

# Basics of C++

# Introduction

This will be a brief introduction to C++

This workshop is intended for those with zero experience with C++ and minimal to no experience with other languages

We will go over the very basic building blocks of the language that you need to start programming

We will then write a game of hangman together to see what we have learned

# Why C++

C++ gives you control of what your computer is doing

Because of this, C++ is:

- Fast and efficient
- Scalable
- Widely used (a majority of video game engines and desktop applications run on C++)

Arduino's language is based on C++

# Differences between Python and C++

Strict syntax

Must end every line with a semicolon (;)

Must declare variable types

C++ is not space sensitive (uses { } instead)

# Data Types

```
int a = 2;
```

```
double d = 3.14;
```

```
char c = 'w';
```

```
string s = "Hello world";
```

```
bool b = true;
```

# if, else if, else

```
int a = 2;  
if (a == 2) {  
    // Do something  
}
```

```
if (a == 3) {  
    // Do something  
}  
else if (a == 4) {  
    // Do something else  
}  
else {  
    // Otherwise, do this  
}
```

# while loops

```
int i = 0;
```

```
while (i < 5) {  
    i = i + 1;  
}
```

```
// This loops 5 times
```

# do while loops

```
int i = 0;
```

```
do {  
    i = i + 1;  
} while (i < 0);
```

```
// This loops 1 time
```



# for loops

```
for (int i=0; i<5; i++) {  
    // Do something  
}
```

1. `int i=0`
2. `while ( i<5 )`
3. `i++` means `i=i+1`

# Arrays

```
int foo[5] = { 16, 2, 77, 40, 12071 };
```

```
char a[3] = {'a', 'b', 'c'};
```

```
int my_variable[5];
```

```
my_variable[0] = 4;
```

```
// Printing my_variable[0] would print 4
```

```
// Printing my_variable[1] would yield
```

```
// an error
```

Notice:

Arrays start at 0

The last index of foo is  
not foo[5] but foo[4]

# Functions

```
int my_function(int a, int b) {  
    int c = a + b;  
  
    return c;  
}
```

```
void other_function(int a, int b) {  
    a = 3;  
}
```

```
int main {  
    int a = 1;  
    int b = 2;  
  
    int r = my_function(a, b);  
    other_function(2, b);  
  
    return 0;  
}
```

# Function Prototyping

```
int my_function(int a, int b);  
int other_function(int a, int b);
```

```
int main {  
    int a = 1;  
    int b = 2;  
  
    int r = my_function(a, b);  
    other_function(2, b);  
  
    return 0;  
}
```

```
int my_function(int a, int b) {  
}
```

```
void other_function(int a, int b) {  
}
```

# Standard Libraries

```
#include <math.h>
```

```
int x = 1;
```

```
int y = 0;
```

```
y = log2(x);
```

# Input/Output

```
cout << "Hello world";
```

```
int a = 2;
```

```
cout << a << endl;
```

```
cin >> a;
```

# Summary

- A computer needs to know how to handle information given to it, data types let it know how
  - int, float, double, char, bool, void
  - All data is stored in memory
    - Can create fancy and fast memory structures to package data together (arrays, structs, classes)
- Coding is about making logical decisions, control structures help us in making these
  - if, else if, else, switch
- Reusing code saves time and avoids bugs.
  - Functions provide an easy way to bundle code under a label
  - Loops (for, while, do while) let us repeat some code a bunch of times without having to manually type it each time.
  - Libraries let us use code others have already written, no point in reinventing the wheel.
    - The #include directive tells the compiler to fetch another library from somewhere
    - Libraries are subdivided into namespaces to avoid conflicting variable/function names.

# Basic Hello World Program

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Hello world" << endl;
6     return 0;
7 }
```

(Line numbers 1, 2, 4, 5)

1. Libraries used to bundle code.  
#include directive seen on line 1.
2. Namespace for even larger code base, like standard syntax rules, etc.  
i.e. standard cout
4. Main returns 0 means everything was ok.
5. cout prints out with << operator.  
endl is "make a new line here".



# Basic Hello World Program

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Hello world" << endl;
6     return 0;
7 }
```

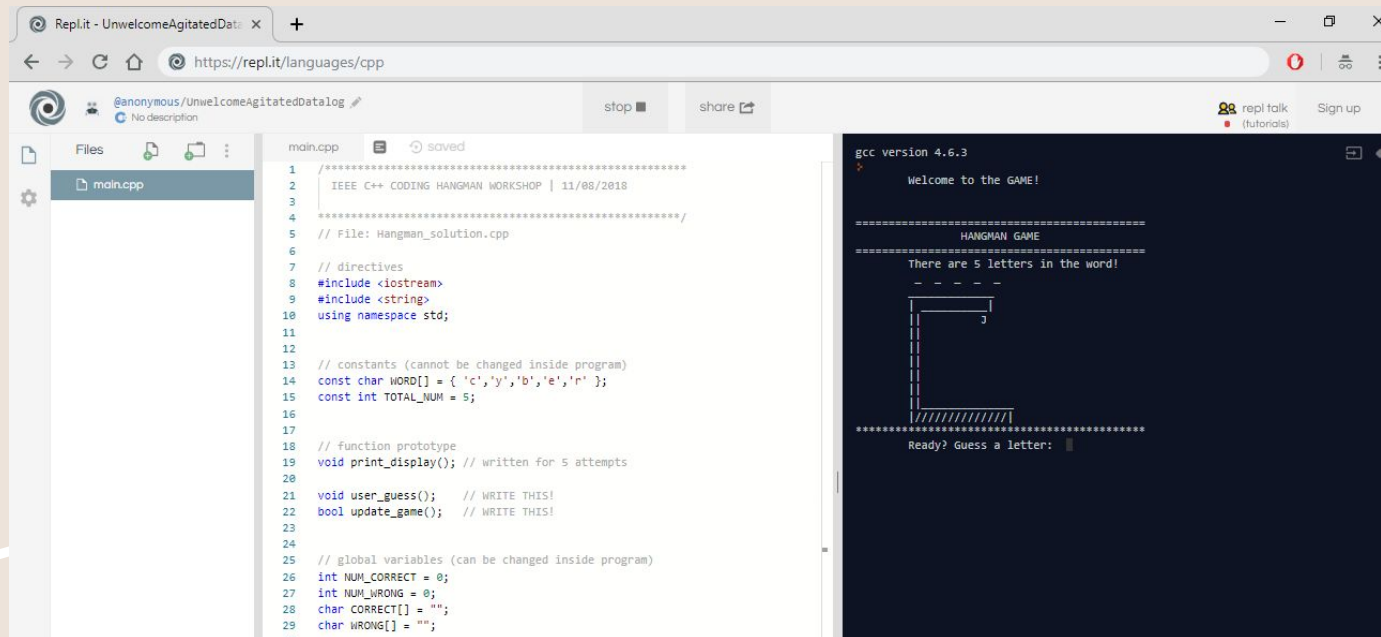
1. Repeating code is a waste of effort and time, instead C++ (and many other languages) bundle code into libraries. To use these libraries, such as the library called `iostream` (used for input and output), we use the `#include` directive seen on line 1.
2. These libraries can be massive so some items are placed into namespaces. This avoids having two items that may conflict (like having two functions named `print`). The stream we will be using, `cout`, and the variable `endl`, are from the `std` namespace. Without line 2 we can still refer to `cout` and `endl` by instead typing `std::cout` and `std::endl` but by saying we are going to be using the namespace `std` we save ourselves some extra typing.

# Basic Hello World Program

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Hello world" << endl;
6     return 0;
7 }
```

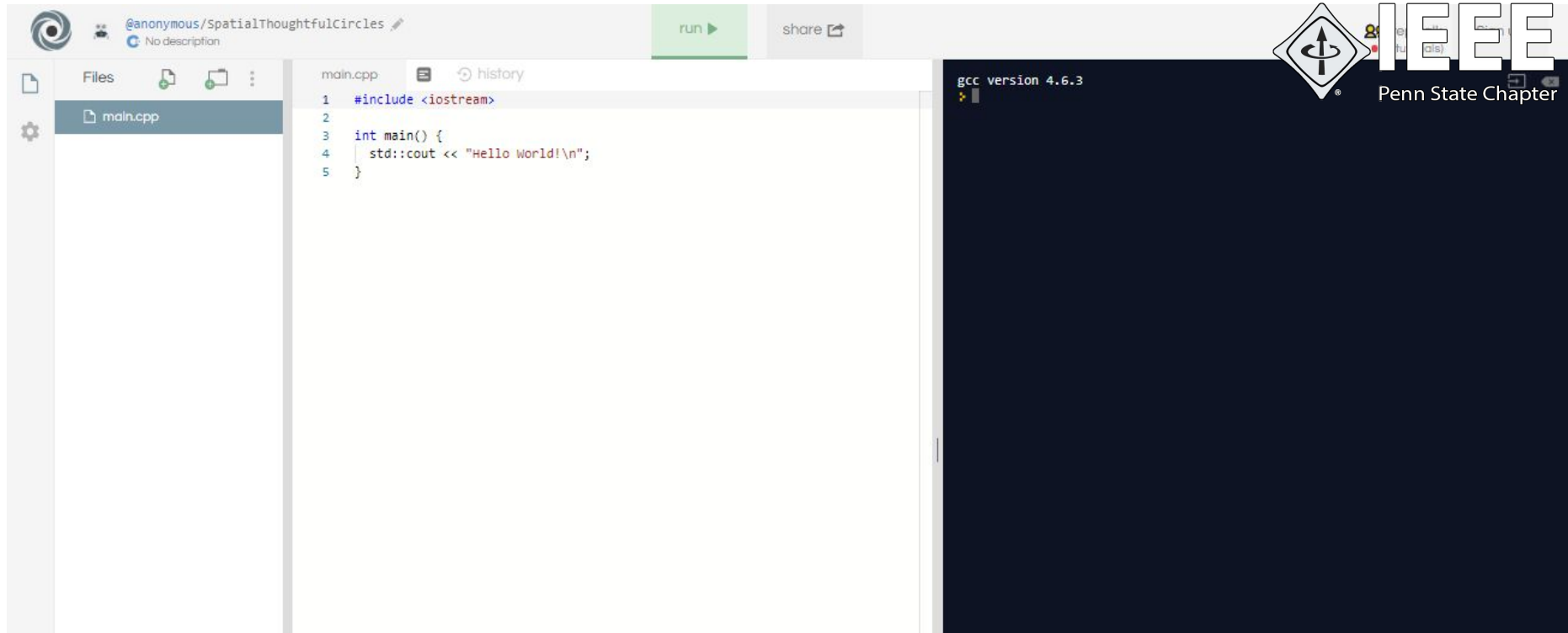
4. When a program is ran 'int main' is the first function to be ran. It returns a number indicating the reason for the program closing. Returning 0 means everything was ok.
5. cout is called a stream, you can feed it in with the << operator. This input is then passed to the standard output (the terminal). endl simply means "make a new line here".

# Let's Get Started !



We'll use this online C++ compiler:

[\*\*repl.it/languages/cpp\*\*](https://repl.it/languages/cpp)



The image shows a web-based C++ IDE interface. At the top, there is a header bar with a logo on the left, a username "@anonymous/SpatialThoughtfulCircles" and a "No description" link, a green "run" button with a play icon, and a grey "share" button with a share icon. On the right side of the header, there is a logo for the IEEE Penn State Chapter. Below the header, the interface is divided into three main sections. On the left is a "Files" sidebar showing a file named "main.cpp". In the center is the code editor, displaying the following C++ code in "main.cpp":

```
1 #include <iostream>
2
3 int main() {
4     std::cout << "Hello World!\n";
5 }
```

On the right is a dark-themed terminal window showing the output of the compilation: "gcc version 4.6.3".

They have fun titles on the top (refresh page to see new names)

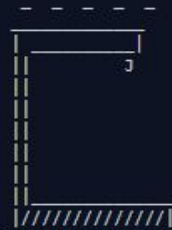
gcc version 4.6.3



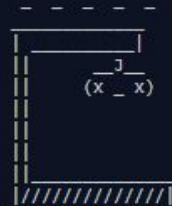
Welcome to the GAME!

HANGMAN GAME

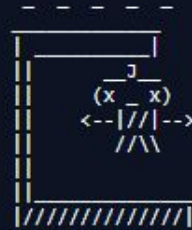
There are 5 letters in the word!



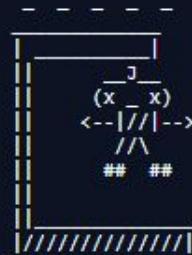
Ready? Guess a letter: p  
SORRY. WRONG GUESS :(



Ready? Guess a letter: o  
SORRY. WRONG GUESS :(



Ready? Guess a letter: n  
SORRY. WRONG GUESS :(



Ready? Guess a letter: m  
SORRY. WRONG GUESS :(

GAME OVER!!! Thanks for playing!  
The word was "JACK"



IEEE

Penn State Chapter

Let's try playing a game of hangman. The completed version :)

# Play one round

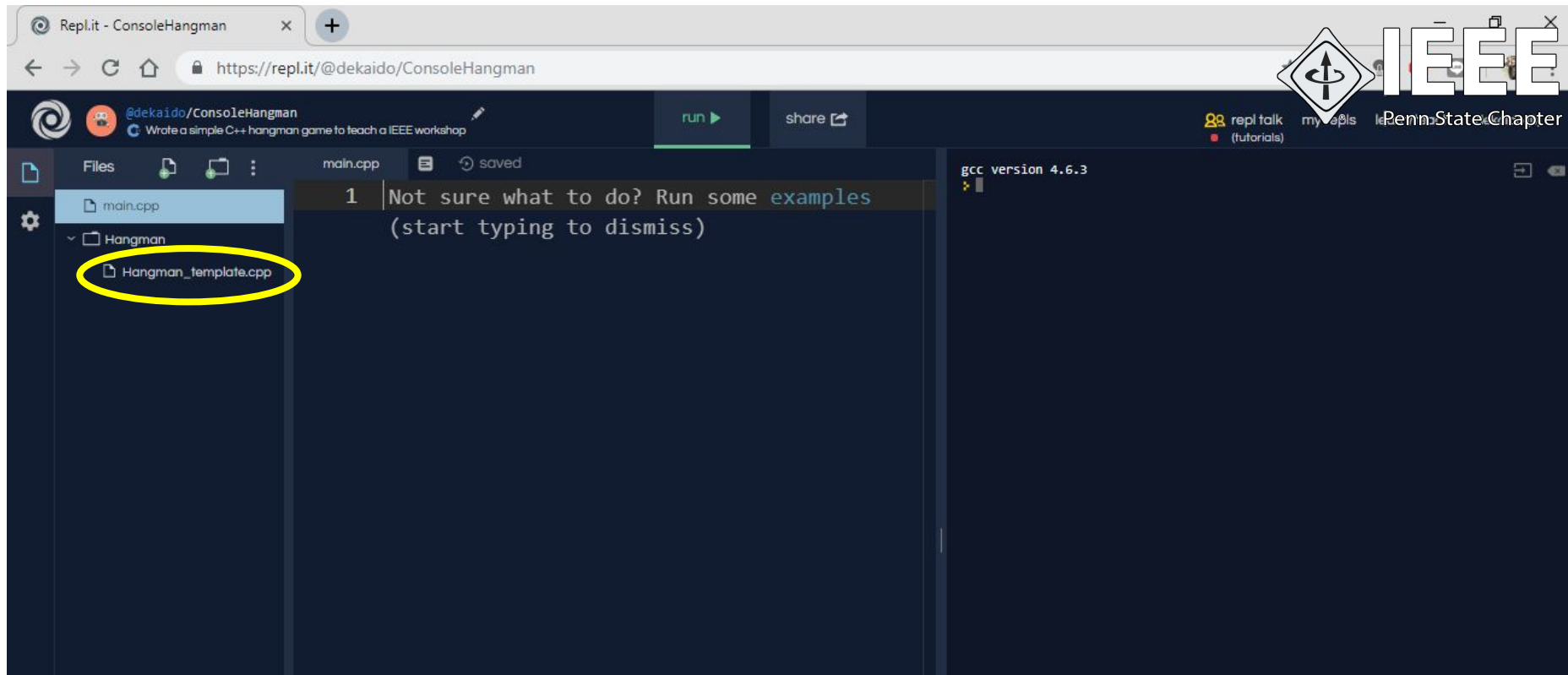
( Test my solution to see how the game should work !)

Now, your turn.

Copy or Download my template to your own!

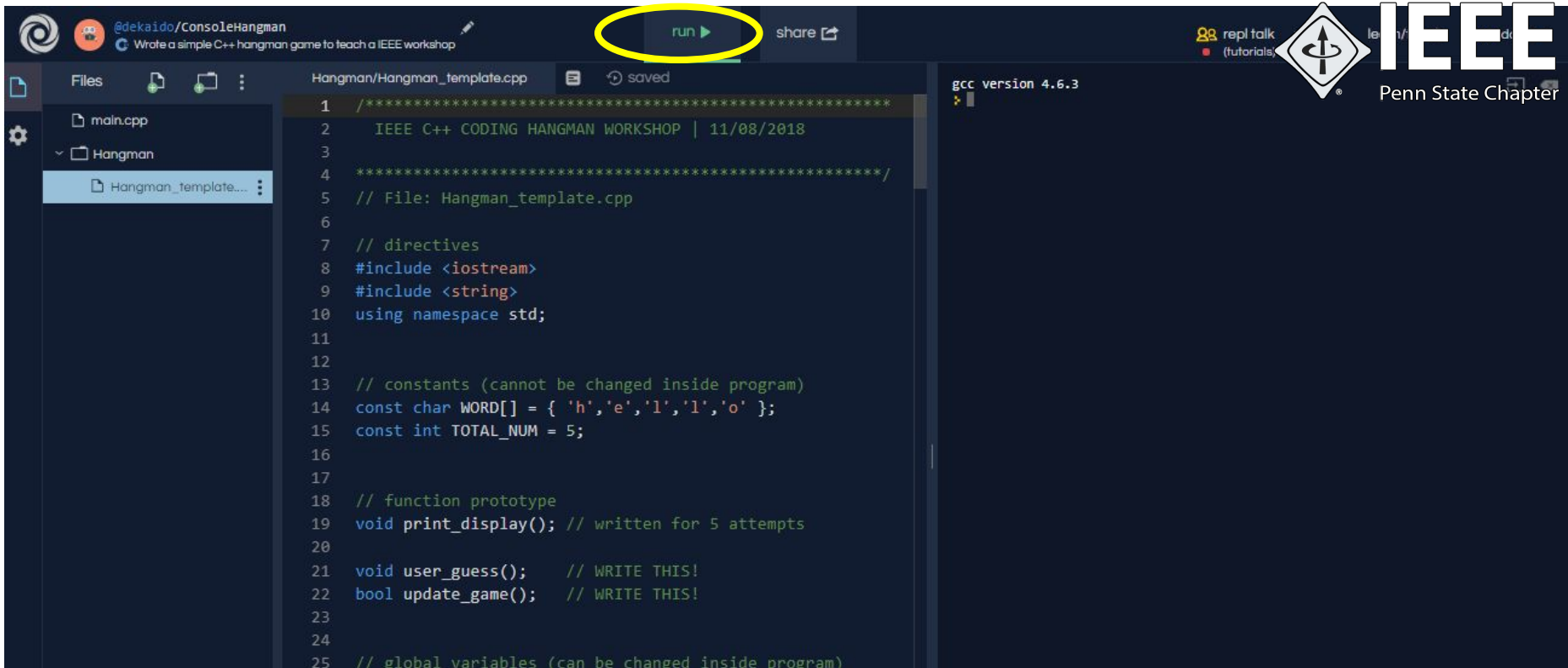
[repl.it/@dekaido/ConsoleHangman](https://repl.it/@dekaido/ConsoleHangman)





The file in the folder “Hangman” . . . named

Hangman\_template.cpp



Try running it

Okay, now I will go over the template :)

Any questions, ask us !