Taming your first program

Due this week

- **Homework 0**
  - Submit zip file on Canvas. Check the due date!
- Participation Quiz
- Start going through the textbook readings and watch the videos
  - Take **Quiz 1**. Check the due date!

Today

1. Analyzing your first program
2. Errors
3. Becoming familiar with your programming environment

Your first program!

Your first program

- The classic first program that everyone writes: Hello World!
  - (yes, everyone who is anyone started with this one)
- Its job is to write the words *Hello World!* on the screen.

```
#include <iostream>
using namespace std;
int main()
{
  cout << "Hello, World!" << endl;
  return 0;
}
```

the #include

- The first line tells the compiler to include a service for “stream input/output”. Later you will learn more about this but, for now, just know it is needed to write on the screen.

```
#include <iostream>

using namespace std;
int main()
{
	cout << "Hello, World!" << endl;
	return 0;
}
```

using namespace std

- The second line tells the compiler to use the “standard namespace”. This is used in conjunction with the <iostream> first line for controlling input and output.

```
#include <iostream>

using namespace std;
int main()
{
	cout << "Hello, World!" << endl;
	return 0;
}
```

int main()

- The next set of code *defines* a ***function,*** named **`main`.**
  - Every C++ program must contain its one `main` function.
  - All function names must be followed by parentheses. In `main`'s case, the parentheses are empty.
- Braces `{ }` must enclose all the code that belongs to `main`. The braces tell the compiler where to start reading the `main` code, and where to finish.

```
#include <iostream>

using namespace std;
int main()
{
	cout << "Hello, World!" << endl;
	return 0;
}
```

`cout` statement

- To show output on the screen, we use **`cout`.**
- What you want seen on the screen is “sent” to the **`cout`** entity using the **`<<`** operator (sometimes called the insertion operator): **`<< "Hello, World!"`**
- **The curious non-word `endl` means end-of-line, which tells the display to move the cursor down to the start of the next line.**

```
#include <iostream>

using namespace std;
int main()
{
	cout << "Hello, World!" << endl;
	return 0;
}
```

`return` statement

- The **`main`** function "returns" an "integer" (that is, a whole number without a fractional part, called **`int`** in C++) with value 0.
- This value indicates that the program finished successfully.

```
#include <iostream>

using namespace std;
int main()
{
	cout << "Hello, World!" << endl;
	return 0;
}
```

Semicolons are Required after Statements

- Each statement in C++ ends in a semicolon;
  - Note that not every line in a program is a statement, so there are no semicolons after the `<iostream>` line and the `main()` line
  - It is a strange idiosyncrasy, but you will get used to it

```
#include <iostream>

using namespace std;
int main()
{
      cout << "Hello, World!" << endl;
      return 0;
}
```

Output Statements and Streaming Operator <<

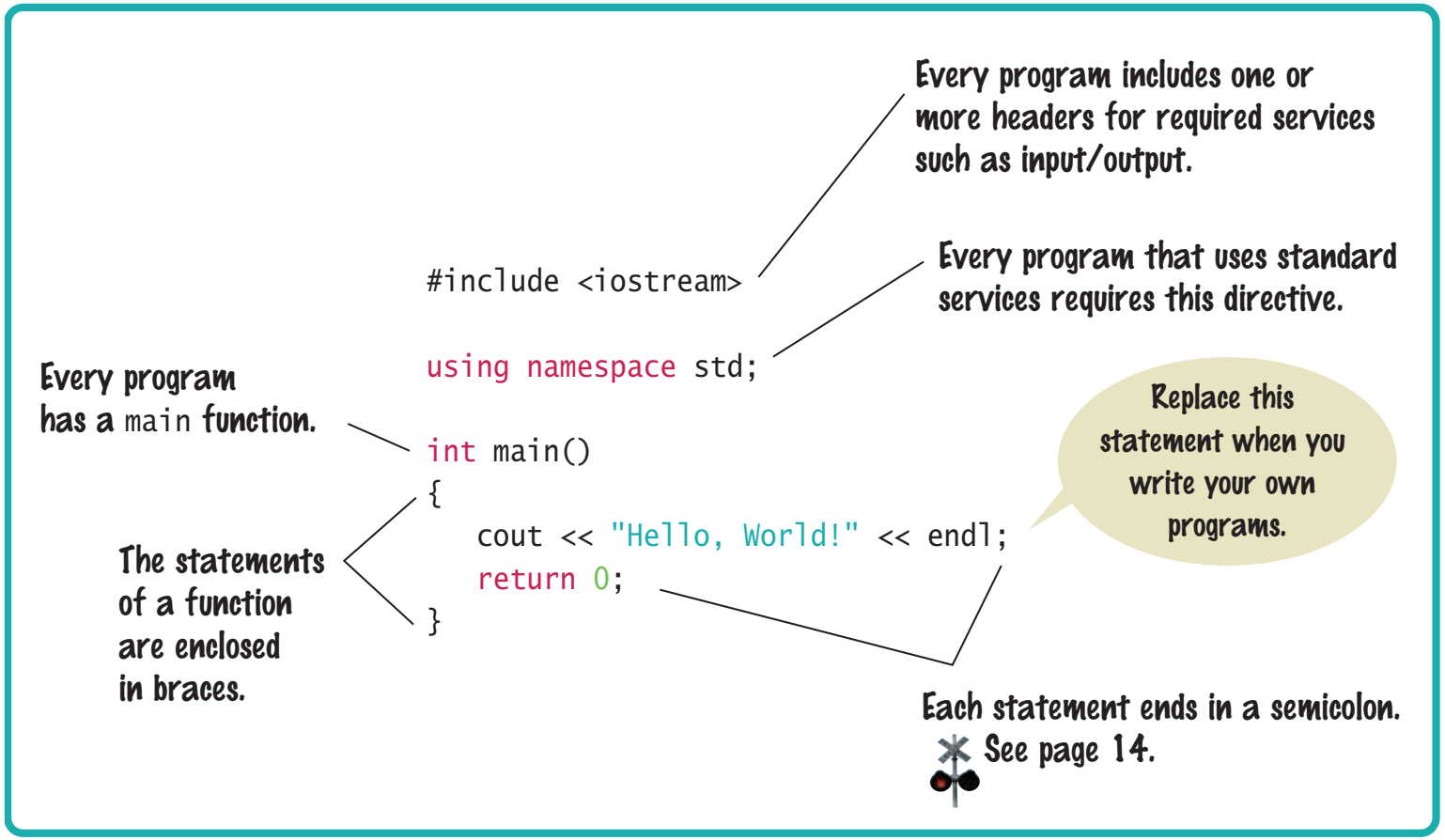
The statement

```
cout << "Hello World!" << endl;
```

is an *output statement*.

- To display values on the screen, you send them to an entity called cout.
  - Which stands for "character output" or "console output".

- The << operator denotes the “send to” command.



Errors!

Common Error – Omitting Semicolons errors

Omitting a semicolon (or two), in this case at the end of the `cout` statement

```
#include <iostream>

using namespace std;
int main()
{
	cout << "Hello, World!" << endl
	return 0;
}
```

Syntax errors

Without that semicolon you actually wrote:

```
cout << "Hello, World!" << endl return 0;
```

which thoroughly confuses the compiler with the `endl` immediately followed by the `return`!

- This is a *compile-time error* or *syntax error*.
- A syntax error is a part of a program that does not conform to the rules of the programming language.

Errors: Misspellings

- Suppose you (accidentally of course) wrote:

```
cot << "Hello World!" << endl;
```

- This will cause a compile-time error and the compiler will complain that it has no clue what you mean by cot.
- The exact wording of the error message is dependent on the compiler, but it might be something like

“Undefined symbol cot” or "Unknown identifier".

How many errors?

- The compiler will not stop compiling, and will most likely list lots and lots of errors that are caused by the first one it encountered.
- You should fix only those error messages that make sense to you, **starting with the first one**, and then recompile (after SAVING, of course!).

Logic Errors

Consider this:

```
cout << "Hollo, World!" << endl;
```

- *Logic errors* or *run-time errors* are errors in a program that compiles (the syntax is correct), but executes without performing the intended action.
- The programmer must thoroughly inspect and test the program to guard against logic errors.
  - *Testing and repairing a program usually takes more time than writing it in the first place, but is essential !*

Errors: Run-Time Exceptions

Some kinds of run-time errors are so severe that they generate an *exception*: a signal from the processor that aborts the program with an error message.

For example, if your program includes the statement

```
cout << 1 / 0;
```

Your program may terminate with a “divide by zero” exception.

Errors: extra or misspelled main() function

- Every C++ program must have one and only one **`main`** function.
- Most C++ programs contain other functions besides **`main`** (more about functions next week).

Errors: C++ is Case Sensitive

C++ is *case sensitive.* Typing:

```
int Main()
```

will compile but will not link.

A link-time error occurs here when the linker cannot find the **`main`** function – because you did not define a function named **`main`**. (**`Main`** is fine as a name but it is not the same as **`main`** and there has to be one **`main`** somewhere.)

If you want to learn more about the build process, read this. The content in this webpage is not a part of the syllabus and will not be on any course related assignments.

Making your Program Readable (by Humans)

C++ has *free-form layout*

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

- *will* compile (but is practically impossible to read)

A good program is readable:

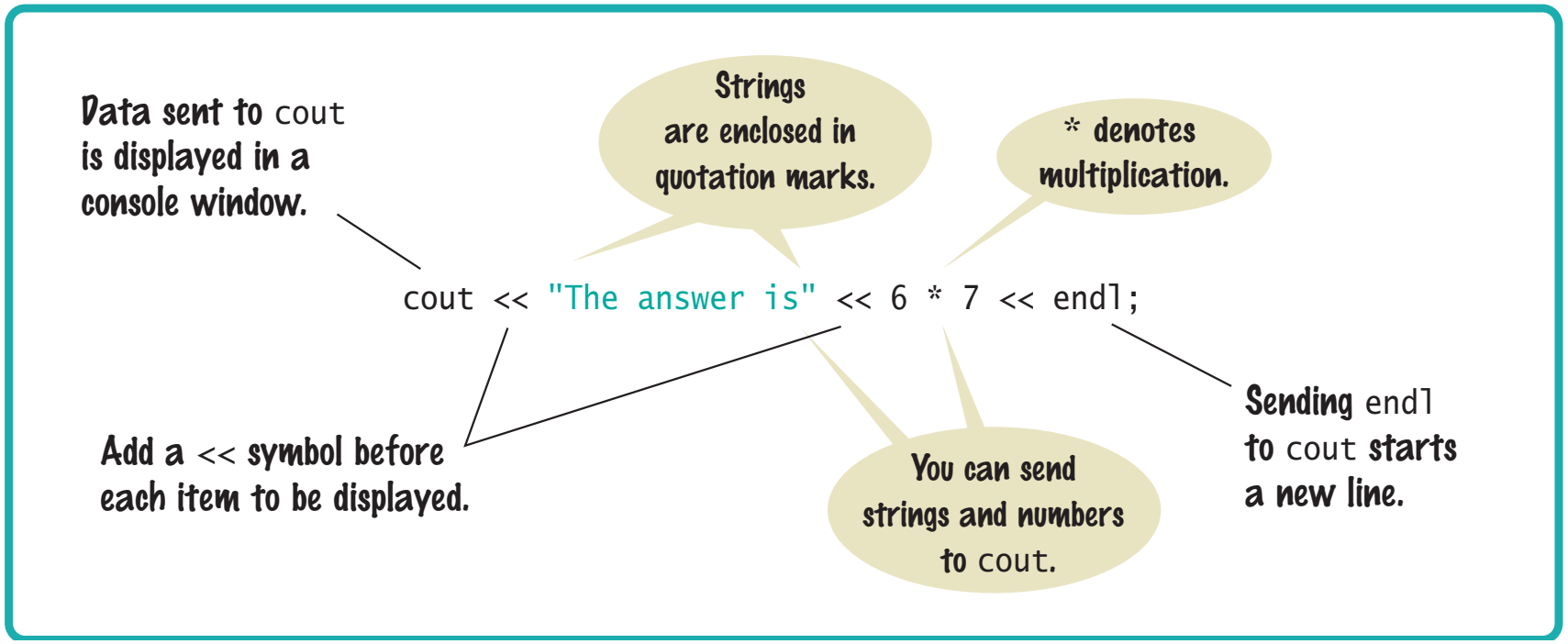
- code spaced across multiple lines, one statement per line
- follows indentation conventions

"Strings" and `endl`

```
cout << "Hello World!" << endl;
```

- **`"Hello World!"`** is called a *`string`*.
- You must put those double-quotes around strings.

- The **`endl`** symbol denotes an *end of line* marker which causes the cursor to move down to the next screen line.



Let’s look at our IDE!

Next time

- Variables and arithmetic