一、在學 C++之前(Before Learning C++)

我認為在學 C++之前,可以先了解一點與電腦相關的基本知識,這樣對於學習 C++會有比較好的領悟，不過這是想要對程式語言有比較深入的了解時才需要的，因此如果沒有這個需求，是可以跳過這一篇的。想要更深入的了解，可以去學習「計算機組織(Computer Organization)」或「計算機結構(Computer Structure)」。

為了跳過這篇的人著想，我把這放前面好了,「C++ reference」是一個關於 C++的網站，是全英文的，但如果英文還可以，這個網站在標準函式庫與基本語法描述算是詳細的，是很有用的網站。

I. 硬體與軟體(Hardware and Software)

1. 硬體(Hardware)

看得見的設備，舉凡我們在使用電腦時所用或是拆解電腦後所看到的都是硬體，基本上就是現實存在的實體。而與電腦執行的程式直接相關的是負責計算與邏輯判斷的處理器，本身內部是一塊複雜的電路。

2. 軟體(Software)

看不見的程式，負責指示處理器做事，本質上是邏輯訊號。電腦所執行的程式會透過對中央處理器下達指示，運用算術邏輯單元進行計算，從輸入裝置取得外界資訊，由記憶體或輔助記憶體記錄這些資訊，計算後透過輸出裝置傳到外界。接下來會簡單介紹這些東西。

II. 電腦基本組成(The Structure of Computer)

1. 輸入裝置(Input Device)

接收外來的資訊，等待後續的處理，是外界(像是我們、其他電腦)與電腦溝通的橋樑。像是鍵盤、滑鼠、麥克風等。

2. 輸出裝置(Output Device)

將處理後的資訊傳到外界，與外界互動。像是螢幕、喇叭、印表機。

3. 算術邏輯單元(Arithmetic and Logic Unit , ALU)

負責做數值運算、邏輯判斷的電路晶片，現在一般被包含在 CPU 之中，是電腦的核心部分。

4. 中央處理器(Central Processing Unit, CPU)

負責安排現在電腦要做甚麼事情，並協調各個裝置，是整台電腦的核心。

5. 記憶體(Primary Memory)

負責存放等待處理的資料，當電腦電源關閉時儲存的資料會消失，像是 DRAM、SRAM。

6. 輔助記憶體(Secondary Storage Memory)

負責存放長期資料，電腦電源關閉時亦會保留資料。像是硬碟、磁碟，光碟、隨身碟等。

III. 程式語言(Programming Language)

1. 機器語言(Machine Language)

電腦真正了解的語言，由 0 與 1 的邏輯信號所構成，與硬體間有相依性，每個處理器所用的語言不同，對於人類來說難以理解，因此出現了組合語言。

2. 組合語言(Assembly Language)

將一些常用的指令如加減乘除等以英文縮寫來描述，這樣就可以不用知道每個指令 0101 的值是多少，比較容易看懂。通過名為組譯器(Assembler) 的程式轉換成機器語言。仍然與硬體有相依性，而且在複雜的程式不易理解，

因此出現了高階程式語言。

3. 高階程式語言(High-level Programming Language)

將程式語言變成由英文單字與符號所組成的語法，與我們所理解的英文貼近，容易理解，通過名為編譯器(Compiler)或直譯器(Interpreter)的程式轉換成組合語言或是機器語言。藉由編譯器與直譯器對各個處理器的支援，可以與硬體無相依性。

也許這樣講解可能還是不太懂這三種語言的差異,就以我學過的 MIPS 組合語言與 C++來寫個簡單的加法例子，自己體會一下吧。同樣一個數值相加再存起來的計算

機器語言長這樣	00100010000100000000000000000001
MIPS 組合語言長這樣	addi s0 , s0 , 1
C++長這樣	i = i + 1;

IV. 編譯器與直譯器(Compiler and Interpreter)

高階程式語言根據轉換方式的不同又簡單分為編譯式與直譯式兩種，各有各的好處。

1. 編譯器(Compiler)

編譯器會將程式全部轉成 0101 的二進位檔，可以直接執行，優點是執行速度快，缺點是程式一改就必須重新編譯一次才可以保持最新狀態，而編譯出來的檔案基本上會與作業系統有相依性，對不同作業系統需要用不同的編譯器來做編譯。C++就是這種類型的語言。

2. 直譯器(Interpreter)

直譯器一邊翻譯程式一邊執行，好處是可以一步一步地看到程式是如何執行的，方便學習與除錯，缺點是執行速度較編譯式來的慢，但是現在有一些最佳化的技術讓這個速度差變得比較小了，而透過對各個作業系統的直譯器，可以對作業系統沒有相依性。TCL、Python 是這種類型的語言。

Java 算是介於兩者之間的程式語言，根據我之前看的資料，它是先以編譯的方式轉成一種名為位元碼(Byte Code)的程式碼，在執行時再透過虛擬機以直

譯的方式執行。

V. C++簡介(C++ Introduction)

C++是從 C 衍生出來的，因此在語法上基本與 C 無異，但這兩個程式語言還是有所差異(亂講兩者一樣可能會被砲轟)。C++引入了物件導向程式設計的概念，讓大型程式的開發變得較為容易理解。

1. C 語言(C)

C 主要由語法與標準函式庫(Standard Library)所構成。一般會將程式分成許多小功能來寫，然後再將這些功能兜起來完成整個程式，達到模組化設計(Modular Programming)。標準函式庫是由其他人寫好的一堆程式碼，包含許多常用的功能，像是輸出資料、數學函數計算等，可以簡化許多複雜、專業的功能。

2. 物件導向程式設計(Object Oriented Programming)

物件導向程式設計主要是因應於結構化程式設計(Structured Programming)在大型程式開發上的困難而興起的技術，結構化程式設計將整個設計看成一段一段的指令敘述，對於我們是比較不直觀的。而物件導向程式設計將設計看成是一堆物件(Object)之間的溝通，把裡面的事物看成現實的一個實體，像是動物、手機、電視等，對於我們是較為直觀的，因此在設計上比較容易除錯，比較適合大型程式，但物件本身仍須用結構化程式設計來實現。這在 C++較進階的部分會比較有感觸。

VI. 編譯流程(Compile procedure)

簡單了解編譯的流程對於學程式的一些概念時會比較理解，程式從寫好到編譯可以執行基本需經過以下幾個步驟

1. 前置處理(Preprocess)

這個步驟基本上會引入一些寫好的檔案(像是標準函式庫)，或是對程式碼做一些文字上的處理，讓接下來的編譯擁有完整正確的資訊。這就像做一份報告時要先查詢資料並做些處理。

2. 編譯(Compile)

這個步驟就是將程式碼轉換成機器語言或組合語言，而在專案開發時，程式碼會分成許多部份，每份程式碼間可能有相依性，因此會編譯成有一些小空缺的目的檔(Object File)。這就像做報告時，分成幾個部份給每個人做，而有些資料是別人查的，因此自己還不能做描述就先空下來。

3. 連結(Link)

這個步驟會將編譯好的目的檔進行合體，將剛剛空下來的部分填上去，完成整個程式。就像做報告最後做統整，將各部分缺少的資料補上去。

VII. 整合開發環境(Integrated Development Environment, IDE)

在做程式設計時基本上需要很多程式來完成，撰寫程式碼時需要編輯器(Editor)，編譯時需要編譯器，連結時需要連結器(Linker)，除錯時需要除錯器(Debugger)，這麼多東西有時候切來切去有夠麻煩的，因此就有人將這些功能做整合，做成名為整合開發環境的程式，一個程式就可以做到所有需要的功能，開發上較為方便。

以下是我在學習中比較常聽或是正在使用的 IDE，還有許多其他的，選個自己喜歡的來用即可。

1. Visual C++

這是微軟(Microsoft)開發的 IDE，支援微軟的.NET 框架(.NET framework)。可以在微軟官網下載免費版，較舊的版本有支援中文。如果有想要專門開發 windows 的程式，我想選擇他們自家做的 IDE 會有比較好的支援。

2. Qt Creater

這主要是用來開發 Qt 框架(Qt framework)使用的 IDE，Qt 框架是一個跨平台讓 C++用來設計使用者介面的應用程式介面(Application Programming Interface, API)，簡單說就是一堆人家寫好的程式碼可以用，詳細自己研究比較有感(不過官方文件全英文)，不過這在學完 C++進階的部分後再研究比較懂。總之也可以用來寫簡單的 C++程式，我自己是覺得它介面、功能還不錯。它有免費與付費版，如果沒有要開發商用軟體，可以去下載免費版的來用。

3. Dev C++

這是我學 C++時用的簡易 IDE，功能較前兩個來說沒那麼強大，但對初學來說還夠用，體積也很小，我自己現在還是用這個 IDE 啦。它是完全免費，而且不用註冊帳號，不過要注意的是它分兩種版本，4.9.9.2 前跟 4.9.9.2 後，之前的開發者開發到 4.9.9.2 就終止開發，後來有好心人士接手出了 5.多的版本，新的版本才有支援到比較新的作業系統，bug 也比較少。

二、基本架構(Basic Structure)

寫程式都要從基礎開始，所謂基本架構也不是絕對的，只是因為絕大多數的程式都會用到一些固定的敘述，因此才稱之基本架構。為了讓沒有想要深入了解的人迅速知道這篇會有什麼，我先將會講到的程式碼與執行結果放上來，如果真的看不懂的話可以繼續看下去。

```
1  /*
2    這是註解
3    註解是給人看的，對程式執行沒意義
4  */
5
6  #include <iostream>   // 引入 iostream 檔案供輸出入使用
7  using namespace std;  // 使用命名空間 std ，不然 cout 要寫成 std::cout
8
9  int main() // 程式開始執行的地方
10 {
11   int number; // 宣告變數
12
13   cout  << "請輸入一個數字:"; // 輸出文字
14   cin   >> number;           // 等待輸入數字，按 enter 結束
15   cin.get();                 // 去掉 enter 鍵
16
17   cout  << "這是一個簡單的C++程式\n";              // 輸出文字並換行
18   cout  << "你剛才輸入的數字是 " << number << endl; // 輸出文字與數字並換行
19
20   cin.get();  // 等待按鍵輸入
21   return 0;   // 程式正常結束
22 }
```

```
請輸入一個數字:123
這是一個簡單的C++程式
你剛才輸入的數字是 123
```

要執行程式，首先將編寫程式的軟體打開，打完上面的程式碼後，尋找工具列中名為「編譯並執行」、「執行」或是「建置並執行」這類的選項按下去就可以了，如果有打錯什麼的話編譯會出現錯誤，一般軟體會跟你說錯誤的原因，只要看懂並修改一下就行了。

I. 主程式區塊(Main Function)

那麼接下來就開始講正題吧，一般教程式會直接一行一行講給你聽，我想換個方式。首先一個程式到底要從哪裡開始執行呢，C++定義了一個特別的名字叫作 main，這個字有主要的意思，而程式就會從這個地方開始執行，完整一點長下面這個樣子。這被稱為 main 函式(函式在進階部分會提到)

```
1  int main()
2  {
3  }
```

第一行的 int 指的是整數(integer)，這是程式執行完後回傳給電腦的資訊，末端的小括號(parentheses)是用來寫輸入的參數(parameter)用的，現在不會用到，因此留空即可，在進階一點的部分會再說明。

為什麼要有輸入與回傳的資訊呢，拿之前上課老師舉的例子來說，媽媽叫我們去買菜，可能會給我們錢，這是輸入，買完菜之後要跟媽媽講一聲，這是回傳資訊。同理電腦執行程式也是這樣，所以會需要這些敘述。而一般程式回傳的資訊是代表執行有沒有順利結束。

至於大括號(curly braces)的部分，括起來的地方稱為一個區塊(block)，當在執行許多敘述時，這些敘述屬於誰呢？在 C++中用區塊來決定，在此處當然是指屬於 main 啦。

值得一提的是，C++中除了一些要識別的部分外，可以用空白字元(null character)來對程式碼作排版，以這邊來說寫成

int main(){}

也是可以的，但是這樣會很難看，而且在寫複雜的程式時這樣寫可是會吃盡苦頭的，我習慣會將區塊的大括號分別各占一行，比較清楚。之後在說明程式碼時我會講一些我自己的習慣作排版的參考，選自己喜歡的來用就可以了。喔對了，int 和 main 不能連在一起喔，這是需要被辨識的部分。

在 C++中空白字元有三種，空白鍵、Tab 鍵、換行鍵(一般是 Enter)，善用這些字元可以讓程式碼容易理解。

II. 主程式回傳值(Return Value)

好了，這樣程式就可以執行了，只是沒有任何功能就是了。你會覺得有哪裡怪怪的嗎，剛剛不是說程式要有回傳的部分，可是目前講到的程式碼沒有做這件事啊。老實說如果沒有回傳資訊，C++預設是回傳成功執行，當然我在學的時候沒人這麼講，所以我自己還是習慣加上回傳的部分，比較明瞭。加上回傳的部分後程式碼就會變成這樣啦。

```
1  int main()
2  {
3    return 0;
4  }
```

return 是 C++的關鍵字(keyword)，關鍵字就是用來描述一些語法的字，不能被用做其他用途。return 就是回傳的意思，後面接的是回傳的資訊，在這裡是 0，一個程式回傳 0 代表成功執行結束，其他值則表示有問題發生，至於什麼值表示什麼意思就看系統或我們自己的定義了。值得注意的是最後的分號 (semicolon)，這相當於我們寫文章的句號，代表一個敘述(statement)的結束，一個敘述就是在程式中會產生作用的語句，初學者很容易忘記這個分號而在編譯的時候跳出錯誤訊息。

III. 標準輸入與輸出(standard input and output)

接下來就是為這毫無作用的程式加點料的時候了，這一篇我會簡單講如何讀到鍵盤輸入的字、如何輸出文字到螢幕，這是許多程式都會用到的功能，因此算是其基本架構之一。

首先呢，C++使用輸入、輸出是用標準函式庫的東西，我們要先引入名為 iostream 的檔案，它代表的意思是 Input / Output STREAM ，直譯就是輸入輸出串流，串流簡單說就是一堆 0101 的資料。那麼引入這份檔案的語法是

#include <iostream>

前一章有提過前置處理器，而#就是 C++中用來跟前置處理器講話的語法，稱為前置處理指令(preprocessor directive)。這種指令與 C++其他語法不同，C++會將#那一行都視為前置處理指令，因此要自己獨立成行。

include 是包含的意思，這表示這份原始碼(source code)中會引用某份檔案中的資料，原始碼就是我們寫程式碼的檔案。而<>就只是語法的一部份，被它包起來的文字就是檔案名稱。

在 iostream 裡面呢，定義了 cout 與 cin 兩個名字，分別表示標準輸出與輸入，代表 Character OUTput、Character INput，而這名字又被定義在 std(STandarD)

的命名空間(namespace)中。什麼是命名空間呢，簡單說就是一個用來區隔其他人寫的程式碼用的東西。之後會講到可以自己定義名字，那如果有許多名字都定義一樣怎麼辦呢？就像我們如何區別文藝復興時期的李奧納多與皮卡丘的李奧納多，家庭教師的納茲與魔導少年的納茲，灌籃高手的阿神與芋園柚子的阿神。發現什麼了嗎，不是動漫人物或有名人士，我們在清楚說明同一個名字的人時會多個名稱區別不同的領域，命名空間就是同樣的道理，定義一個領域的名稱。C++的標準函式庫基本上都是定義在 std 這個命名空間之下

而使用 cout 與 cin 很簡單，照著以下規則即可

std::cout << 輸出的資料 1 << 輸出的資料 2 …;
std::cin >> 輸入的資料 1 >> 輸入的資料 2 …;

要表示一個命名空間中的名字就要用::來描述，std 中的 cout 就是 std::cout。而要輸出資料就只要將資料用<<丟給 cout，簡單明瞭，方向都指明了，cin 也是類似的道理，只是我們是用>>取出資料。cout 預設對應到電腦螢幕，cin 預設對應鍵盤。

我們先將現在講到的程式碼整理一下，一個簡單輸出文字的程式長這樣

```
1  #include <iostream>
2
3  int main()
4  {
5    std::cout << "我會寫程式" << std::endl;
6    std::cout << "我出師啦!\n";
7
8    return 0;
9  }
```

奇怪，怎麼好像少了些什麼，然後又多了些什麼。嗯...我會再說明啦，首先用雙引號(quotes)括起來的叫做字串(string)，簡單來說就是一堆文字，之後會再詳細解釋。接著後面的 endl 表示 END Line，也就是換行的意思，這是 cout 特有的參數，未來的篇章會再多講。再來注意到第二個字串中的\n，這是字串中特有的東西，以\加上文字或符號稱之為跳脫字元(escape character)，是有特殊意義的字，這裡的 n 表示 new line，也是換行的意思。關於跳脫字元之後會再詳述。

至於為什麼要分別用\n 與 endl 來作換行呢，看了一下說明文件，在內部實際的行為上兩者有點差別，但對於一般的使用是沒有分別的，所以挑個自己喜歡用的就好了。

那為什麼沒有講到 cin 呢，這是因為在用 cin 之前還得做些準備，首先我們接收了輸入之後要送去哪裡勒？總要有個容器去接收拿到的資料吧，在程式裡用來接收輸入的東西叫做變數(variable)，之後的篇章會詳細的介紹，現在為了使用 cin，我先簡單講一下，要使用一個變數要先對其作宣告(declaration)，在這裡我想用整數，其宣告方式如下

int number;

這樣就可以使用名為 number 的變數了，那就修改一下程式碼來接收鍵盤的輸入吧。

```
1  #include <iostream>
2
3  int main()
4  {
5    int number;
6
7    std::cout << "請輸入一個數字:";
8    std::cin  >> number;
9
10   std::cout << "這是我的第一份程式\n";
11   std::cout << "你剛才輸入的數字是" << number << std::endl;
12
13   return 0;
14 }
```

這裡有個建議，在使用 cin 時，前面最好作個 cout 來告訴使用者現在要輸入東西，不然莫名其妙程式就停了誰知道你要做什麼，適時的 cout 出一些文字可以讓使用者較為明瞭程式要做什麼。cin 會讓使用者輸入任意的字數，直到按下 enter 鍵，如果怕使用者不會用的話可以多加幾句話作說明。

IV. 暫停程式(Pause the Program)

那麼第一份程式就這樣結束了嗎，你可以將程式作編譯後執行看看，你看到了什麼？如果輸入完數字後沒什麼大問題，那你改成直接執行編譯好的執行檔看看，包準你輸入完文字程式直接跳掉(正常的話)，這是為什麼勒。首先可以看到整個程式中除了 cin 的地方會停下等你輸入文字之外，有哪裡會停的嗎，答案是沒有。

電腦執行速度是很快的，如果沒有寫讓程式暫停的敘述，那程式一下子就執行結束並關閉了，最簡單的方式是在 return 之前加上 cin.get()這個敘述，這與 cin >>是類似的，同樣是取得輸入，但是 cin.get()只會取得一個字元(character)，字元簡單講就是一個文字或符號，而 cin.get()與 cin >> 同樣都是按下 enter 鍵才

會開始取得輸入。

但是加上這個敘述之後似乎還是會跳掉，這是為什麼呢。根據我的理解，在 cin 時所按下的 enter 也會被取到，所以當 number 得到數字之後，輸入還有一個 enter 鍵，而它就被 cin.get()取到了，cin.get()有跟沒有一樣。最簡單的方法是在 cin 完之後馬上接一個 cin.get()，將這個 enter 鍵消掉，這樣等輸出完文字後就還要再等一個 enter 鍵，就可以達到停止的效果了。

講了這麼多，整理一下現在講到的程式碼就會變成這樣

```
1  #include <iostream>
2
3  int main()
4  {
5    int number;
6
7    std::cout << "請輸入一個數字:";
8    std::cin  >> number;
9    std::cin.get();
10
11   std::cout << "這是我的第一份程式\n";
12   std::cout << "你剛才輸入的數字是" << number << std::endl;
13
14   std::cin.get();
15   return 0;
16 }
```

V. 命名空間 std (Namespace std)

好了，那基本上一個簡單的輸入輸出程式就完成了，有沒有覺得一直打 std:: 很麻煩？如果有的話再教一個敘述

using namespace std;

很直白的意思就是使用 std 這個命名空間，這表示之後的程式碼若有 std:: 都可以省略，因為程式知道 std 這個命名空間裡所定義的任何名字，那麼程式就可以改成這樣啦

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

int main()
{
  int number;

  cout << "請輸入一個數字:";
  cin  >> number;
  cin.get();

  cout << "這是我的第一份程式\n";
  cout << "你剛才輸入的數字是" << number << endl;

  cin.get();
  return 0;
}
```

有沒有簡單明瞭呢？沒有的話就是我講太爛了，算了吧就繼續講下去。基本上 C++最基本的概念大略就是這樣，雖說有些東西學完後面會更了解一點，但現在就先知道這些就好。接下來講一點額外的東西。

VI. 註解與縮排(Comment and Indentation)

首先講一個對寫程式碼很重要的兩個東西，一個是註解，一個是縮排，註解顧名思義就是對程式碼作解釋，寫程式碼註解是一件很重要的事，主要是讓看程式的人可以了解這些程式碼在做什麼。這在大型程式尤為重要，大型程式很複雜，如果不加些註解，久沒看就會不知道程式在做什麼。而註解對程式的執行是沒有任何影響的，單純給人看用，編譯器與前置處理器會略過它。

在 C++中有兩種註解方式，分別如下

```cpp
// 這是註解
/*
  這是註解
*/
```

//是單行註解(line comment)，也就是那一行//之後所寫的東西都是註解，而/**/是區塊註解(block comment)，只要包在/* */之間的文字都會被忽略，我們就對剛剛寫的程式做點註解吧。

```cpp
#include <iostream>   // 引入 iostream 檔案做輸入輸出
using namespace std;  // 讓 cout, cin 等方便使用

int main() // 主程式
{
  int number; // 宣告變數

  cout << "請輸入一個數字:";  // 輸出文字
  cin  >> number;             // 等待輸入數字
  cin.get();                  // 消去enter

  cout << "這是我的第一份程式\n";                  // 輸出文字並換行
  cout << "你剛才輸入的數字是" << number << endl; // 輸出文字與數字並換行

  cin.get();  // 讓程式暫停
  return 0;   // 程式正常結束
}
```

註解要做到多細端看個人習慣，像現在這個程式很簡單，在學久之後一看就懂，我就不會作註解(除了去掉 enter 那行，不註解很不直觀)。

縮排的話前面應該也有提過，主要是讓程式碼清晰容易閱讀，像{}我自己習慣各占一行，而一些功能有區別的部分我會空一行，像是讀入數字與輸出取得的數字算是兩個分開的項目，我覺得空一行比較清晰。

值得一提的是{}內的程式碼一般前面會空幾個空白鍵，將階層排出來會更容易閱讀。我習慣是用 tab 鍵，而我建議設定一下編輯器，一般會有一個地方可以設定要不要將 tab 改成空白鍵，最好是改成空白鍵，這樣才不會在不同人設定不同 tab 長度的情況下讓程式碼排版跑掉。tab 長度建議是 2~4 個空白鍵，看自己喜好，我自己是用 2 個。文字與符號間建議是隔個空白鍵，以 cout 為例自己感受一下(如果你覺得沒差我也沒辦法)

```cpp
cout<<"沒空空白鍵";
cout << "有空空白鍵";
```

註解與縮排是一個程式設計的重要精神，也就是可讀性(readability)，一份好的程式碼應當是容易看懂的。

VII. 其他的暫停程式方式(Other Way to Pause the Program)

最後講講暫停的敘述，我知道的方式有三種

```
cin.get();
getchar();
```

system(“pause”);

第一種是前面講的，這種方式比較符合 C++的風格(C++ style)。第二種方式是 C 的風格，作用與 cin.get()一樣，只是要引入 cstdio 檔案，代表 C STandarD Input / Output。第三種比較不建議，不過我學的書都是教這種，是後來看到一篇文章才不用的。先講講它的作用，system(“pause”)會讓程式強制暫停，直到按下任意按鍵，需要引入 cstdlib 檔案，代表 C STandarD LIBrary，也就是標準函式庫，不過是狹義的。

第三種方式為什麼不建議呢，據我所知有兩點，一個是它與作業系統有相依性，pause 是 MS-DOS 上的一個指令，而不保證每個作業系統都適用，儘管程式編譯後只會在特定作業系統上執行(一般 windows 都有 MS-DOS)，但程式碼本身盡量不要對作業系統有相依性，這是設計程式的另一個精神，稱為可移植性(portability)。當然如果很確定只會在 MS-DOS 下執行程式，那就無所謂，但是同樣一個功能，如果能用與平台相依性低的方式，我會傾向那種方式。另一個原因是有可能出問題，除非能保證指令不會錯。詳細還是看那篇文章比我講得來的好(如果有興趣的話)，網址我就放在下面

http://www.cplusplus.com/articles/j3wTURfi/