Portfolio 1: Intro To C++

Kieran Morris

What is C++?

C++ is an extension of C, one of earlier low level programming languages developed by Bell Labs in the 1970s. Thanks to this C++ also inherits the low level designation, making it considerably faster than more modern high level langages like python. We will see in this portfolio and onwards why C++ is so much more powerful than others, although a good way to remember it is that C++ is very pedantic and will complain at the slightest problem.

C++ is a compiled language rather than an interpreted language, meaning that the code is compiled into machine code before it is run, unfortunately this means an error will not say "on line 283", it will just say

```
ve_iterator<_IteratorL>&)'
 1667 |     operator<(const move_iterator<_Iterator>& __x,
      |     ^~~~~~~~
/usr/include/c++/11/bits/stl_iterator.h:1667:5: note:   template argument deduction/substitution failed
pain.cpp:4:41: note:   'std::basic_ostream<char>' is not derived from 'const std::move_iterator<_Iterat
    4 |     std::cout << "Hello, World!" < std::endl;
      |                                   ^~~~
In file included from /usr/include/c++/11/bits/basic_string.h:48,
                 from /usr/include/c++/11/string:55,
                 from /usr/include/c++/11/bits/locale_classes.h:40,
                 from /usr/include/c++/11/bits/ios_base.h:41,
                 from /usr/include/c++/11/ios:42,
                 from /usr/include/c++/11/ostream:38,
                 from /usr/include/c++/11/iostream:39,
                 from pain.cpp:1:
/usr/include/c++/11/string_view:589:5: note: candidate: 'template<class _CharT, class _Traits> constexpr
  589 |     operator< (basic_string_view<_CharT, _Traits> __x,
      |     ^~~~~~~~
/usr/include/c++/11/string_view:589:5: note:   template argument deduction/substitution failed:
pain.cpp:4:41: note:   'std::basic_ostream<char>' is not derived from 'std::basic_string_view<_CharT, _
    4 |     std::cout << "Hello, World!" < std::endl;
      |                                   ^~~~
In file included from /usr/include/c++/11/bits/basic_string.h:48,
                 from /usr/include/c++/11/string:55,
                 from /usr/include/c++/11/bits/locale_classes.h:40,
                 from /usr/include/c++/11/bits/ios_base.h:41,
                 from /usr/include/c++/11/ios:42,
                 from /usr/include/c++/11/ostream:38,
                 from /usr/include/c++/11/iostream:39,
                 from pain.cpp:1:
/usr/include/c++/11/string_view:595:5: note: candidate: 'template<class _CharT, class _Traits> constexpr
  595 |     operator< (basic_string_view<_CharT, _Traits> __x,
      |     ^~~~~~~~
```

```
/usr/include/c++/11/string_view:595:5: note:   template argument deduction/substitution failed:
pain.cpp:4:41: note:   ‘std::basic_ostream<char>’ is not derived from ‘std::basic_string_view<_CharT, _
    4 |     std::cout << "Hello, World!" < std::endl;
      |                                   ^~~~
In file included from /usr/include/c++/11/bits/basic_string.h:48,
                 from /usr/include/c++/11/string:55,
                 from /usr/include/c++/11/bits/locale_classes.h:40,
                 from /usr/include/c++/11/bits/ios_base.h:41,
                 from /usr/include/c++/11/ios:42,
                 from /usr/include/c++/11/ostream:38,
                 from /usr/include/c++/11/iostream:39,
                 from pain.cpp:1:
/usr/include/c++/11/string_view:602:5: note: candidate: ‘template<class _CharT, class _Traits> constexp
  602 |     operator< (__type_identity_t<basic_string_view<_CharT, _Traits>> __x,
      |     ^~~~~~~~
/usr/include/c++/11/string_view:602:5: note:   template argument deduction/substitution failed:
pain.cpp:4:41: note:   couldn’t deduce template parameter ‘_CharT’
    4 |     std::cout << "Hello, World!" < std::endl;
      |                                   ^~~~
In file included from /usr/include/c++/11/string:55,
                 from /usr/include/c++/11/bits/locale_classes.h:40,
                 from /usr/include/c++/11/bits/ios_base.h:41,
                 from /usr/include/c++/11/ios:42,
                 from /usr/include/c++/11/ostream:38,
                 from /usr/include/c++/11/iostream:39,
                 from pain.cpp:1:
/usr/include/c++/11/bits/basic_string.h:6340:5: note: candidate: ‘template<class _CharT, class _Traits,
 6340 |     operator<(const basic_string<_CharT, _Traits, _Alloc>& __lhs,
      |     ^~~~~~~~
/usr/include/c++/11/bits/basic_string.h:6340:5: note:   template argument deduction/substitution failed
pain.cpp:4:41: note:   ‘std::basic_ostream<char>’ is not derived from ‘const std::__cxx11::basic_string
    4 |     std::cout << "Hello, World!" < std::endl;
      |                                   ^~~~
In file included from /usr/include/c++/11/string:55,
                 from /usr/include/c++/11/bits/locale_classes.h:40,
                 from /usr/include/c++/11/bits/ios_base.h:41,
                 from /usr/include/c++/11/ios:42,
                 from /usr/include/c++/11/ostream:38,
                 from /usr/include/c++/11/iostream:39,
                 from pain.cpp:1:
/usr/include/c++/11/bits/basic_string.h:6353:5: note: candidate: ‘template<class _CharT, class _Traits,
 6353 |     operator<(const basic_string<_CharT, _Traits, _Alloc>& __lhs,
      |     ^~~~~~~~
/usr/include/c++/11/bits/basic_string.h:6353:5: note:   template argument deduction/substitution failed
pain.cpp:4:41: note:   ‘std::basic_ostream<char>’ is not derived from ‘const std::__cxx11::basic_string
    4 |     std::cout << "Hello, World!" < std::endl;
      |                                   ^~~~
In file included from /usr/include/c++/11/string:55,
                 from /usr/include/c++/11/bits/locale_classes.h:40,
                 from /usr/include/c++/11/bits/ios_base.h:41,
                 from /usr/include/c++/11/ios:42,
                 from /usr/include/c++/11/ostream:38,
                 from /usr/include/c++/11/iostream:39,
```

```
                 from pain.cpp:1:
/usr/include/c++/11/bits/basic_string.h:6365:5: note: candidate: ‘template<class _CharT, class _Traits,
 6365 |     operator<(const _CharT* __lhs,
      |     ^~~~~~~~
/usr/include/c++/11/bits/basic_string.h:6365:5: note:   template argument deduction/substitution failed
pain.cpp:4:41: note:   mismatched types ‘const _CharT*’ and ‘std::basic_ostream<char>’
    4 |     std::cout << "Hello, World!" < std::endl;
      |                                   ^~~~
In file included from /usr/include/c++/11/bits/ios_base.h:46,
                 from /usr/include/c++/11/ios:42,
                 from /usr/include/c++/11/ostream:38,
                 from /usr/include/c++/11/iostream:39,
                 from pain.cpp:1:
/usr/include/c++/11/system_error:269:3: note: candidate: ‘bool std::operator<(const std::error_code&, co
  269 |   operator<(const error_code& __lhs, const error_code& __rhs) noexcept
      |   ^~~~~~~~
/usr/include/c++/11/system_error:269:31: note:   no known conversion for argument 1 from ‘std::basic_os
  269 |   operator<(const error_code& __lhs, const error_code& __rhs) noexcept
      |             ~~~~~~~~~~~~~~~~~~^~~~~
/usr/include/c++/11/system_error:398:3: note: candidate: ‘bool std::operator<(const std::error_condition
  398 |   operator<(const error_condition& __lhs,
      |   ^~~~~~~~
/usr/include/c++/11/system_error:398:36: note:   no known conversion for argument 1 from ‘std::basic_os
  398 |   operator<(const error_condition& __lhs,
```

sucks right?

Compiling a C++ Program

By convention we create text files with the extension .cpp, this is the file extension for C++ source code. So you can pop anything you want in there, say in VS code, and try to compile it. Below is an example of C++ code, although this cannot be evaluated since again, C++ is compiled, and will not work in a markdown document.

```cpp
#include <iostream>

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

This file is saved as `pain.cpp`, it was called this because I used it for the nightmare error message above. To compile this file, you need to use a compiler, the most common one is `g++`, which is the GNU C++ compiler. To compile the file, you need to run the following command in the terminal:

```
g++ pain.cpp -o pain
```

This indicates that we wish to compile the file `pain.cpp` with `g++` and output (`-o`) to an executable file called `pain`. We see now the file `pain` in the directory by running the `ls` command.

```
ls
```

```
## pain
## pain.cpp
## Portfolio_1.pdf
## Portfolio_1.rmd
```

Now we can run the file by running the following command:

```
./pain
```

```
## Hello, World!
```

To summarise how many things we're combining, we are using an Rmarkdown document, to access terminal commands, to compile a C++ file, then to run the compiled file. Wack!

Writing a C++ Program

It's all well and good to compile hello world, but that's boring and we need to do better, so lets take a dive into some of the basic properties of C++. If you've ever programmed before, you'll recognise all the basics.

Syntax

The syntax of C++ is pretty grim to be honest, possibly the ugliest programming language I've ever seen. Here are some important points:

- C++ does not use indentation to define code blocks, instead it uses curly braces `{}`.
- Command lines do not end when you press enter, they must end with a semicolon `;`.
- Comments are defined with `//` for single line comments, and `/* */` for multi-line comments.
- Variables must be declared with a type, and can be assigned a value at the same time.
- Variables cannot change type, once a variable is declared as an integer, it will always be an integer, so we cannot say `int x = 5; x = 0.3;`, this is actually one of the main reasons C++ is so fast, as its not doing background work to check the type of a variable at every stage of computation.
- To import libraries, we use `#include <library>`.
- The main function is defined as `int main() { }` and must always be run in C++ code, even if it doesn't do anything.
- You will have to manually define how complex objects like vectors or matrices are outputted, as only the standard data types have defined `print` functionality.

Variables

We have the 4 standard variable types in C++, `int`, `float`, `bool`, and `char`, along with the type `double` which is a float with double precision, honestly just use double. Below are a few examples of how to declare variables in C++ and how to print them into the console.

```cpp
#include <iostream>

int main() {
    int x = 5;
    float y = 0.3;
    double z = 0.3;
    bool a = true;
    char b = 'a';

    std::cout << x << std::endl;
    std::cout << y << std::endl;
    std::cout << z << std::endl;
    std::cout << a << std::endl;
    std::cout << b << std::endl;

    return 0;

}
```

Notice the rubbish output method, which depends on the `iostream` library. We begin with `std::` to indicate that we are using the standard library, and then use `cout` to print to the console. We then use `<<` to append the variable to the output, and `std::endl` to end the line. We actually could have just added `"\n"` in the case of `char` but its good practice to use `std::endl`.

We also have the `auto` assigment, which isn't a variable in of itself, which uses certain conventions when working with variables to save programmers time figuring out what datatype they need.

We can import more expotic variables with libraries, the most important one is probably vectors, which exist in the `<vector>` library, which of course take arbitrary elements and store them together - even other vectors! Below is an example of using vectors in C++:

```cpp
#include <iostream>
#include <vector>

int main() {
    std::vector<int> x = {1, 2, 3, 4, 5};

    for (int i = 0; i < x.size(); i++) {
        std::cout << x[i] << std::endl;
    }

    return 0;

}
```

Notice that we have to define the type of the vector, indicating what type of elements it will store. Above we have created a function which outputs the elements of the vector, since as we said before there is no standard way to print a vector, such as in python.

Conditionals

Conditionals in C++ are pretty standard, we have `if`, `else if`, and `else` statements, along with the standard comparison operators `==`, `!=`, `>`, `<`, `>=`, and `<=`. Below is an example of how to use them:

```cpp
#include <iostream>

int main() {
    int x = 5;

    if (x == 5) {
        std::cout << "x is 5" << std::endl;
    } else if (x > 5) {
        std::cout << "x is greater than 5" << std::endl;
    } else {
        std::cout << "x is less than 5" << std::endl;
    }

    return 0;

}
```

Notice that the `if` and `else` statements do not require a `;` at the end, as they are code blocks rather than a line.

Loops

Again loops are pretty standard, we have `for` and while loops, here's how you use each of them:

```cpp
#include <iostream>

int main() {
    for (int i = 0; i < 5; i++) {
        std::cout << i << std::endl;
    }

    int i = 0;
    while (i < 5) {
        std::cout << i << std::endl;
        i++;
    }

    return 0;

}
```

Notice that C++ begins indexing at 0, so the loop will run from 0 to 4, not 1 to 5, good.

Functions

We've kind fo already seen functions, the `main` function is a function, and again they're pretty standard, they do need to always return something though, even if its just `0`. Below is an example of how to define a function in C++:

```cpp
#include <iostream>

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

int main() {
    int x = 5;
    int y = 3;

    std::cout << add(x, y) << std::endl;

    return 0;

}
```

So in the above code, when we run the program, we are running `main` which then calls the `add` function. Note that we must say what type of output the function returns `int` and also the type of arguments it takes.