Introduction to C++

Thanks to these websites – go here for good tutorials
https://www.w3schools.com/cpp/default.asp
https://www.learncpp.com/

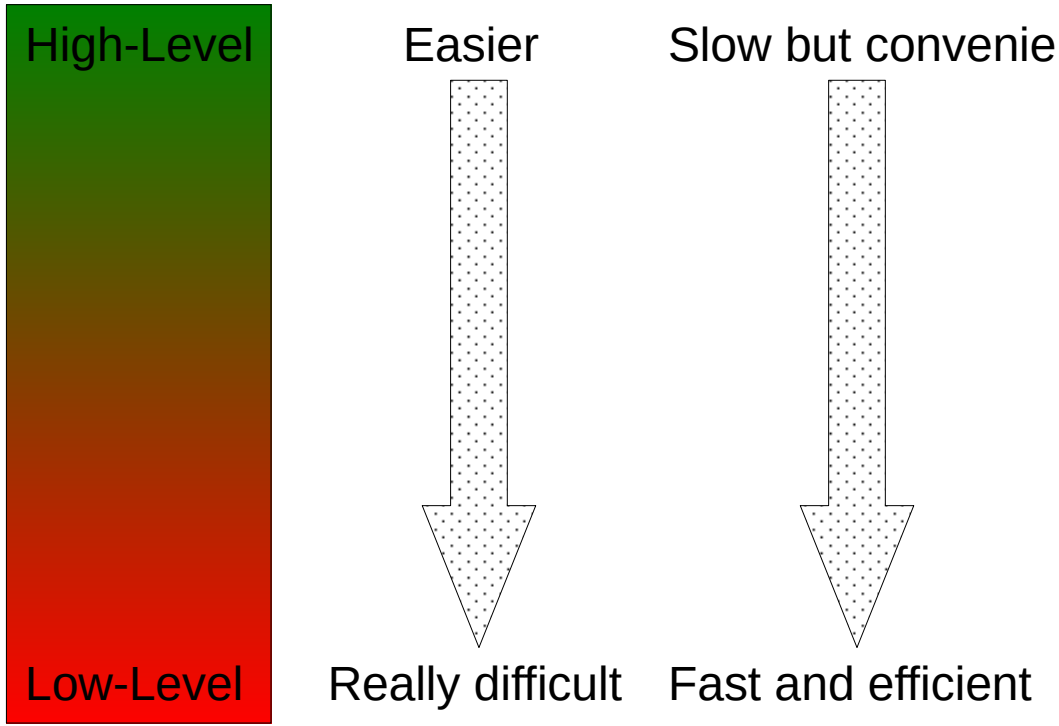
What I use c++ for

- Data analysis
- Automation
  - Simulations; changing run files between runs

I am NOT an object-oriented programmer

There are structures in C++ for this, but I’ve never really used them

Why C++?



Examples:

- Python
- Java
- C++/C
- Machine code

Basics

Google is your friend

- Basic Structure
  - Functions
- Variables
  - Datatypes
- Operators
- Compile
- Strings
- Input & Output
- Conditionals
  - If statements
  - Switch
- Loops
  - While & For
- Arrays

What to do now

- Create a file “<filename>.cpp”
- Open it in your favourite text editor
- Try out stuff that I talk about

Basic Stuff

- Comments:
  - // line comment
  - /* block comment */
- Put ; after EVERY … SINGLE … LINE
- Main function: (every file needs this)

```
int main () { // code goes in here  }
```

Functions

return datatype function name (inputs) {

function definition

}

- Return: int, double, void
- Inputs: datatype & parameter name

If declared in same document as main function, it can be called at any time inside the main function
Though it has to be defined before the main function (above)

Variables

datatype variable name = value;

- You always need to define datatype for variables
- Define variables with or without a value
  - int a = 2 ; int a;
- Define more than one variable on the same line
  - int a, b, c;
- Constants cannot be changed (const int a = 2)

Datatypes

- int = whole numbers (no decimals)
- float = fractional numbers
- double = fractional numbers (but longer this time)
- boolean = true or false
- char = single characters/letters
- string = more than one character (needs #include <string>)

Bonus: “\n” & endl symbolises a new line

Operators

For numbers (int, float, double)

Operator	Name	Description	Example
+	Addition	Add variables	2 + 2 = 4
-	Subtraction	Subtract variables	4 – 2 = 2
*	Multiplication	Multiply variables	2 * 2 = 4
/	Division	Divides variables	4 / 2 = 2
%	Modulus	Returns division remainder	5 % 2 = 1 (5 = 2*2 + 1)
++	Increment	Increase value by 1	2++ = 3
--	Decrement	Decrease value by 1	2-- = 1

Strings

- Add them: + or .append()
  - string full = first + second;
  - string full = first.append(second)
- Get length of string: .lenght() or .size()
- Get to characters in strings:
  - string something = “something”;
  - something[0] = s, something[1] = o, etc.

You cannot add things with different datatypes!!

Indexing from 0!!

Compile

g++ filename.cpp -o executable_filename

- -o: is called a flag
  - You can add other flags with -x (where x can be various letters)
  - o specifies the name of the executable file, if you don’t include this the file will just be called “a.out”

Compile

```
g++ filename.cpp -o executable_filename
```

- You will get errors
  (usually helpful so you can figure out what went wrong)
  - Usually you just forgot to put a ; at the end of a line

Input & Output

```
#include <iostream>
```

- cout: standard output
- cin: standard input

```
using namespace std;
```

(this specifies that the standard library is used, so that you don't have to write std::cout or std::cin every time)

Input & Output

- cout: standard output
  - cout << “text \n”;
    - Prints the word “text” and changes line
  - cout << text << endl;
    - Prints the content of the variable text and changes line
  - cout << “This is the text variable: “ << text;
    - Prints the string followed by the content of the variable text
    - This does not change to a new line, next output will be put on the same line

Input & Output

- cin: standard input

  int variable;

  cin >> variable

  - Program will wait for the user to input something in the command line

Exercise

- Make simple calculator:
  - Get two numbers from the user (just do integers)
    - Explain this using outputs
  - Add numbers together and output them

If … Else if … Else Statements

```
if (condition){
    // code will be executed if this condition is true
}
else if (condition){
    // code will be executed if this condition is true
}
else {
    // code will be executed if none of the conditions are true
}
```

- Use conditional operators
  - (>, <, >=, etc.)
- Can only do if-statement on it’s own

Operators

Operator	Name	Returns true	Returns false
==	Equal to	2 == 2	2 == 1
!=	Not equal to	2 != 1	2 != 2
>	Greater than	2 > 1	1 > 2
<	Less than	1 < 2	2 < 1
>=	Greater than or equal to	1 >= 1	1 >= 2
<=	Less than or equal to	1 <= 1	2 <= 1
&&	Logical AND	Returns true if both statements are true, otherwise returns false	
||	Logical OR	Returns true if one of the statements are true, otherwise returns false	
!	Logical NOT	Reverses result; returns true if result is false, and vice versa	

Switch statements

```
switch (expression){
    case x:
        // code if expression == x
        break;
    case y:
        // code if expression == y
        break;
    default:
        // if none of the cases match
        the expression in
}
```

- Break & Continue:
  - Break: breaks out of current loop/switch
  - Continue: break current iteration of loop

Loops

```
while (condition) {
    // run code while condition is true
}
```

```
do {
    // run code while condition is true
}
while (condition)
```

```
for (statement 1; statement 2;
statement 3) {
    // run code
}
```

```
for ( int i=0; i < 5; i++ ){
    cout << i << endl;
}
```

Arrays

datatype variable_name[length of array];

datatype variable_name[length of array] = { x, y, z }

- Access array element by; variable_name[index]
  - Indexing starts from zero

```
string days[7] = {“Mon”, “Tue”, “Wed”, “Thu”, “Fri”, “Sat”, “Sun”}
cout << days[4] << endl; // this will print out Fri
```

Libraries

- Very different to python, there are some libraries for functions, but mostly you have to write functions yourself
- #include <library_name>
- Now you can just the library's functions as if they were functions you have written in the file

Libraries

Most commonly used libraries;

- Iostream
  (input and output)
- Fstream
  (read & write to files)
- String
  (operations for strings)
- Cmath
  (mathematical operations)
- String
  (operations for strings)

Defining Functions in Separate Files

- Cleaner main function file
- Define and write out functions in a separate file
- Use forward deceleration
  - Declare “empty” function prototype in main file

Defining Functions in Separate Files

Main.cpp

```
int add(int x, int y);
int main() {
    int sum = add(2,2);
    // sum = 2 + 2 = 4
    return 0;
}
```

Add.cpp

```
int add(int x, int y){
    return x + y;
}
```

Header File

- Don’t want to declare functions in the main file
- Move declarations to separate file

- #include “header_file_name.h”

Header File

Main.cpp

```
#include “add.h”
int main() {
    int sum = add(2,2);
    // sum = 2 + 2 = 4
    return 0;
}
```

add.cpp

```
int add(int x, int y){
    return x + y;
}
```

add.h

```
int add(int x, int y);
```

Compile

g++ filename.cpp -o executable_filename

- If your header files are not located in the same directory as your main file
- -I/<path_to_header_file>

Plotting

- Use other programs
- Examples;
  - GnuPlot
  - ROOT
- There are C++ packages/libraries
  - I’ve never used one

How to structure a c++ program?

- Write out a plan for what you need to do
  - Figure out the general structure
- Define the overall variables that you will need (including their datatype)
- Define functions in separate file

General good practise

- Define functions outside main function, call them inside the main function
- Use indentation and write comments!
- Give variables useful names
  - Not too long, but not too short (‘numberOfpeople’ or ‘n’)