```
//============================
// Section 5 - Structure of a C++ program
//============================
Keywords:
Part of the vocabulary.
Reserved and cannot be redefined
C++ has about 90
C has 32

Identifiers:
Something that the programmer names, like variable names

Operators:
+,-,-,/
<<  Stream insertion operator
>>  Stream extraction operator
::  Scope resolution operator

Punctuation:
Semicolons, braces, quotes, etc

Syntax:
The structure and the meaning of what you want the compiler to understand

// Preprocessor
What is it?
Processes the source code before the compilers sees it
Strips comments and replaces them with a single space
Looks for directives that start with #
#include    #if #elif   #else   #endif
#ifdef      #ifndef #define #undef  #line
#error      #pragma
The preprocessor does not understand C++
It just follows the directives and gets things ready for the compiler

// Comments
Single line comments start with a //
MultiLine comments are old C style /* */

// The main() function
Every C++ must have exactly one main() somewhere
must be in lowercase letters
OS calls main(){} and the code in the {} executes
When the code gets to the return statement, it sends the value to the OS
Two versions of main:

- this one does not expect any arguments
int main(){
    //code
    return 0;
}
program.exe runs with no arguments

- This one expects some arguments from the OS
- argc stands for argument count - the number of pieces passed in
- The actual arguments are in argv which is an argument vector
- It's basically a bunch of strings
int main(int argc, char *argv[]){
    //code
    return 0;
}

program.exe argument1 argument2
```

```
// Namespaces
Naming conflicts happen when developers name things the same
std is the name for the C++ standard namespace
Third party frameworks have their own namespaces
Use the scope resolution operator to access items in a namespace
Namespace::itemWeWantToUse
Alternatively add a using namespace directive
NOTE: it's poor form to use the std namespace.
Use the scope resolution operator and explicitly target your namespaces

// Qualified namespace variant
using std::cout;
using std::cin;
using std::endl;

This way you can target only the items you want out of the namespace

// Basic input and output
cout, cin, cerr, and clog are objects that represent streams

cout
- standard output stream for the console

cin
- standard input stream from the keyboard

<<
- insertion operator used with output streams

>>
- extration operator used with input streams

// Insert data into the cout stream:
std::cout << "Data is " << variable << std::endl;
Note: Does not automatically add line breaks
Either use "\n" to add a newline character or
std::endl to add a line break and clear the buffer

// Extract data into the cin stream
std::cin >> data >> data2;

- Can fail if the entered data cannot be interpreted
- This will leave the variables with an undetermined value
- The way the information is interpreted is based on the variable typeof
- The characters entered will only be processed when the enter key is pressed
- cin will use whitespace as terminating the values being extracted
- this basically means that you can chain values into the input and read into
- multiple variables from the same line:
value1 value2 value3 // will be read into var1 var2 var3, respectively

characters are read from the keyboard and stored in a buffer (efficiency)
items are read from the buffer and it processes what makes sense for the data type
Whitespace is ignored (except for the purposes of separating items getting read in)

- If it reads an integer and a double and you input 10.5, it will read the 10
- and then the .5 into the double
```