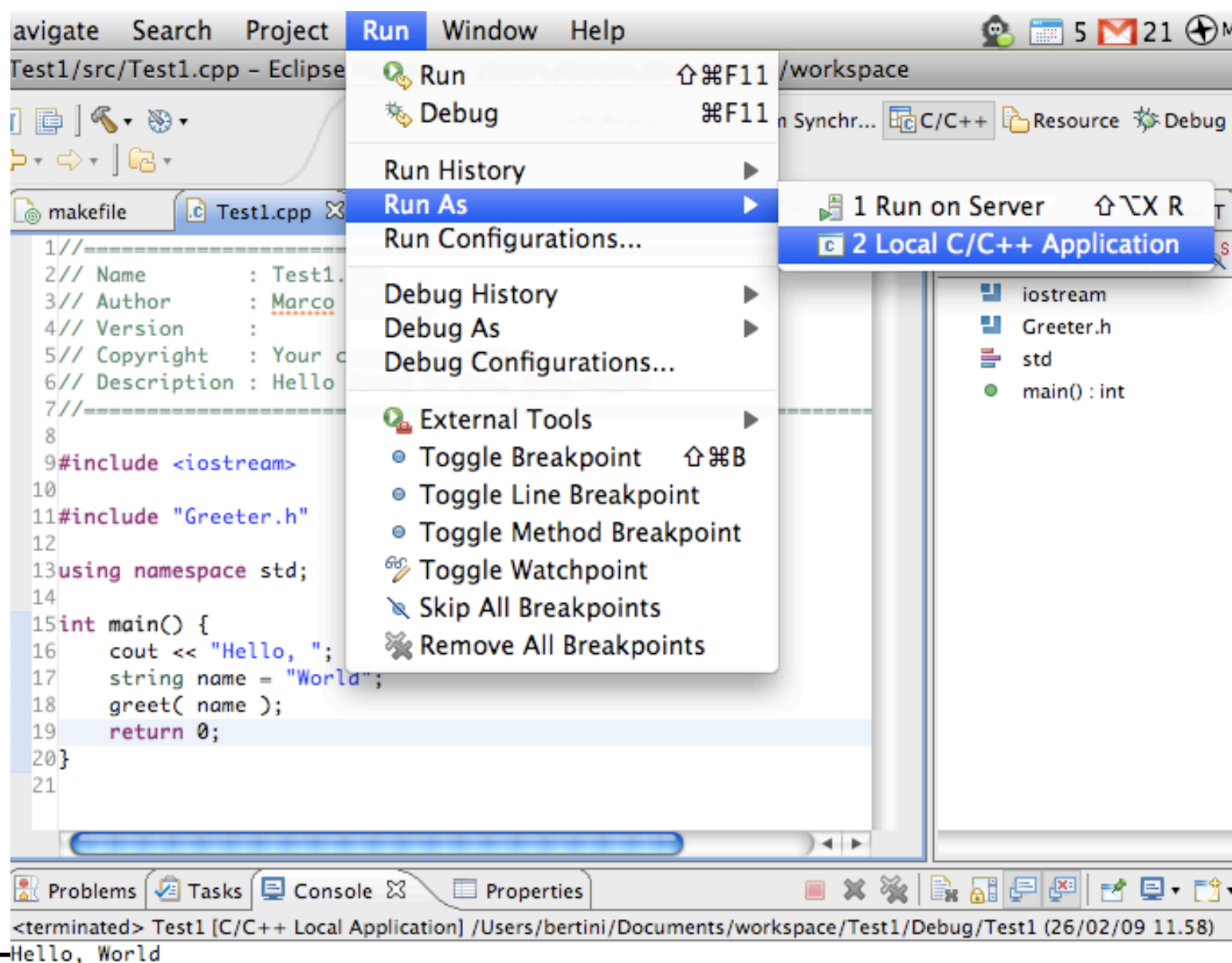
Building file: ../src/Test1.cpp
Invoking: GCC C++ Compiler
g++ -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/Test1.d" -MT"src
../src/Test1.cpp"
Finished building: ../src/Test1.cpp

Building target: Test1
Invoking: MacOS X C++ Linker
g++ -o "Test1" ../src/Greeter.o ../src/Test1.o
Finished building target: Test1
```



# Run the program

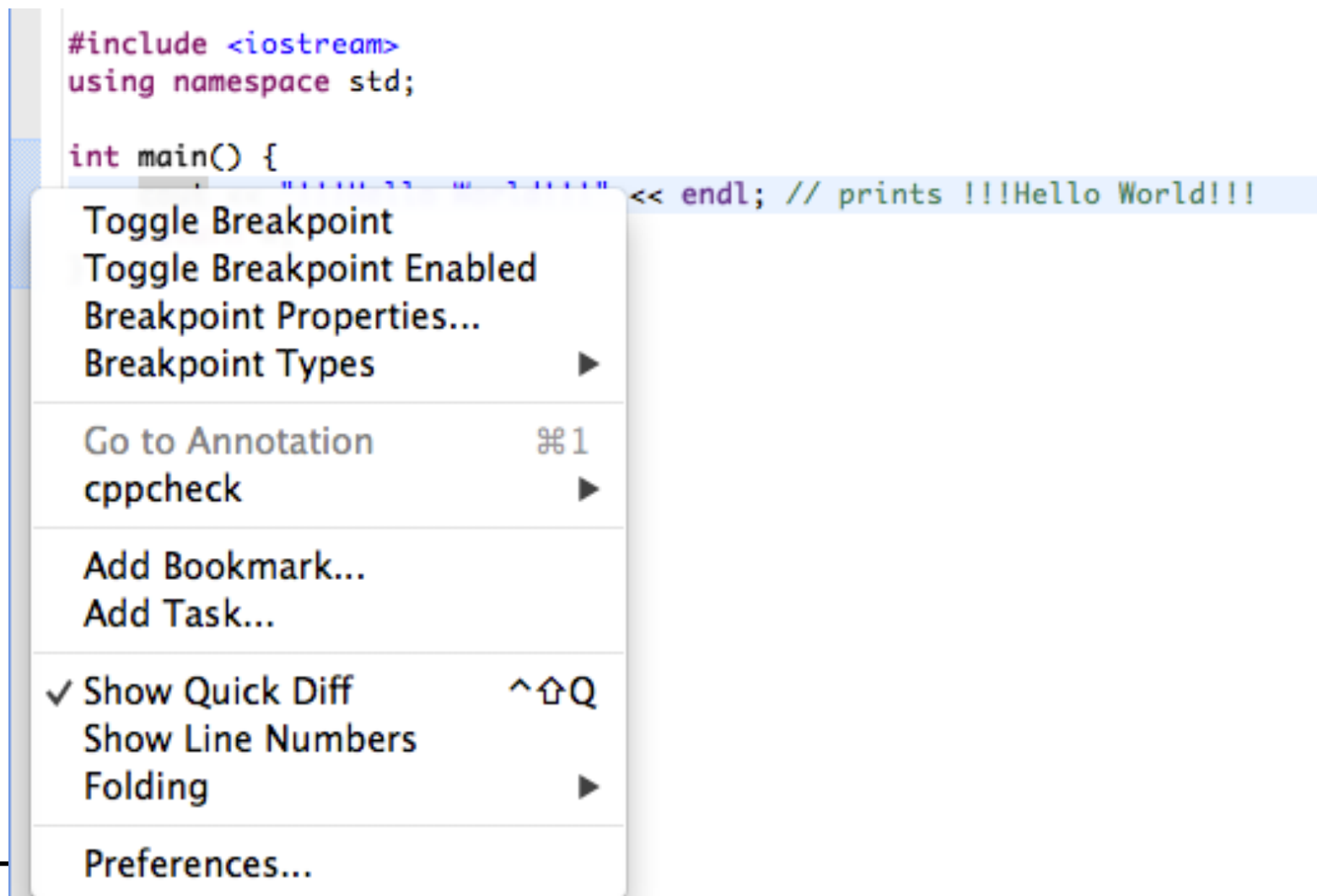
- Use the menu Run > Run as > Local C/C++ application. Later on the program will appear in the Run History





# Debug the program

- In order to debug the program must be compiled so that additional information, useful for the debugger, is added to the files
- Add a breakpoint in Eclipse (right menu on the right side of the line), then execute the program in the debugger (Run > Debug as > Local C/C++ application)





# Debug the program

- In order to debug the program must be compiled so that additional information, useful for the debugger, is added to the files
- Add a breakpoint in Eclipse (right menu on the right side of the line), then execute the program in the debugger (Run > Debug as > Local C/C++ application)

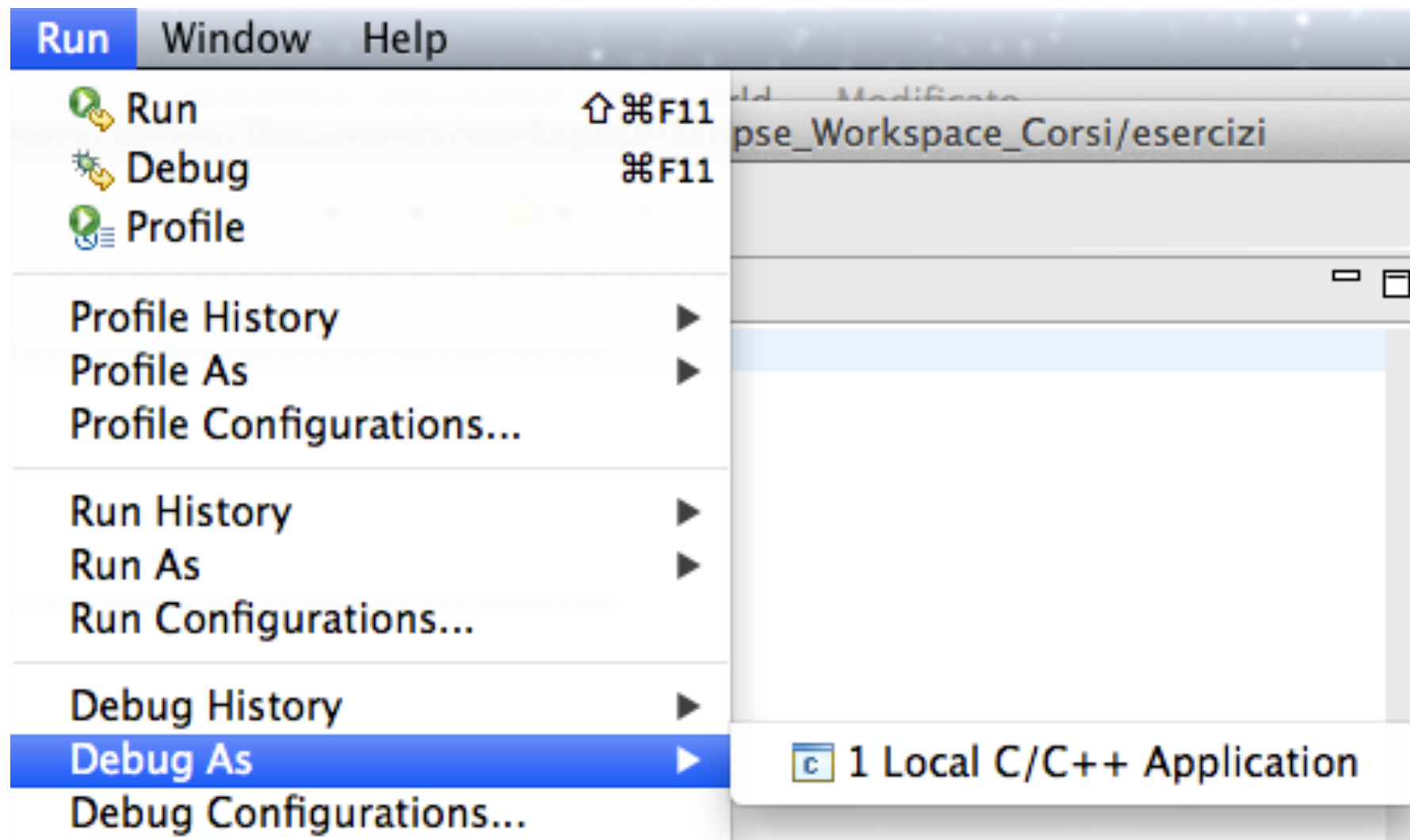
```
#include <iostream>
using namespace std;

int main() {
    cout << "!!!Hello World!!!" << endl; // prints !!!Hello World!!!
    return 0;
}
```



# Debug the program

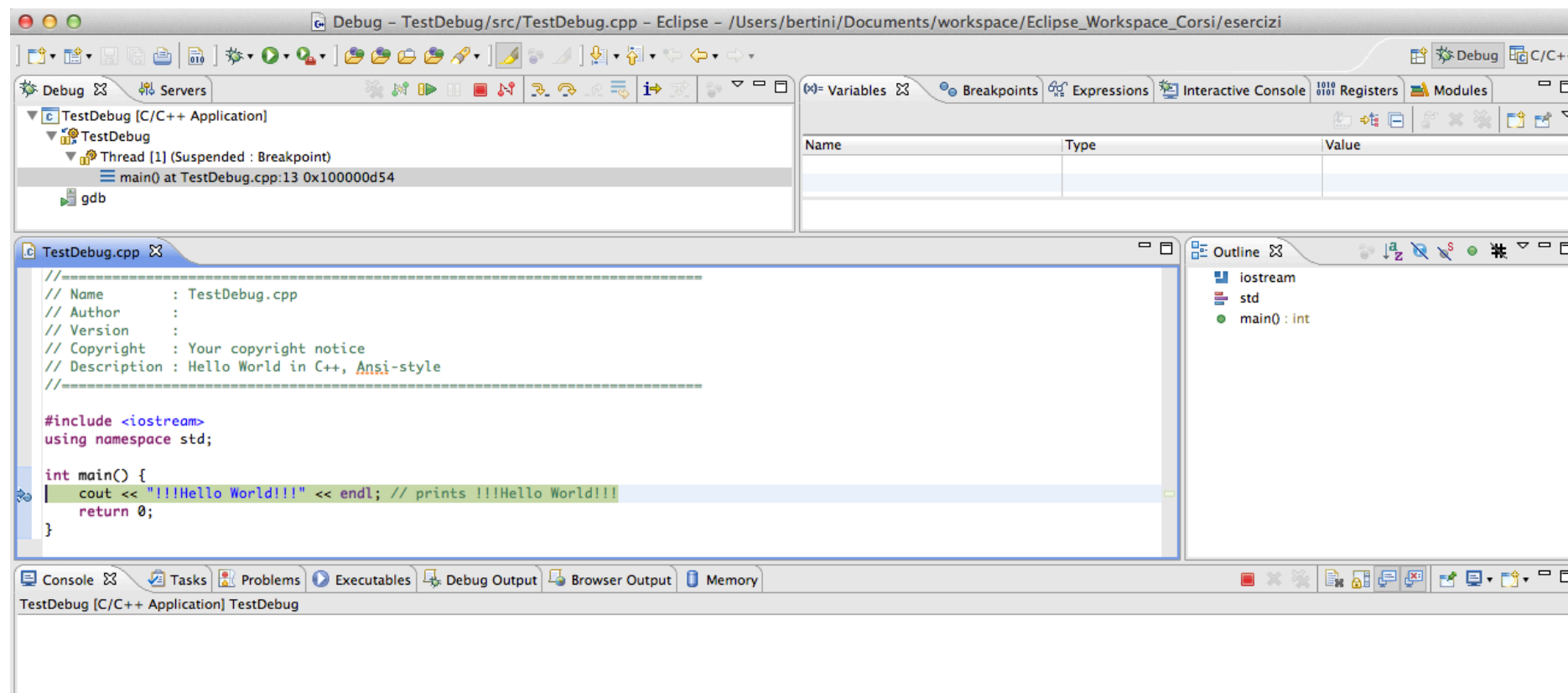
- In order to debug the program must be compiled so that additional information, useful for the debugger, is added to the files
- Add a breakpoint in Eclipse (right menu on the right side of the line), then execute the program in the debugger (Run > Debug as > Local C/C++ application)





# Debug the program

- In order to debug the program must be compiled so that additional information, useful for the debugger, is added to the files
- Add a breakpoint in Eclipse (right menu on the right side of the line), then execute the program in the debugger (Run > Debug as > Local C/C++ application)







# Some style guidelines

- There are a plethora of C++ coding style recommendations, sometimes even contradictory.
- Two very good recommendations:
  1. Any violation to the guidelines is allowed if it enhances readability.
  2. The rules can be violated if there are strong personal objections against them.



# Naming conventions

- Names representing types must be in mixed case starting with upper case: follow this rule when writing classes.
- Variable names must be in mixed case starting with lower case (like Java).
- Names representing methods or functions must be verbs and written in mixed case starting with lower case (like Java).





# Naming conventions - cont.

- Names representing namespaces should be all lowercase.
- All names should be written in English.



# Files

- C++ header files should have the extension `.h` (preferred) or `.hpp`. Source files can have the extension `.c++`, `.C`, `.cc` or `.cpp`.
- A class should be declared in a header file and defined in a source file where the name of the files match the name of the class.
- Header files must contain an include guard.
- Include statements must be located at the top of a file only.