2811 Lab 1: Introduction to C++
Rafael Kuffner dos Anjos

Goals for this lab:

- Understand why we're learning C++
- To introduce you to the main features of C++, highlighting how it differs from C and Java
- Introduce the Qt Creator *Integrated Development Environment* (IDE)
- Complete the numbered exercises that appear throughout this document

*Please perform all **numbered exercises** or you will not learn the skills needed for the courseworks. You should read and understand the text between the exercises, but don't need to do anything.*

Exercise 0: Installing Qt Creator

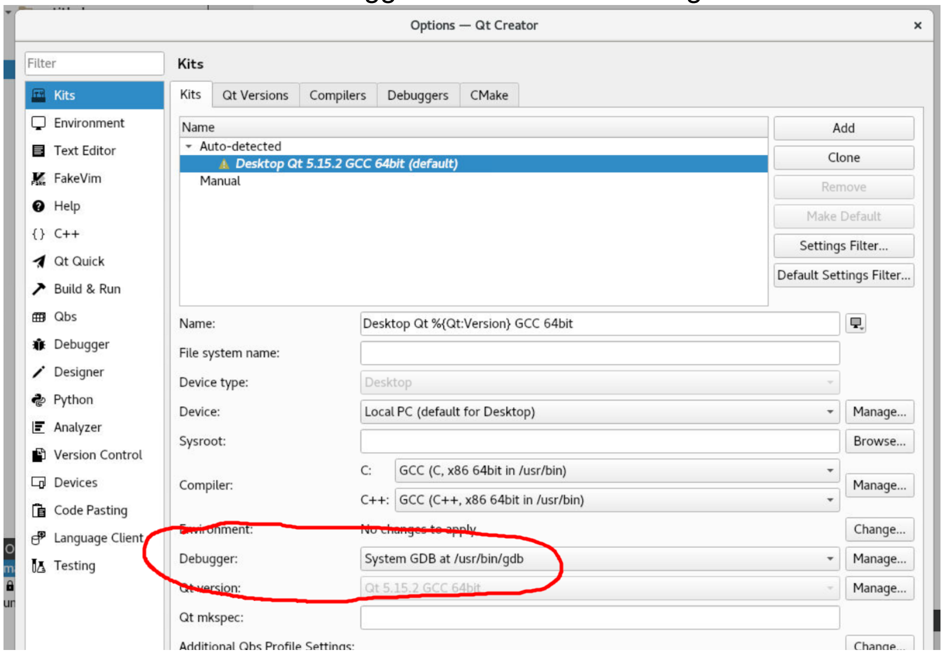
Qt is available in the labs and we will use that version (C++ and Qt 5.15.2) to grade your assignments. You can also install the latest Qt and QtCreator on your local PC. We will install Qt for "Open Source Use" because it is free. (There is also a proprietary license if you want to release your software and keep your source code secret.)

- Labs:
  - Log into the computer using your username and password
  - Open a terminal by right-clicking on the desktop and selecting *"Open Terminal"*
  - to start QtCreator type the following commands followed by enter
    - `module load legacy-eng`
    - `module add qt/5.15.2`
    - `module add gcc`
    - `qtcreator`
- Windows 10:
  - download link for Qt Creator
  - Instructions from Qt if you get stuck
- MacOS:
  - Install XCode before Qt Creator.
  - download link for Qt Creator
  - Youtube video (sorry no recent video)
  - Instructions from Qt if you get stuck.
- Other linux:
  - download link for Qt Creator
  - Instructions from Qt for other distros/if you get stuck.

- Faculty VPN:
    - Enable a university VPN
    - Log into a feng-linux.
    - Follow instructions for "Labs" above.

Common problems:

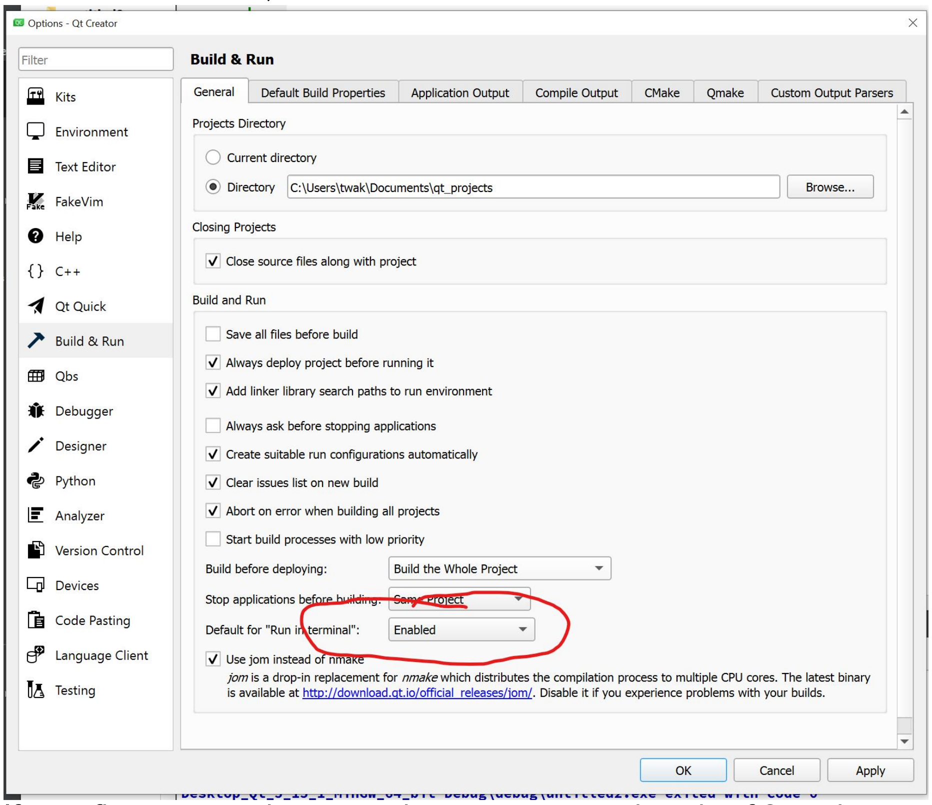
- If you get an "Unable to create a debugging engine" on Centos
    - Go Tools → Options → Kits
    - Select "Desktop Qt 5.15.2…"
    - Scroll down to find the Debugger options
    - Set this to the suggested value "/user/bin/gdb"



Name
Auto-detected
Desktop Qt 5.15.2 GCC 64bit (default)
Manual

Name: Desktop Qt %{Qt:Version} GCC 64bit
File system name:
Device type: Desktop
Device: Local PC (default for Desktop)
Sysroot:
Compiler: C: GCC (C, x86 64bit in /usr/bin)
C++: GCC (C++, x86 64bit in /usr/bin)
Environment: No changes to apply.
Debugger: System GDB at /usr/bin/gdb
Qt version: Qt 5.15.2 GCC 64bit
Qt mkspec:
Additional Qbs Profile Settings:

- If you do not see a console appear when you run your program, or you have problems typing into the console (reading with `cin`):
    - try going to the "Tools" menu, "Options…", then the "Build and Run" section, and set "Default for Run in terminal" to "Enabled":



Projects Directory
○ Current directory
● Directory C:\Users\twak\Documents\qt_projects

Closing Projects
✓ Close source files along with project

Build and Run
☐ Save all files before build
✓ Always deploy project before running it
✓ Add linker library search paths to run environment
☐ Always ask before stopping applications
✓ Create suitable run configurations automatically
✓ Clear issues list on new build
✓ Abort on error when building all projects
☐ Start build processes with low priority
Build before deploying: Build the Whole Project
Stop applications before building: Same Project
Default for "Run in terminal": Enabled
✓ Use jom instead of nmake
jom is a drop-in replacement for *nmake* which distributes the compilation process to multiple CPU cores. The latest binary is available at http://download.qt.io/official_releases/jom/. Disable it if you experience problems with your builds.

- If your first program is not running, you get an exit code of 2, and your username or path contains special (non-ASCII) characters, and the "Console Output" tab contains an error such as "cannot find .pro file":

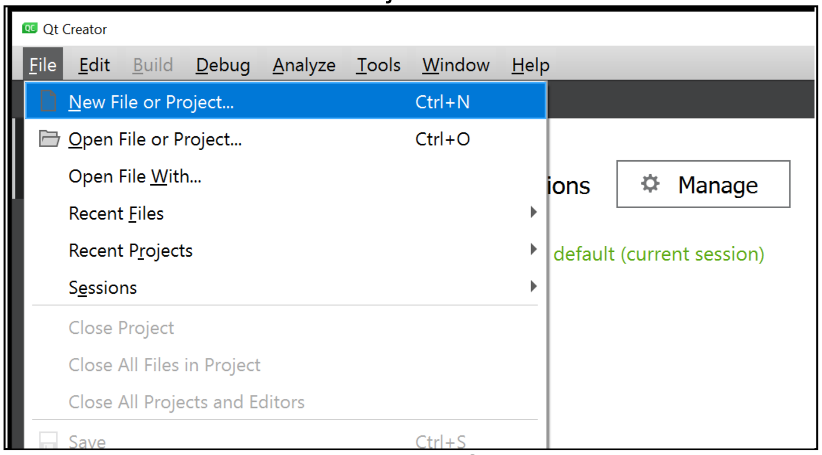
- try removing the special characters.

Compiling your first C++ program in Qt Creator

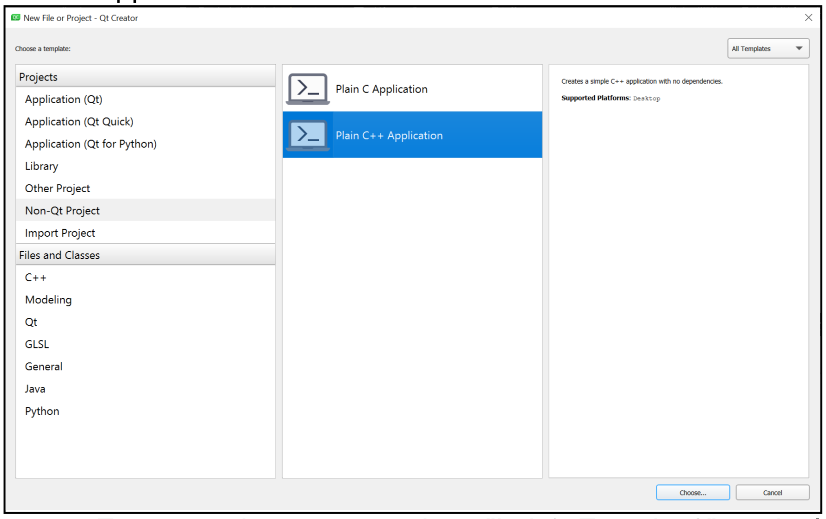
Qt Creator is an IDE for programming C++. It is available for most desktop platforms and is released under the GPL licence. We use it here because it is easy to install and integrates well with the GUI toolkit Qt.

Exercise 1: Follow these instructions.

1. Start Qt Creator
2. Create a new console project.
    - File → New File of Project...


Qt Creator
File
Edit
Build
Debug
Analyze
Tools
Window
Help
New File or Project... Ctrl+N
Open File or Project... Ctrl+O
Open File With...
Recent Files
Recent Projects
Sessions
Close Project
Close All Files in Project
Close All Projects and Editors
Save Ctrl+S
ions
Manage
default (current session)

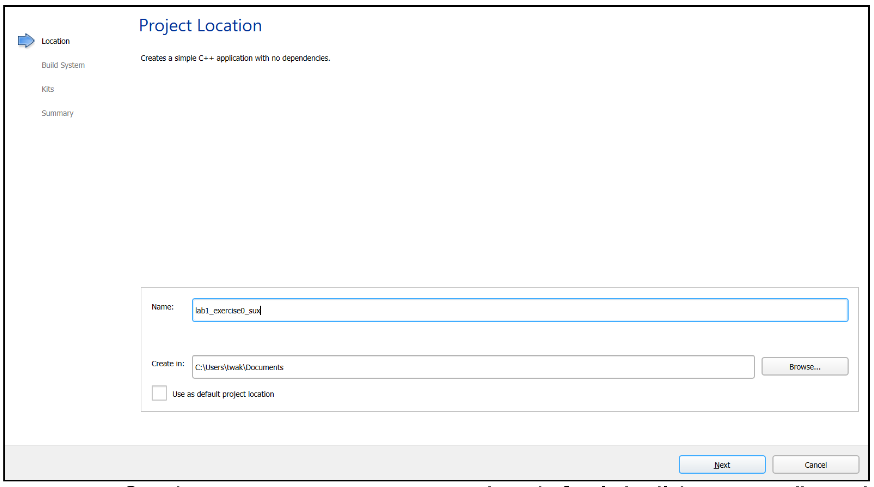
    - Under template select "Non-Qt Project" and then "Plain C++ Application". Click "Choose…"



Creates a simple C++ application with no dependencies.

Supported Platforms: `Desktop`

    - Enter a project name such as "Lab1_Exercise0", and a location to save it. Click "Next".



Project Location

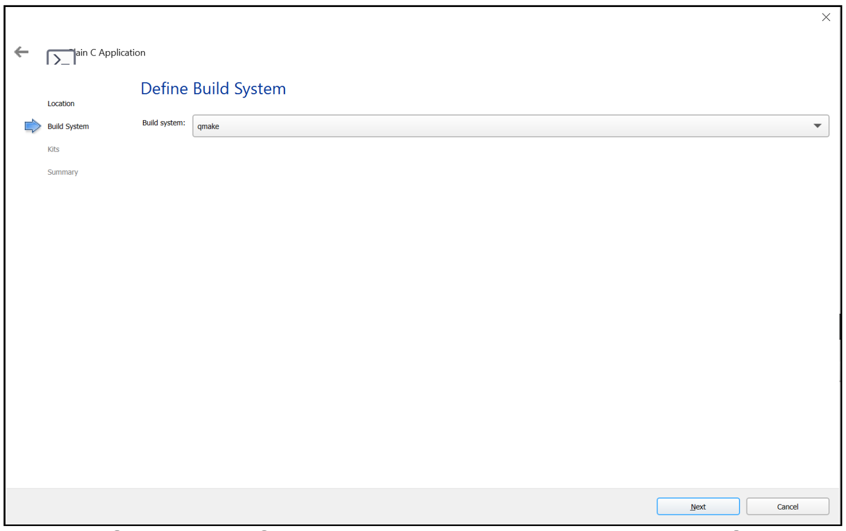
Creates a simple C++ application with no dependencies.

Name: lab1_exercise0_sux

Create in: C:\Users\twak\Documents

☐ Use as default project location

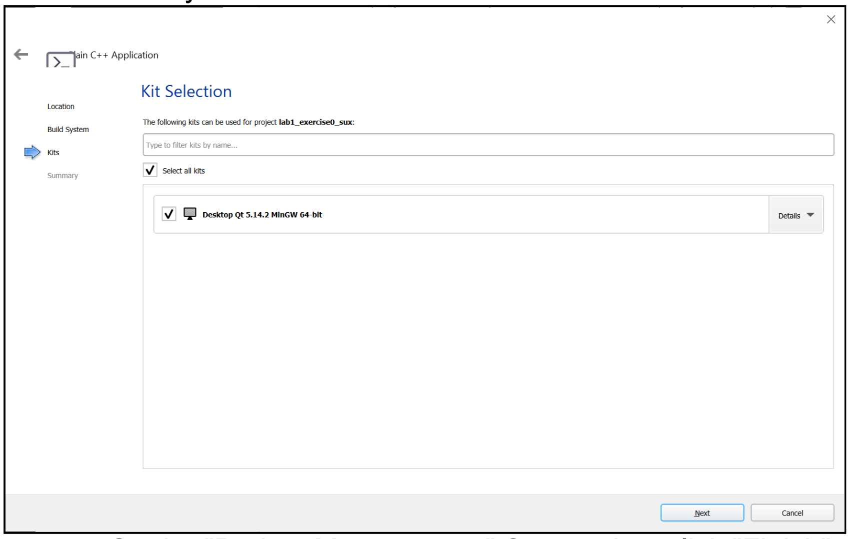
- On the next screen, accept the default build system "qmake". Click Next.



Define Build System

Build system: qmake

- On the "Kit Selection" page, select the Desktop Qt 5.XXX with your C++ compiler (where XXX are some numbers). This is usually the default. Click Next.



Kit Selection

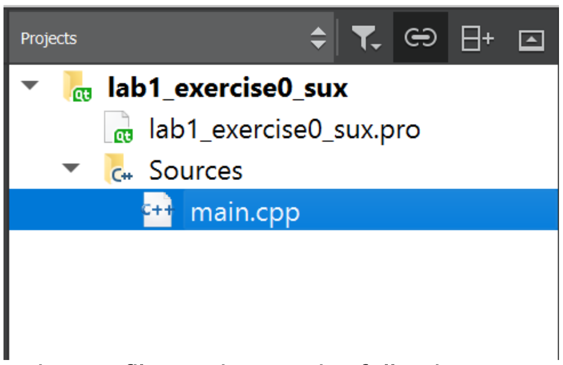
The following kits can be used for project **lab1_exercise0_sux**:

✓ Select all kits

✓ **Desktop Qt 5.14.2 MinGW 64-bit**

- On the "Project Management" Screen, just click "Finish".

3. Wait for Qt Creator to bring up the new project. When done shows the files in the project on the left. Right clicking on them will let you "Show in Explorer" (or similar) to see your code on disk.

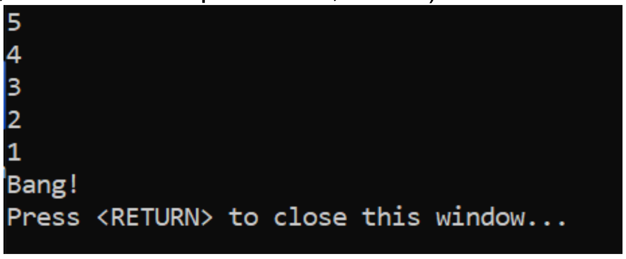


4. Open the main.cpp file, and enter the following.

```
#include <iostream>

int main() {

      for (int i = 5; i > 0; i--) {
            std::cout << i << std::endl; // this is like
      printf in c
      }

      std::cout << "Bang!" << std::endl;  // or println in
      Java
      return 0;
}
```

5. Save the code (Ctrl-S), and run the program with the debug button

6. In a new output window we see the console output (if this doesn't happen, see "common problems", above):



```
5
4
3
2
1
Bang!
Press <RETURN> to close this window...
```

7. Read the code you typed carefully. Understand the code.
8. Let's play - edit our program: change the `cout` to say print something crass. Run the program.
9. Edit the program, adding the following to the start of the program.

```
        using namespace std;
```

10. You can now remove both the `std::` from your code because the compiler will also search the `std` library.
11. `endl` does the same thing as `"\n"` in C. You can still use `"\n"` if you prefer, but it doesn't behave as well on different platforms.
12. Check the code still compiles and runs.
13. Edit the loop to compute the sum of numbers from 1 to 10. The program should put the sum at the end to `cout`.
14. This is the end of exercise 1. Just read until exercise 2.

Compiling C++

Just like C, you can also compile C++ on the command line. For example, you can use the `g++` program to compile a single file (`main.cpp`):

```
g++ -Wall main.ccp
```

However, Qt Creator uses qmake by to simplify this for you. It uses the *.pro* file to list the source files to be compiled and any options.

I/O in C++

IO in C++ is based around the concept of streams. This is similar to the Linux concept of pipes. Streaming is a very general concept which allows

- Including `iostream` gives you access to `cin` and `cout`, representing console input stream (`stdin`) and console output stream (`stdout`), respectively
- Including `fstream` allows you to define your own file-based streams, for reading from or writing to files.
- The << operator is used to *put* data into a stream
- The >> is used to *get* data from a stream into variables

Exercise 2: Follow these instructions.

1. Create a new project.
2. Enter the following function into main.cpp. Save your changes.

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

int main() {

      double x, y;
      cout << "Enter x & y coordinates: ";
      cin >> x >> y;
      cout << "Coordinates are (" << x << "," << y << ")"
<< endl;
}
```

3. Run the program. Click into the output window and type a number, then enter, then another number. Finally press enter again.
4. How do we *get* the values for x and y?
5. How do we use << to concatenate (join) strings?
6. Modify the program to test if the cartesian coordinate is within a rectangle
   a. the rectangle corners (0,0) and (3,3)
   b. print `"Inside"`if the point is inside the rectangle, otherwise print `"Outside"`
7. Edit the code to ask for 3 lengths, and prints the volume of a cuboid with those side-lengths:

```
Enter x length (m)> 3
Enter y length (m)> 2
Enter z length (m)> 1.5
The volume of the cuboid is 9!
```

C++ Strings

C has a `string` object, much like Java's `java.lang.String`

- C's representation of strings as null-terminated (they always end the magic character \0) char arrays. This is low-level, hard to use and prone to error.
- C++ strings are much more like those in Java – i.e., they are instances of a string class
- C++ strings know their size, manage their own storage and have many useful functions, just like Java strings

---

Exercise 3: Follow these instructions.

1. Create a new project.
2. Add the following to the start of *main.cpp*

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

3. Replace the main function with the following.

```
string target = "World";
string message = "Hello " + target + "!";
cout << message << endl;
cout << message.length() << endl;
cout << message[6] << endl;
cout << message.find("W") << endl;
message.replace(0, 5, "Goodbye");
cout << message << endl;
```

4. run the file:
   a. What does the `string::length()` function do?
   b. What does the `string::find("W")` function do?
5. The API for `string` lists other functions for the string object. Read the documentation and use `string::substr` in a `for` loop to give the following output:

```
hello world
hello worl
hello wor
hello wo
hello w
hello
hello
hell
he
h
```

C++ Collections

There are several ways to create collections of objects in C++. Let's look at some.

- Primitive local arrays work as C

```
int data[100];
int data2[] = {1,2,3,4};
data2[1] = -2;
cout << data[2] << endl;
```

- You can also store things in the collection types defined in standard
    - `vector` (like Java's ArrayList)
    - `list` (like Java's LinkedList)
    - `set` (like Java's HashSet)
    - `map` (like Java's HashMap)
- For example we can create `vector`s using a variety of constructors:

```
#include <vector>
#include <string>
using namespace std;

int main() {
      vector<int> a;         // empty vector for ints
      vector<int> b(10);     // 10 ints, all 0 (default)
      vector<int> c(5, -1);  // 5 ints, all -1
      vector<int> d(c);      // d will be a copy of c
      vector<string> p;      // empty vector for strings
      vector<string> q(5, "xyz"); // 5 strings, all "xyz"
}
```

- `<int>` specifies the type of the array, like in Java. C++ calls these *template parameters*.
- For example:

```
#include <string>
#include <iostream>
#include <vector>

using namespace std;
...

vector<string> words;

words.push_back("Apple");           // puts "Apple" at
end
words.push_back("Banana");          // "Banana" after
"Apple"
words.push_back("Kiwi");            // "Kiwi" after
"Banana"
cout << words.size() << endl;       // prints 3
cout << words[1] << endl;           // prints Banana
cout << words.at(1) << endl;        // same as above,
                                    // but at() does
bounds checking

words.pop_back();                   // removes "Kiwi"
cout << words.size() << endl;       // prints 2

words.clear();                      // vector now empty
```

Exercise 4: Complete the following tasks:

1. Create a new program which reads 5 numbers from the command line into an array and then computes the (mean) average:

```
enter 5 numbers:
>5
>6
>7
>8
>9
The mean is: 7
```

   a. write a `for` loop which reads in 5 `double`s, and stores them in an array.
   b. write a function which takes an array and returns its mean (average).
   c. use the function to print out the mean.

2. Modify the program to use a `vector<double>` instead of an array. You can iterate over a vector using the *range-based* `for` shorthand:

```
for (double i : doubleVector)
        cout << i << endl;
```

3. This also works for arrays - try out the range-based `for` for an array of strings!

Algorithms

The C++ standard library includes many functions to manipulate collections and strings – mostly accessed by including the `algorithm` header. The API is here. Have a read - there are many time-saving functions here.

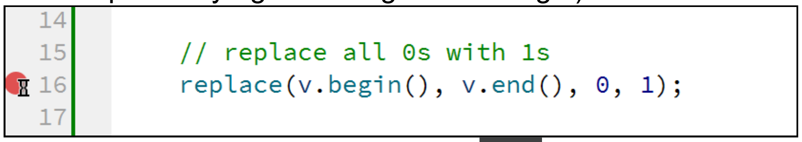
- Operations supported include sorting & searching, in-place transformations, accumulating sums, etc
- Iterators are used to specify the range of elements over which the operation is performed
- Collection types have `begin` and `end` functions that return iterators spanning the collection's elements
- For example:

```
#include <algorithm>
#include <vector>
using namespace std;

bool is_negative(int value) { return value < 0; }

int main() {

        vector<int> v;
        v.push_back(1);
        v.push_back(0);
        v.push_back(0);
        v.push_back(2);
```

```
          // replace all 0s with 1s
          replace(v.begin(), v.end(), 0, 1);

          // count negative values
          int n = count_if(v.begin(), v.end(),
is_negative);

          // replace negative values with 0
          replace_if(v.begin(), v.end(), is_negative, 0);
}
```

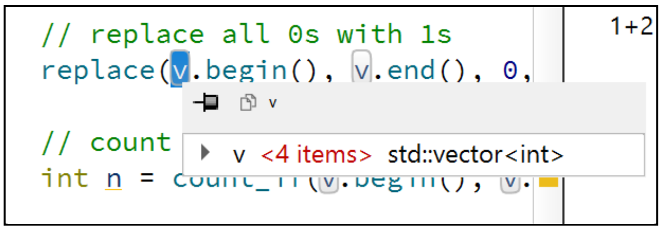
Exercise 5: Complete the following tasks:

1. The above program doesn't print out the result. Let's use the debugger to view the internal state of the variables. Setup the code as before in a new main() function. Add a breakpoint (the red circle) in Qt Creator by clicking in the margin the `replace` line. (You can also set breakpoints by right-clicking in the margin):



```
    // replace all 0s with 1s
    replace(v.begin(), v.end(), 0, 1);
```

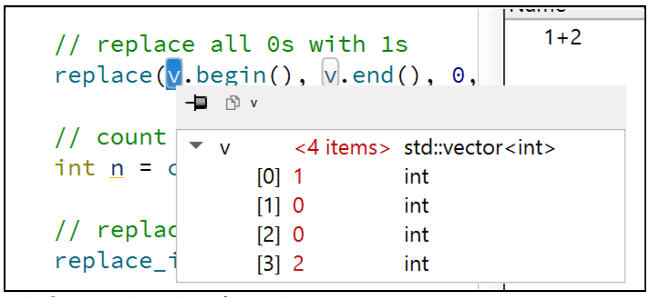
   a. now debug the program using . (Note the little grey bug on the icon). It will pause the program before the breakpoint. If you hover the mouse over variable `v`, it will let you explore it's values:



```
// replace all 0s with 1s
replace(v.begin(), v.end(), 0,

// count
int n = count_if(v.begin(), v.
```

```
▸ v  <4 items>  std::vector<int>
```

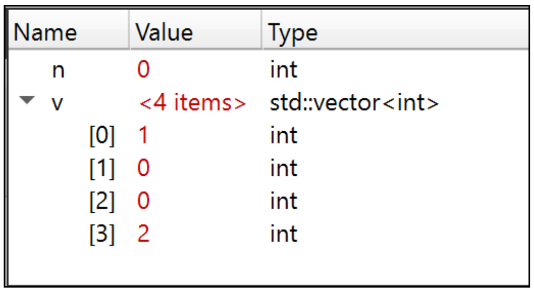
   b. Click the ▶ to see the values in the vector v



```
// replace all 0s with 1s
replace(v.begin(), v.end(), 0,

// count
int n = c

// replac
replace_i
```

```
▾ v        <4 items>  std::vector<int>
     [0]  1          int
     [1]  0          int
     [2]  0          int
     [3]  2          int
```

   c. This information is also shown in the debug pane at the top right.



Name	Value	Type
n	0	int
▾ v	<4 items>	std::vector<int>
[0]	1	int
[1]	0	int
[2]	0	int
[3]	2	int

2. Click the continue button ( ) to continue running the program to the end.
    a. These buttons let you control execution of the program:
    b. Run the program again. Try using the step-over button (or F8 key) to step through the rest of the program.
    c. What does `replace` do?
    d. What did `replace_if` do?
3. Create a program which reads in 5 doubles from the command line, and use `std::sort` in the `algorithm` header to print out a sorted list to the console.
4. Create a program which reads in 5 `double`s, multiplies each value by 2, then reverses the collection, and finally prints it out:

```
enter 5 numbers:
>5
>6
>7
>8
>9

18,16,14,12,10
```

    a. Hint: use the functions `std::transform` to apply a function which multiplies, and `std::reverse` to reverse a `vector`.

References

It was necessary in C to use *pointers* if we want a function to alter the variables passed to it:

```
void swap(int* x, int* y)
{
	int temp = *x;
	*x = *y;
	*y = temp;
}

int main()
{
	int a = 2, b = 7;
	swap(&a, &b);
}
```

If you've forgotten how pointers work, here is a recap of pointers in C (pages 1-4).

In C++, we can use pointers, but we can also use *references* – references avoid the need for special syntax in the function body or function call. They are similar to C's pointers, but can only point to a single object, and can never be `NULL`. This makes them safer to use. We can rewrite our swap function:

```
void swap(int& x, int& y)
{
	int temp = x;
	x = y;
```

```
     y = temp;
}

int main ()
{
     int a = 2, b = 7;
     swap(a, b);
}
```

Using const parameters

- By default C++ is call-by-value: function parameters are copies of the arguments passed to the function. This is costly for large arguments such as vectors with many elements
- Declaring a parameter as a reference avoids a potentially costly copy operation
- Declaring a parameter const means that the function cannot change the parameter. If the parameter is an object you may not change the object.

```
// Read data from a file into a vector
void read_data(const string& fname, vector& data) { ... }

// Compute mean value of data
float mean_value(const vector& data) { ... }
```

Classes

Like Java, C++ lets you define classes. This object oriented approach allows us to keep the code near the data it will operate on. The following is a simple C++ class.

```
class Foo{
public:

    Foo() {
        cout << "created" << endl;
    }

    ~Foo() {
        cout << "destroyed" << endl;
    }
    Foo(Foo& f) {
        cout << "copied" << endl;
    }
};
```

This class is called `Foo` and has three member functions (methods) defined

- `Foo()` is the constructor. We call this to create a new `Foo` object.
- `~Foo()` is the destructor. This is called when the object is destroyed. Often this is automatically called.
- `Foo(Foo& f)` is the copy function. This function is also often automatically called.

Let us see what happens when we call a function on a new `Foo` object

code	console output

`void aFunction(Foo g) {`
`}`
 	
`int main() {`
`   Foo f;`
	`created`
	`copied`
	`destroyed`
	`destroyed`

```
   aFunction(f);
}
```

We see that C++ default call-by-value behaviour means that we `copy` the original `Foo f` before we call the `aFunction` function. We then destroy the copied `Foo g`, and then finally destroy the original `Foo f`. If the copy function was slow this would have been a bad thing.

We can use a reference to avoid this behaviour:

code	console output

```
void aFunction(Foo& g) {
}

int main() {
   Foo f;
   aFunction(f);
}
```
	```
	created
	destroyed
	```

Member variables and class constructors

Like Java, we can add member variables and function to our class.

```
class Person {
public:
   string name;
   Person(string n) {
       name = n;
   }

   void printName() {
      cout << name << endl;
   }

...class continues
```

In this case we can create a new local object called firstPerson with the name bob with the following:

```
int main() {
   Person firstPerson ("bob");
}
```

Exercise 6: Complete the following tasks:

1. Start a new project and copy the above `Person` class into the *main.cpp*.
2. Add the above `main` function; add any `#include` or `namespace` statements you might need; run the file. What does it print out? Are you surprised?
3. Change the `Person` class so that it contains an integer `age` as well as a string `name`. Be sure to also change the constructor to assign the `age`.
4. Change the main function to:
    a. create a `vector` of 3 Person objects from names and ages given on the command line.

   b. print out the names and ages for each person in the list.
5. C++ has some shortcuts to assign member variables from the constructor. You can use the *member initializer lists*:

```
class Person {
public:
  string name;
  int age;

  Person(string n, int a) : name(n), age(a) { // nothing
else  }

  ...
}
```

6. Update your program to use this shorthand. Modify the `Person` class to take a first and a second name as well as an age.

Exercise 7 (optional): Additional exercise if you have extra time or want to improve your confidence with C++

Exercise 7 (optional): Additional exercise if you have extra time.

1. Write a function `palindrome` which tests if a string is a palindrome. You can use array notation (`aString[3]`) to get a character from the string. A palindrome is a word which is the same backwards as forward. The function should ignore spaces, for example:

```
palindrome("wanna"); // returns false
palindrome("anna");  // returns true
palindrome("hello"); // returns false
palindrome("are we not drawn onward to new era"); //
returns true
```

Further reading (these links may be useful for future assignments too).

- C++ language tutorial
- C++ FAQ
- Book: Lippman, Lajoie & Moo (2012) C++ Primer (5th ed.)

Acknowledgements:

- Nick Efford's courses https://comp1721.info/, and http://comp2811.efford.org
- Hamish Carr's MJ21 course
- Tom Kelly