在 Windows 上使用命令行编写 C++程序

1．安装 MinGW。

如果你安装了 Code::Blocks，并且选择了带 MinGW 的那个版本，那你应该已经安装过 MinGW 了，可以跳过步骤 1。

什么是 MinGW?

MinGW(Minimalist GNU For Windows)是个精简的 Windows 平台 C/C++编译器。

a) 下载 MinGW。官方网址：http://www.mingw.org/

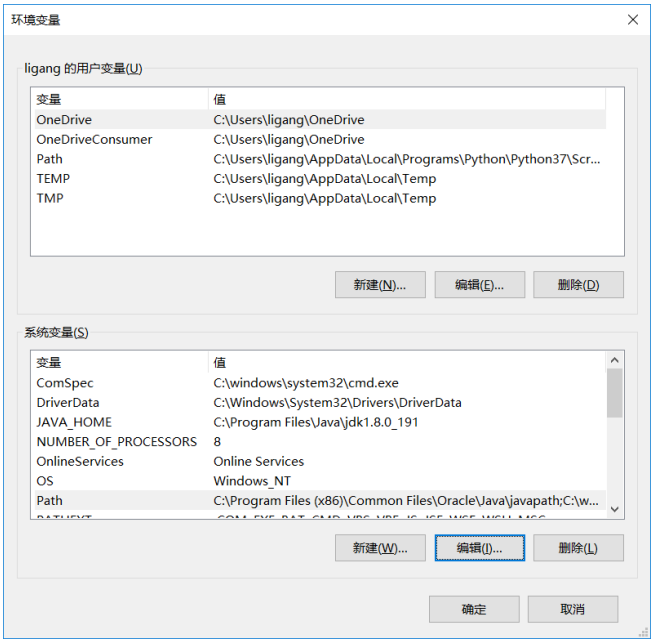
b) 默认安装目录是 C:\MinGW

c) 可以参考：《MinGW 安装和使用》https://www.jianshu.com/p/e9ff7b654c4a

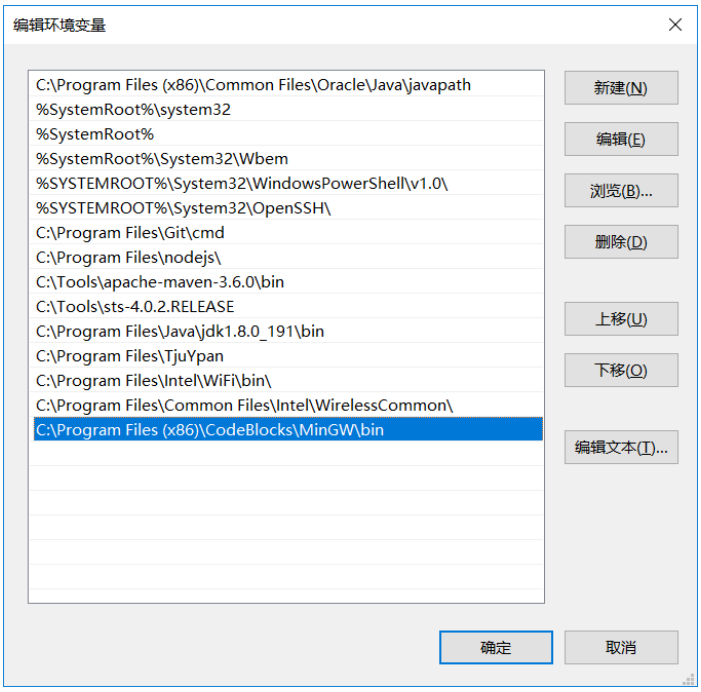
2．设置环境变量 PATH。

a) Win+E，呼出资源管理器，在“此电脑”上打右键，选“属性”，出现“系统”设置。

b) 点高级系统设置，选“环境变量”，在里面的“系统变量”中找到 PATH 这个变量，点“编辑”



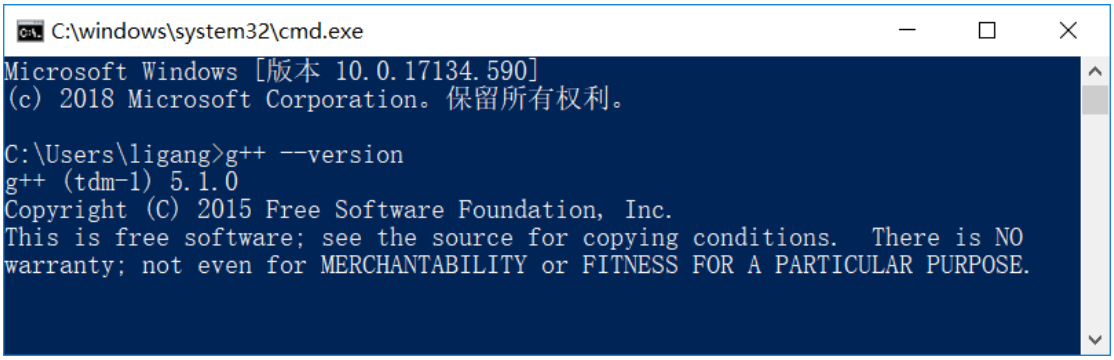
c) 点“新建”，把 MinGW 的可执行文件目录加到 PATH 中。
如果缺省安装 Code::Blocks，那么 MinGW 的可执行文件目录是:
C:\Program Files (x86)\CodeBlocks\MinGW\bin
如果直接安装的 MinGW，那么是：C:\MinGW\bin



3．测试一下编程环境

a) Win+R，键入 cmd，运行“命令行终端”

b) 在命令行上键入 g++ --version，如果安装 MinGW 成功，可以看到 g++的版本信息。注意 g++和--version 之间有一个空格。



4．编写 helloworld 测试一下。

a) 用任意编辑器（比如记事本），编写 main.cpp，保存到某目录下，例如，d:\code 。
注意记事本默认在文件名后面加上.txt，保存时保存类型要选“全部文件”。程序如下:

```
#include <iostream>
using namespace std;
int main()
{
    cout <<"Hello world!" << endl;
    return 0;
}
```

提示：main()函数的返回类型必须是 int，程序正常执行时的返回码必须是 0。这个返回码是执行你程序的调用者需要的。

b) Win+R, cmd，运行“命令行终端”，使用下面的命令：

```
d:
cd \code
g++ -std=c++11 main.cpp -o main.exe
```

解释：如果你的代码放在 D 盘，d:就是切换到 D 盘的意思。

cd 是把当前的工作目录切换到代码所在的目录。

g++是编译工具，-std=c++11 的意思是使用 ISO C++11 标准进行编译，然后是源代码文件的名字，然后-o 是指定生成的可执行文件的名字是 main.exe，注意各部分中间都有一个空格，即

g++<空格>-std=c++11<空格>main.cpp<空格>-o<空格>main.exe

如果程序中有错误，会产生 warning 或 error，一般 warning 可以不理。

如果程序没有任何错误，那么就也没有任何提示出现。没有消息就是最好的消息。

c) 用 dir 命令，看看当前目录下有没有生成一个 main.exe。有的话，执行 main，就可以了。

5．编写一个带自定义类的程序。

a) 在与 main.cpp 相同的目录下，新建 MyClass.cpp 和 MyClass.h。

b) 在 MyClass.cpp 和 main.cpp 中，都是使用下面的语句包含头文件。

```
#include "MyClass.h"
```

说明：自定义类的头文件要用引号，系统的头文件可以使用尖括号<>

c) 编译的语法：

```
g++ -std=c++11 *.cpp -o hello.exe
```

*.cpp 代表当前目录下

6．编写带命令行传参的程序。

a) C++标准的 main()应该是这样定义的：

```
int main(int argc, char* argv[]) {
    return 0;
}
```

解释：argc 是参数的个数，argv 的参数的数组。

例如：main ant bee cat dog，那么，argc=5，argv[0]=”main.exe”, argv[1]=”ant”, argv[2]=”bee”, argv[3]=”cat”, argv[4]=”dog”

b) 参考例程如下：

```
#include <iostream>
using namespace std;
int main(int argc, char* argv[]) {
    cout << "You input " << argc << " arguments." << endl;
    for (int i=0 ; i<argc ; i++) {
        cout << "The No. " << i << " argument is "<< argv[i] << endl;
    }
    return 0;
}
```

7．输入输出重定向和管道

一个程序如果有输入输出，默认情况下，输入来自键盘，输出目的地是屏幕。也就是说，当你 cin 的时候，程序会停下来等你敲键盘，当你 cout 的时候，程序会把要打印的内容放在屏幕上。但是，这一切都是可以改的。

a) 如果你把要在 cin 时输入的内容，提前存入一个文本文件中，使用输入重定向，就可以让程序在 cin 时不再等待你的键盘输入，而是直接读取输入文件，例如下面的代码：

```
#include <iostream>
using namespace std;
int main(int argc, char* argv[]) {
    string message;
    cin >> message ;
    cout << "Your message is " << message << endl;
    return 0;
}
```

在与可执行文件相同的目录中，建立一个 message.txt 文件，内容是：helloworld

执行：main < message.txt

程序就不再等待键盘输入，而是直接输出：Your message is helloworld

输入重定向在你要做多次大量的输入非常有用。

b) 同样的，如果你使用了输出重定向，就意味着本来应该输出到屏幕上的内容，被放到输出文件中了，屏幕上就干干净净了。例如：

执行：main < message.txt > result.txt

就会在相同目录下多出来一个 result.txt，内容就是 Your message is helloworld

输出重定向在你有大量的输出需要保存下来进一步分析时非常有用。

c) 管道，是将前一条命令的输出，直接作为后一条命令的输入。例如：

执行：echo hello | main

结果：Your message is hello

echo 的作用就是输出一个字符串，此处 echo hello 本来应该往屏幕上输出 hello，但是经过管道传给了 main，后者就将 hello 作为输入了。