Object-Oriented Programming in C++

Pre-Lecture 1: A simple C++ program

Prof Niels Walet (Niels.Walet@manchester.ac.uk)

Room 7.07, Schuster Building

January 4, 2020

Some very basic properties

- C++ Is a compiled language (i.e., there is a stage between writing the code and running it called "compilation");
- No fixed indentation;
- multiple statement per line or multiple lines per statement;
- Each statement ends with a semicolon (;)
- C++ is a strongly typed language, i.e., each variable must be declared to contain a certain type of object (e.g., integers, floats, doubles, strings etc.).
- There are methods to convert (“cast”) between types. Both *explicit* and more dangerous *implicit.* More later!
- Computers use finite storage to store a variable. The typical storage for each variable is specified in a number of bytes: a byte is 8 bits, i.e., a binary number of 8 digits. In decimal this ranges [0..255].

A few variables types

Integer: Normally specified as an `int`. Integers typically require at least 4 bytes of memory space and ranges from -2147483648 to 2147483647. Also see `long int` and `unsigned int`.

Character: a `char` can store one character. Normally requires 1 byte of memory space (and can thus be mapped on an integer 0 to 255).

Boolean: A `bool` stores logical values, either true or false or 0 and 1 (deprecated).

Floating Point: The basic real number type is a `float` for storing "single precision" floating point values. Typically requires 4 bytes of memory space, in which case the precision due to finite storage is about $7-8$ decimal places

Double Floating Point: The more commonly used `double` typically uses 8 bytes of memory space, and has a precision of $14-15$ significant digits.

void: Void means without any value. void datatype represents a valueless entity. Void data type is used for those function which does not returns a value.

size_t: A specific unsigned integer variable always available in C++. Size depends on compiler.

Conversion between variables types

We may want to convert between variables

```
double r; r=1;
```

Here we assign an implicit integer (1) to a double variable: the integer is *cast* to a double (a non-exact number!)

We can also do the following

```
int i{1}; double r; r=static_cast<double>(i);
```

This is called an explicit cast.

What happens if we write

```
double r{1.5}; int i; i=static_cast<int>(r);
```

Well, if you check the variable `i` it is now 1. OK, and now what happens for `r{-1.5}`?

A simple C++ program

Here is a simple example of a C++ program

```
1  // PL1/simple.cpp
2  // A particularly simple example of C++ in action!
3  // Niels Walet, last updated 04/12/2019
4  #include<iostream>
5  int main()
6  {
7    const int current_year{2020}; //Declare and initialise
8    std::cout << "C++ is the best programming language in "<<current_year<<"!"<<std↩
     ::endl;
9    return 0;
10 }
```

Listing 1 : PL1/simple.cpp

which prints the following message to the screen

```
C++ is the best programming language in 2020!
```

A simple C++ program:

the “main” function

Let us have a look at some of the key features.

- The line

```
int main()
```

  defines a special function called `main()`, which according to the language standard *must* return an integer. (This is why we give the function type `int`)
- Some compilers allow for other types; please refrain from doing so.
- The *convention* is that successful execution returns the value 0, see

```
  return 0;
```

- Other numbers are used to denote an error...

A simple C++ program:

comments

- The first line

```
// PL1/simple.cpp
```

  is an example of a C++ comment. Comments of this type do not need closing - they last for one line only.

- C++ comments can also be put at the end of a line

```
  const int current_year{2020}; //Declare and initialise
```

- What is called "C-style comments" (i.e., inherited from the C language)

```
/*...*/
```

  can also be used when detailed (long) comments are required (and thus need to be spread over multiple lines):

```
/* This is an example of a traditional C comment.
   As you can see it needs to be opened and closed
   but can be spread over several lines as is done here
*/
```

A simple C++ program: standard headers

standard headers

- The next line

```
#include<iostream>
```

  is usually present in a C++ program that does input and/or output. It allows functions and variables from the C++ I/O library (part of the C++ standard library) to be used.
- We use this in the program to print our message to the screen

```
 std::cout << "C++ is the best programming language in "<<current_year<<"!"<<↩
std::endl;
```

  using the variables `cout` and `endl`, and the operation `<<`
- For completeness: Alternative C-style/python style output is also available

  But avoid this like the plague in this course!

```
#include<cstdio>
printf("C++ is the best programming language in %d!\n",year);
```

A simple C++ program:

the standard namespace

- Another feature of C++ is also used, a **namespace**, in this case the standard namespace `std`
- A namespace is a container (accessed through its name) within which declarations of variables, arrays, functions, classes etc. can all be placed. We can use the *same* variable name in different namespaces. This avoids name clashing (important for large programs).
- The above namespace is for the C++ standard library (where `cout` and `endl` reside).
- Usually namespaces are specified when referring to an object using the name resolution operator `::`.
- We shall **not** use

```
using namespace std;
```

  which allows us to use cout etc. without a prefix.
- We will learn more about namespaces later in the course.

A simple C++ program:

constants

The line

```
  const int current_year{2020}; //Declare and initialise
```

uses two useful C++ features.

- The first is the `const` qualifier which forbids the value of `year` to be changed once initialised.
- You may see in existing codes, but must never use, the following technique

```
#define year 2020
```

This uses the “Pre-processor”. The modern method (using `const`) is safer because it specifies the type of the constant (instead of relying on the pre-processor to get it right). This feature becomes even more important when more complicated objects are constructed (see later).

A simple C++ program:

initialisation

- We have also used a specific way to initialize the constant:

```
const int current_year{2020}; //Declare and initialise
```

(This is the universal brace initialization, as introduced in C++11)

- The same method applies for (non-constant) variables, e.g.

```
int a{0}; // declare a new integer and initialize to zero
```

- But we can also use the C++03 form

```
int a(0); // declare a new integer and initialize to zero
```

- The old C-style initialisation is also allowed

```
int a=0; // declare a new integer and initialize to zero
```

Please stick to the first form! We will discuss some of the differences later in the course.

A simple C++ program:

declaring variables

- In C++ declarations can appear anywhere; the only constraint is that they must occur before the variable or function is used, e.g.,

```
1  // PL1/midstream.cpp
2  // An example of decaring variables "just in time"
3  // Niels Walet, last updated 04/12/2019
4  #include<iostream>
5  int main()  {
6    const int current_year{2020};
7    std::cout<<"C++ is the best programming language in "
8         <<current_year<<"!"<<std::endl;
9    const double solar_mass_in_kg{1.989e30}; // I only need to declare this here↩
     !
10   std::cout<<"Did you know that the mass of the sun is "
11        <<solar_mass_in_kg<<" kg?"<<std::endl;
12   return 0;
13 }
```

Listing 2 : PL1/midstream.cpp

gives the output

```
C++ is the best programming language in 2020!
Did you know that the mass of the sun is 1.989e+30 kg?
```

- We only declare and initialize `msun_in_kg` when we need it.

A simple C++ program:

standard I/O streams

We need to understand a little more about I/O to write effective codes.

- We encountered the C++ method for printing output (using `cout`) above. C++ interfaces with I/O devices using a model, called streams.
- The stream connected to the standard output (in our case usually a text window on your screen) is called `cout`. The stream connected to the standard input (usually the keyboard) is called `cin`.
- The streams (and thus the devices connected to the stream) are accessed through two new operators:
  - The insertion operator, <<, *inserts* data to a stream (device)
  - The extraction operator, >>, *extracts* data from a stream (device)
- We will learn a lot more about other streams (files and strings) in the next 2 lectures.

A simple C++ program:

standard I/O streams examples

- Output to the screen:

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

  inserts the string `Hello!` to the stream `cout`, followed by a new line. Equally

```
cout<<"Hello!\n";
```

  produces the same result.

- Input from the keyboard:

```
int year; cin>>year;
```

  extracts a value from `cin` and stores it in `year`, declared with type `int`.

- The `>>` operation can sometimes leave a newline character - to ignore this (e.g. when mixing with `getline`) one often adds the line

```
cin.ignore();
```

  after every `cin>>` statement. (Only really necessary in complex inputs.)

A simple C++ program:

standard I/O streams: errors

- We can also check for bad input

```
4  #include<iostream>
5  int main()
6  {
7    int any_year;
8    std::cout << "Enter a year: ";
9    std::cin >> any_year;
10   // Check input is valid
11   while(std::cin.fail()) {
12     std::cout <<"Sorry, your input was not valid, please enter a year: ";
13     // Clear fail bit and ignore bad input
14     std::cin.clear();
15     std::cin.ignore();
16     std::cin >> any_year;
17   }
18   std::cout<<"C++ is the best programming language in "<<any_year<<"!"<<std::↩
     endl;
19 }
```

Listing 3 : selection of PL1/chkinput.cpp

A simple C++ program:

standard I/O streams: errors

- This outputs (with input displayed)

```
Enter a year: MMXX
Sorry, your input was not valid, please enter a year: Sorry, your input was ↩
 not valid, please enter a year: Sorry, your input was not valid, please ↩
 enter a year: Sorry, your input was not valid, please enter a year: 2020
C++ is the best programming language in 2020!
```

- Can you explain why we get *four* error lines?

A simple C++ program:

concluding remarks

- We have looked at a few simple features of C++
- We have concentrated on a the basic features required for writing simple codes
  - Use of I/O streams C++
  - Standard headers
  - the standard namespace `std`
  - constants
  - declaration and initialisation
  - casts
- I suggest you look at the examples (they are all on the course site) and make some changes!