Module 03: Syntax, Literals, Operators

Intro to Computer Science 1 - C++
Professor Scott Frees

Syntax

Syntax is the “rules” of the language.

C++ programs are **collections** of statements, typically containing *expressions*.

Lets focus on each of these elements

Collection of statements - main function

We will talk of functions a lot, but for the next few weeks, we'll just have one… *main*

```
int main() {

   cout << “This is my first C++ course” << endl;
   cout << “I hope it goes well…” << endl;

}
```

The braces form groups of statements
Indentation matters… pay attention!!!!!

Collections of statements - libraries

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

int main() {

   cout << “This is my first C++ course” << endl;
   cout << “I hope it goes well…” << endl;

}
```

C++ comes with build in libraries of code, which you need to include to use many of its features

Statements

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

int main() {

   cout << “This is my first C++ course” << endl;
   cout << “I hope it goes well...” << endl;

}
```

A bit of an odd-ball, ,we’ll get back to this “statement” later in the semester...

Statements end with a semi-colon

Keywords

- Some we've already seen:
- Cannot be used for anything but their intended purpose
- Always lower case
- C++ understands them.
- Some we will see today:
- There will be dozens more…

```
using
namespace
#include
int
return
```

```
float
double
bool
char
sizeof
```

Literals

Strings:	`“This is my first C++ course”`
Integers:	`3, 1009, -42`
Floating-point numbers:	`4.5, 6.9834, -3.145`
Characters:	`‘A’, ‘b’, ‘5’` ← Note - this is not a number!
Booleans:	`true, false`

Operators

For numeric data, we have standard mathematical operators

```
3 + 4

12 - 3

5.3 * 8.9

5 / 2
```

Not the answer you think!

The results of these operations can be printed

```
cout << 3 + 4 << endl;

cout << 12 - 3 << 5.3 * 8.9 << endl;
```

$$\underbrace{\texttt{cout << 12 - 3 << 5.3 * 8.9 << endl;}}_{\text{Doesn't print out very nicely!}}$$

Operators

cout works with a specialized “operator” as well

- << is the “insertion operator”
- Inserts characters into the output stream

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

int main() {
    cout << 5 / 2 << endl;
    cout << 3 + 4 << endl;
    cout << 12 - 3 << 5.3 * 8.9 << endl;
}
```

Lets code this up and run it to see what the issues are

Integer Division and %

- 3 / 2 is the division of two integers, which **must** result in another integer
- The computer doesn't round for you - so the answer is simply 1~~.5~~

We also have a % operator, which gives us the **remainder** of integer division

5 % 2 = 1

51 % 8 = 3 *(6 x 8 = 48 and 51 - 48 = 3)*

Observations

Division of two integers yields an integer!

```
    cout << 5 / 2 << endl;
    cout << 5.0 / 2 << endl;
```

Data types

cout prints exactly what you ask it to - nothing more!

```
cout << 12 - 3 << 5.3 * 8.9 << endl;
cout << 12 - 3 << “  “ << 5.3 * 8.9 << endl;
```

Output formatting

Topics coming up...

Observations

THEY ARE ALL ERRORS
AND **THEY ALL MATTER**!

- Errors come in many flavors:
  - Formatting errors (indentation)
    - Compiler doesn't complain
    - No problem at runtime
    - People complain!
  - Syntax errors (missing a semicolon)
    - Compiler complains - no program!
  - Runtime errors (no space between printed numbers)
    - Compiler doesn't know
    - Program "runs" fine
    - Program is still incorrect.