C++ Program Design

-- VC_Win32控制台应用程序 (ConsoleApplication)

Junjie Cao @ DLUT

Summer 2021

https://github.com/jjcao-school/c

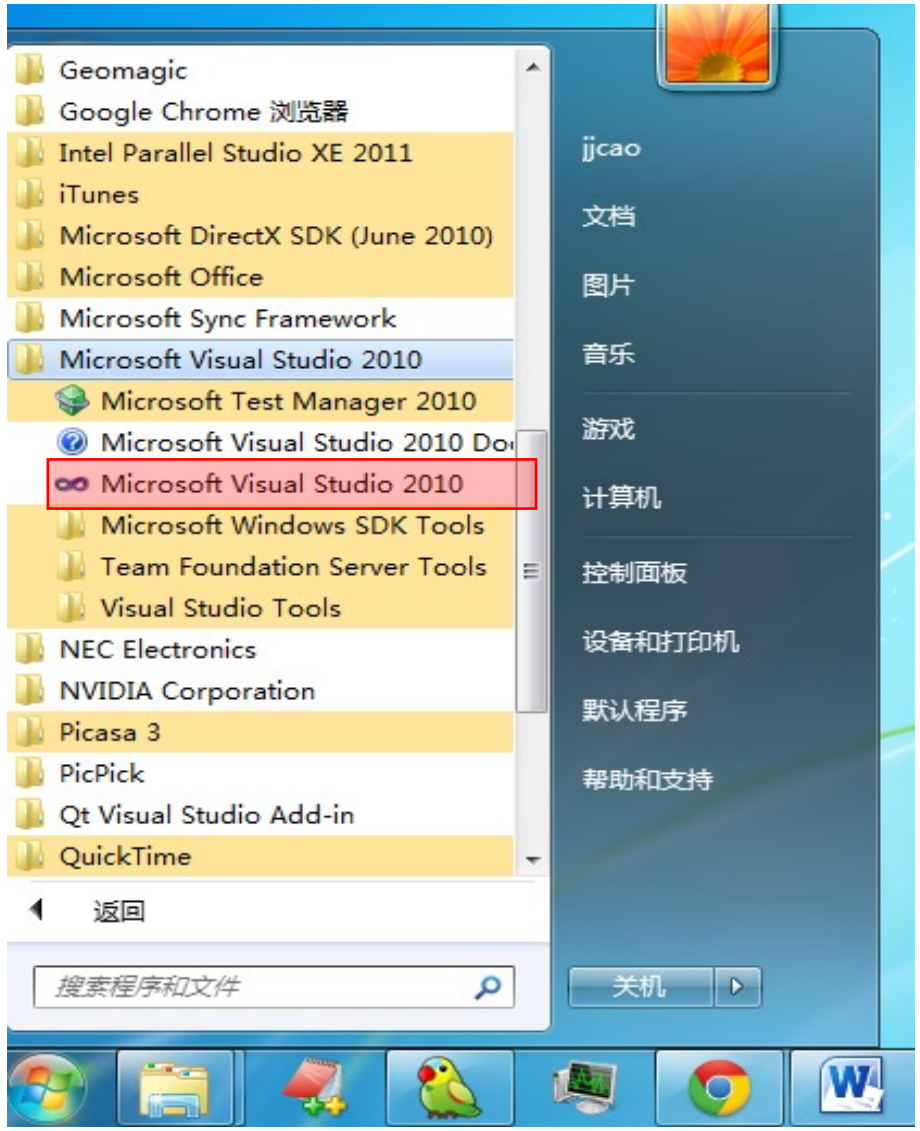
大纲

1. 建立控制台应用程序 (Setting up a Console Application in VC 10)
2. 命令行参数 Command Line Argument
3. 其它

1. Setting up a Console Application in VC 10

① Create a Console Project(项目)
② Solution(工程) Explorer
③ Add New Source File
④ Add Code
⑤ Class View
⑥ Setup Intermediate Directory (Optional)
⑦ Build Project
⑧ Run the Program

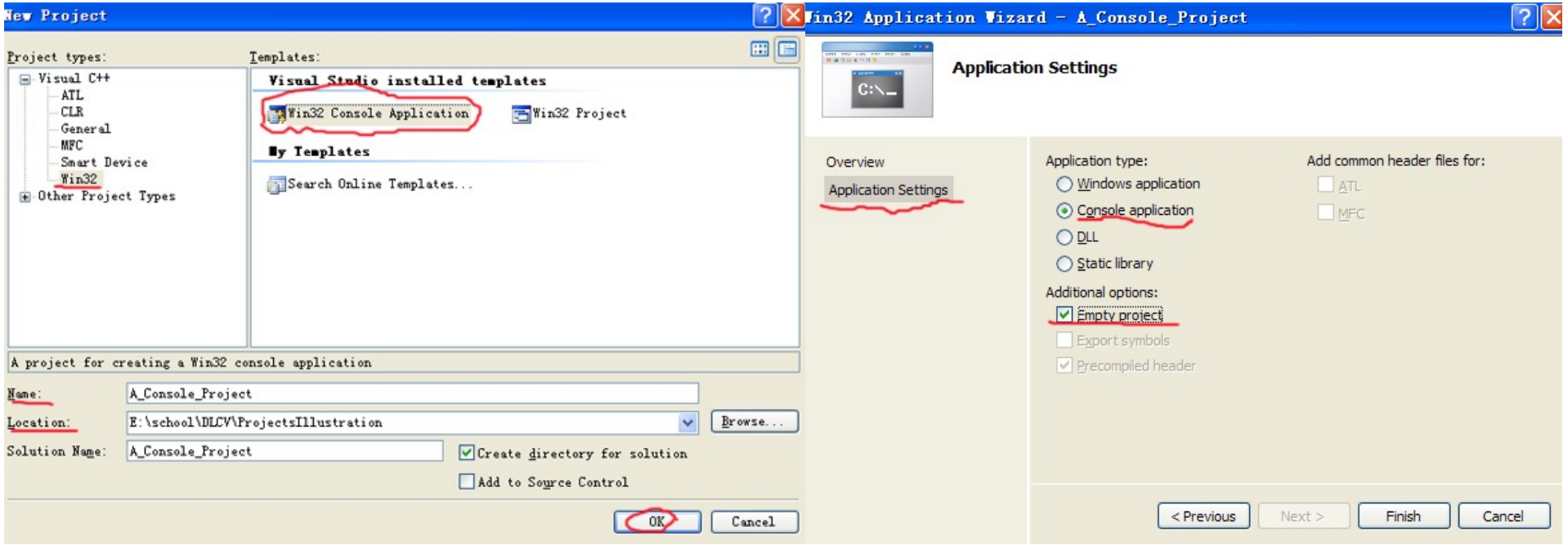
Start VC IDE


Geomagic
Google Chrome 浏览器
Intel Parallel Studio XE 2011
iTunes
Microsoft DirectX SDK (June 2010)
Microsoft Office
Microsoft Sync Framework
Microsoft Visual Studio 2010
Microsoft Test Manager 2010
Microsoft Visual Studio 2010 Do
Microsoft Visual Studio 2010
Microsoft Windows SDK Tools
Team Foundation Server Tools
Visual Studio Tools
NEC Electronics
NVIDIA Corporation
Picasa 3
PicPick
Qt Visual Studio Add-in
QuickTime
返回
搜索程序和文件
jjcao
文档
图片
音乐
游戏
计算机
控制面板
设备和打印机
默认程序
帮助和支持
关机


Microsoft
Visual
Studio 2010

Step 1: Create a Console Project

1. Choose **File -> New -> Project** from the VS210 menus => **Win32 Console Application** (chosen from the templates on the right side).
2. Set the location to someplace on your drive and give the project a name, such as **A_Console_Project**.
3. Click **OK**.
4. Go to the **Application Settings** tab. Make sure **Console application** and **Empty project** are selected.
5. Click **Finished**.

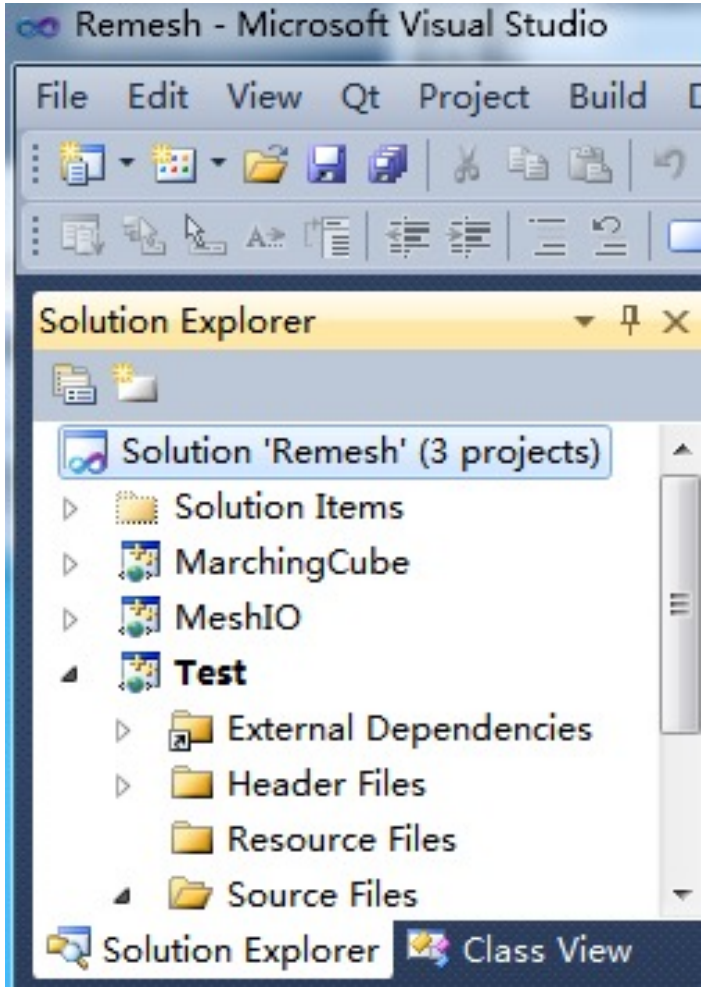


Step 2: Solution Explorer

1. Choose **View- Solution Explorer** from the VS menus

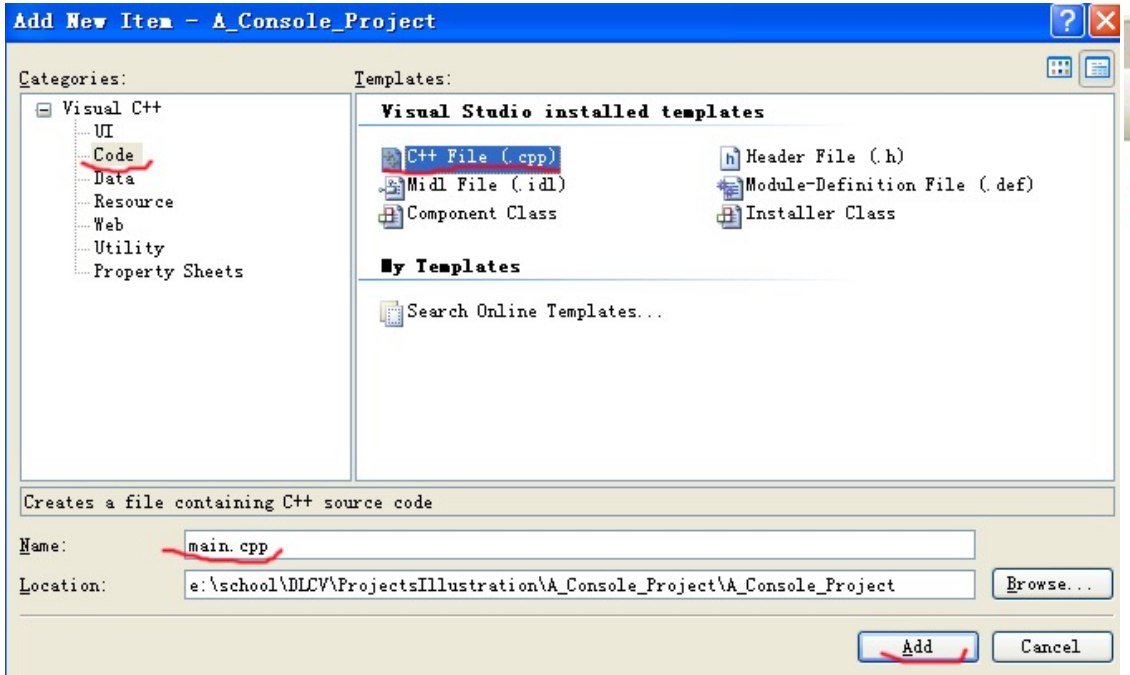
2. It shows you a **tree** representing your current solution.

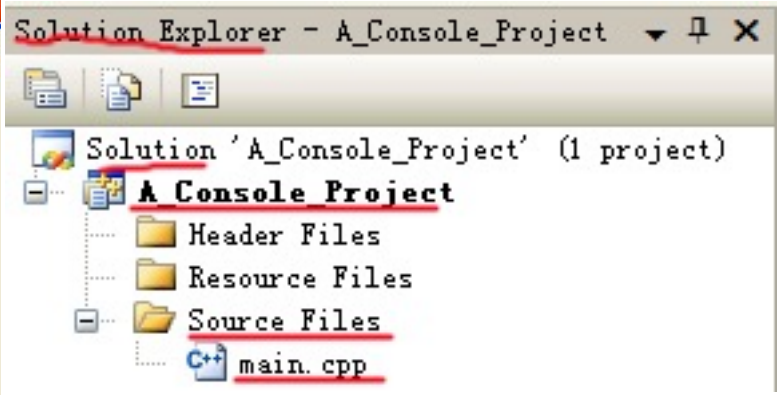
3. **Solutions** are made of one or more **projects**, which in turn are composed of one or more files. We currently have one project in our solution, namely **A_Console_Project**



Step 3: Add New Source File

1. Right click on our project **A_Console_Project** at **solution explorer** and choose **Add->Add New Item**.
2. We want a C++ source file so choose the **Code** category and **C++ File (.cpp)** from the **Templates**.
3. Enter a name for our source file, **main.cpp** in our case. The location should already be under the project you've created.
4. Click **Add**.
5. You should now see a blank source file in the **editor** titled main.cpp.
6. **Solution Explorer** shows that **A_Console_Project** has currently a single file **main.cpp** under **Source Files**.





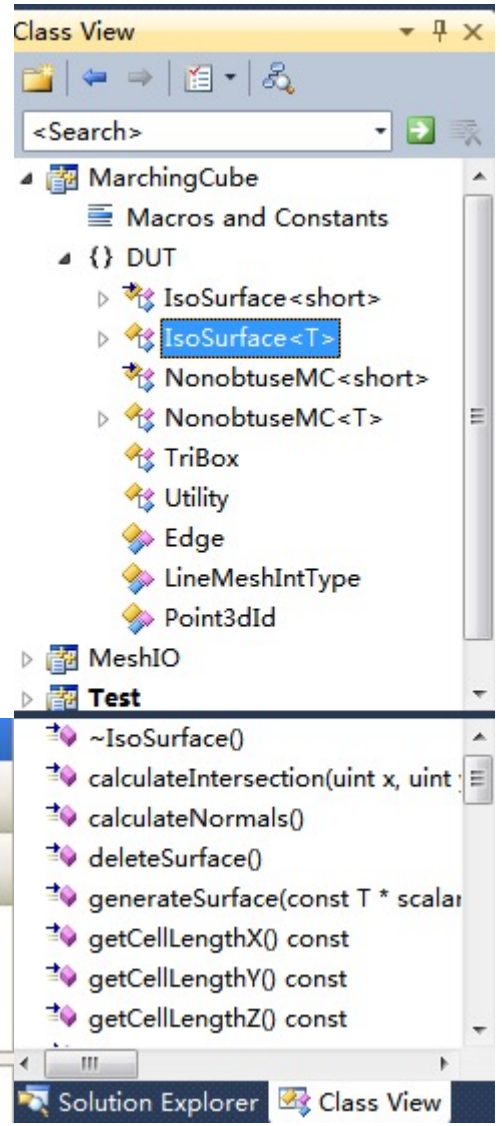
Step 4: Add Code

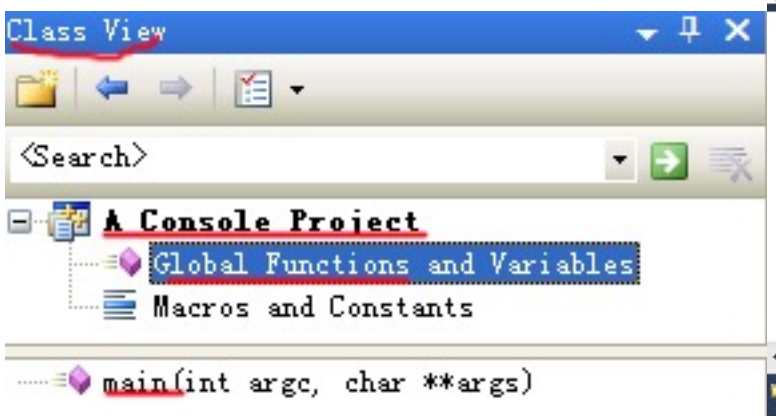
Enter the following code into **main.cpp** which should already be open in the main source window.

```
#include <iostream>
using namespace std;
int main(int argc, char** args){
        cout << "Hello world" << endl;
        return 0;
}
```

Step 5: Class View

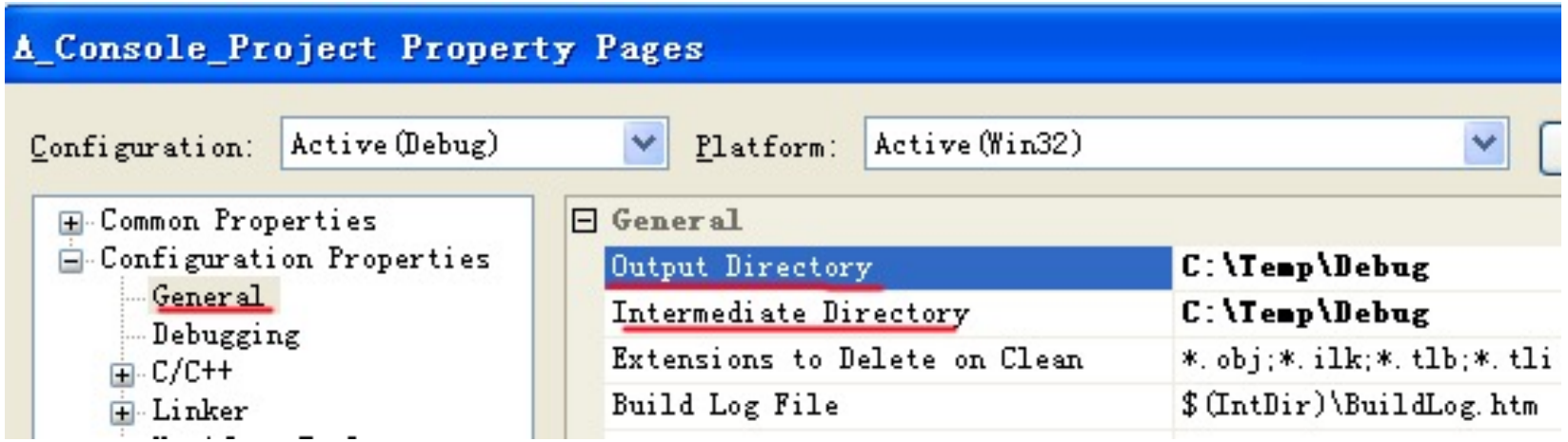
1. We have a single function: **main()** currently, so it is easy to locate the code. However, the project can become rather large and finding the function you want to modify may not be so easy.
2. Choose **View -> Class View** => **Class View** window, which is similar to **Solution Explorer**. Hence you'll probably want to **place them together**.
3. It breaks the project up by **class and function rather than by file**.
   - Right now we have only a single function **main**
   - but later your projects will grow to **multiply classes each containing many functions**.
   - The Class View window will prove invaluable when navigating such projects later in the course.





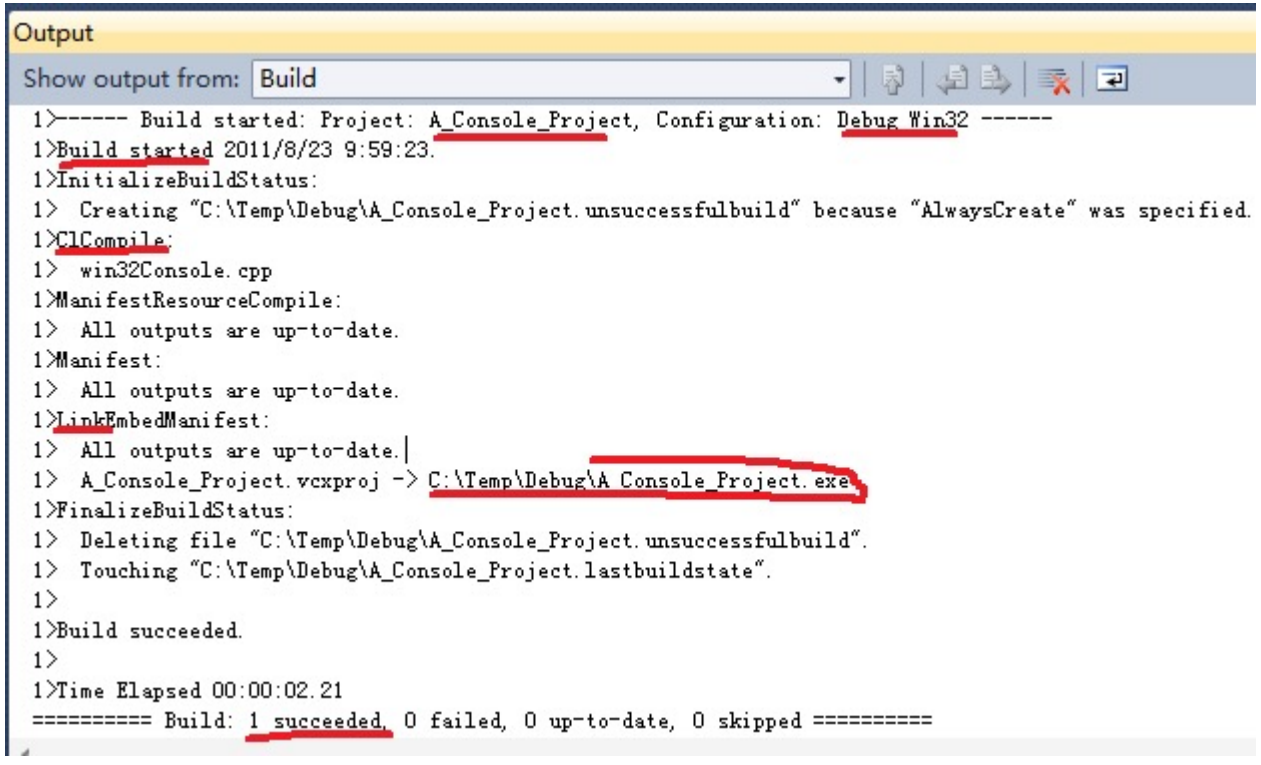
Step 6: Setup Intermediate Directory (Optional)

1. Open the Class View window, right click on our project **A_Console_Project** and choose **Properties**. Select **General** from the left hand pane and set the **Output Directory** and **Intermediate Directory** both to **C:\Temp\Debug**.
2. Click OK and we're all set to compile our project.



Step 7: Build (compile+link) Project

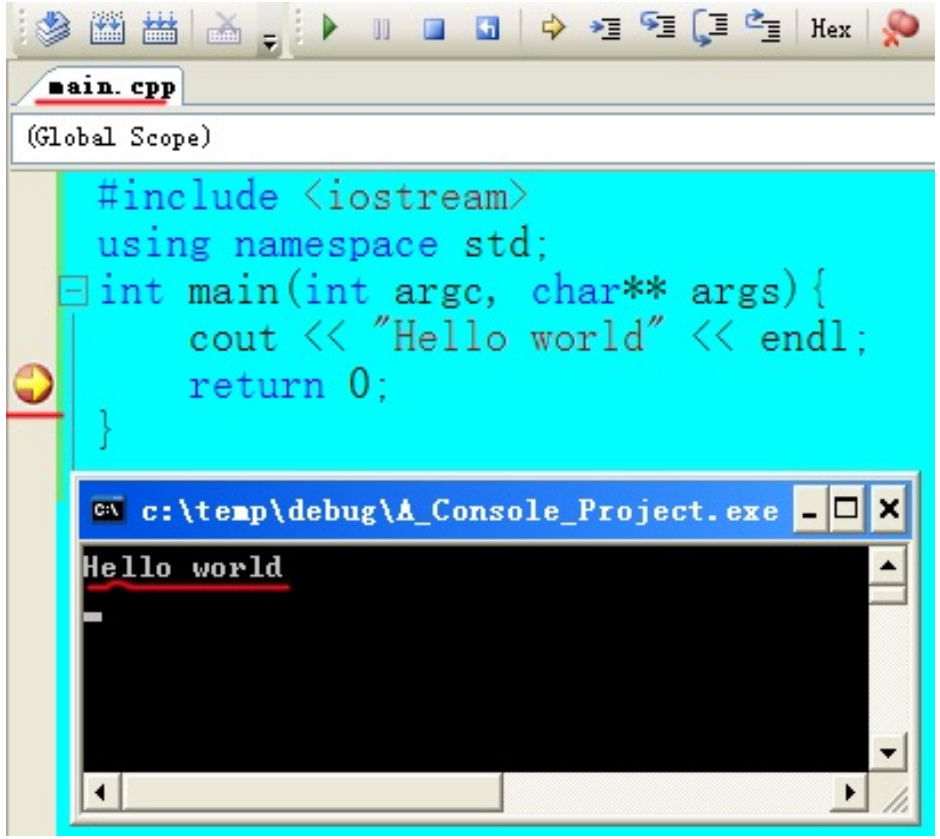
1. **View->Other Windows->Output** from VS2010 menus. The **Output** window shows the output from the compile and linking process.
2. Chose **Build->Build Solution (F7)** from the VS2010 menus.
3. If you've copied the code correctly, you should see the following output to the right.
4. If the build failed you should see some text explaining why it failed. If the error is a **compiling error**, you should be able to double click on the error and the source window should update to show you the line on which the error occurred.



Step 8: Run the Program

- Place a **breakpoint** on the line “return 0” by left clicking on the position of the **big red point** (left click again, it will disappear) and press **F5**.
- A console window appears and the program will break at the line “return 0”.
- Press **F5** again to continue the execution of the program, and the program will exist successfully.

Congratulations!



Questions?

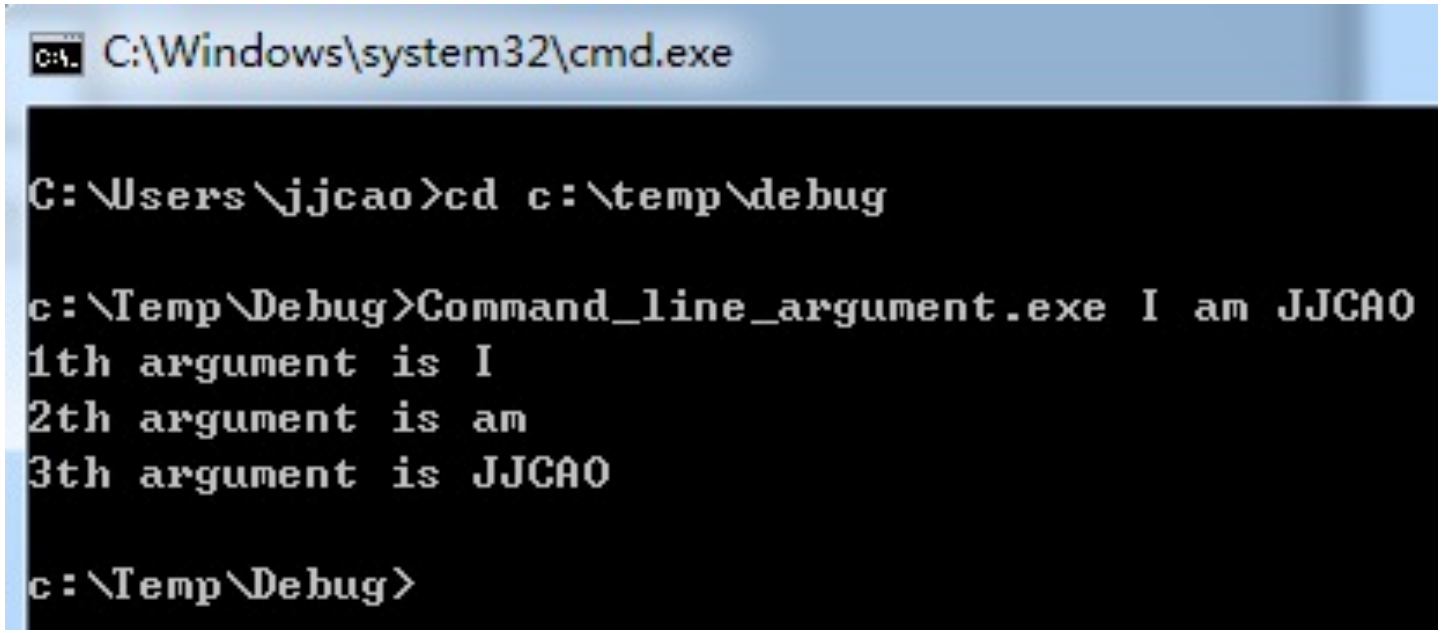
2. Command Line Argument命令行参数

1.In Windows, the GUI will be used to communicate with the user.

2.In DOS, command line arguments are used to parse in users' specified parameters.

3.In the following figure, "**Command_line_argument.exe**" is the command and **I am JJCAO** is the command line arguments being parsed in.

1. Modify Main
2. Modify Project's Command Arguments
3. Build Project
4. Run the program



Step 1: Modify Main

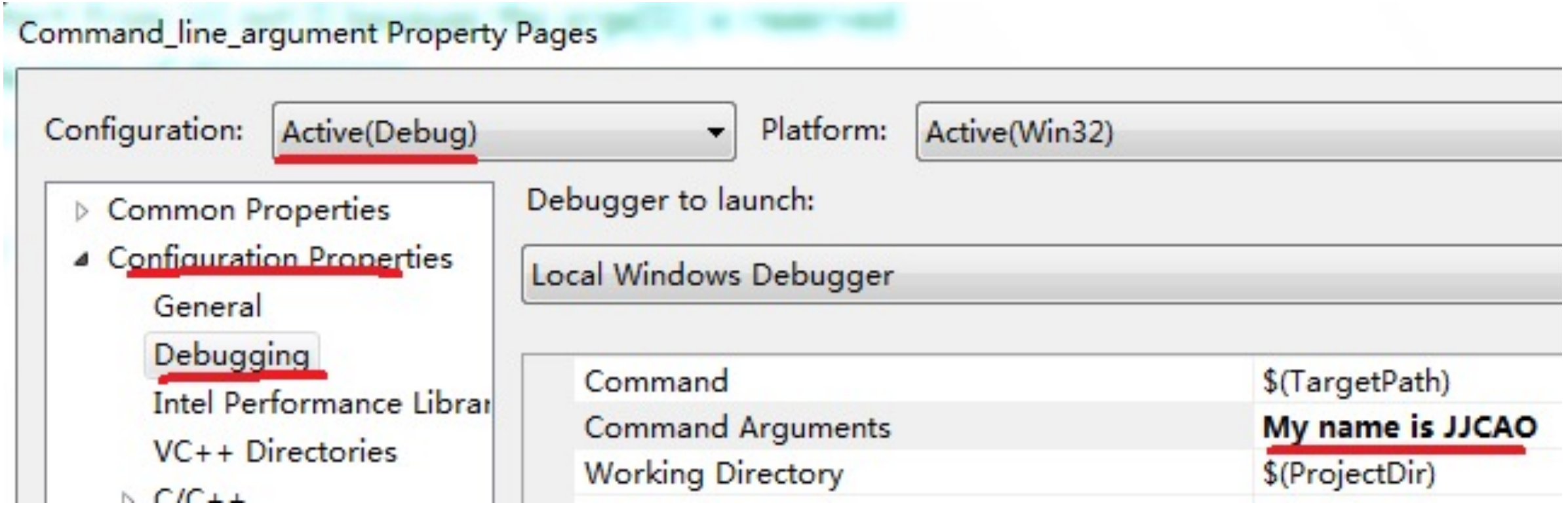
```
#include <iostream>
using namespace std;
int main(int argc, char** args)
{
// Notice i start from 1 not 0 because the args[0] is reserved for the name of this program.
        for(int i = 1; i < argc; i++)
        {
            cerr << i << "th argument is " << args[i] << "\n";
        }
}
```

Note: The relationship of argc and args:

1. args is **an array of char***
2. argc is the size of the array: args, which is determined when command line arguments are passed to the main() function. So after you change the size of args, argc is not updated automatically.

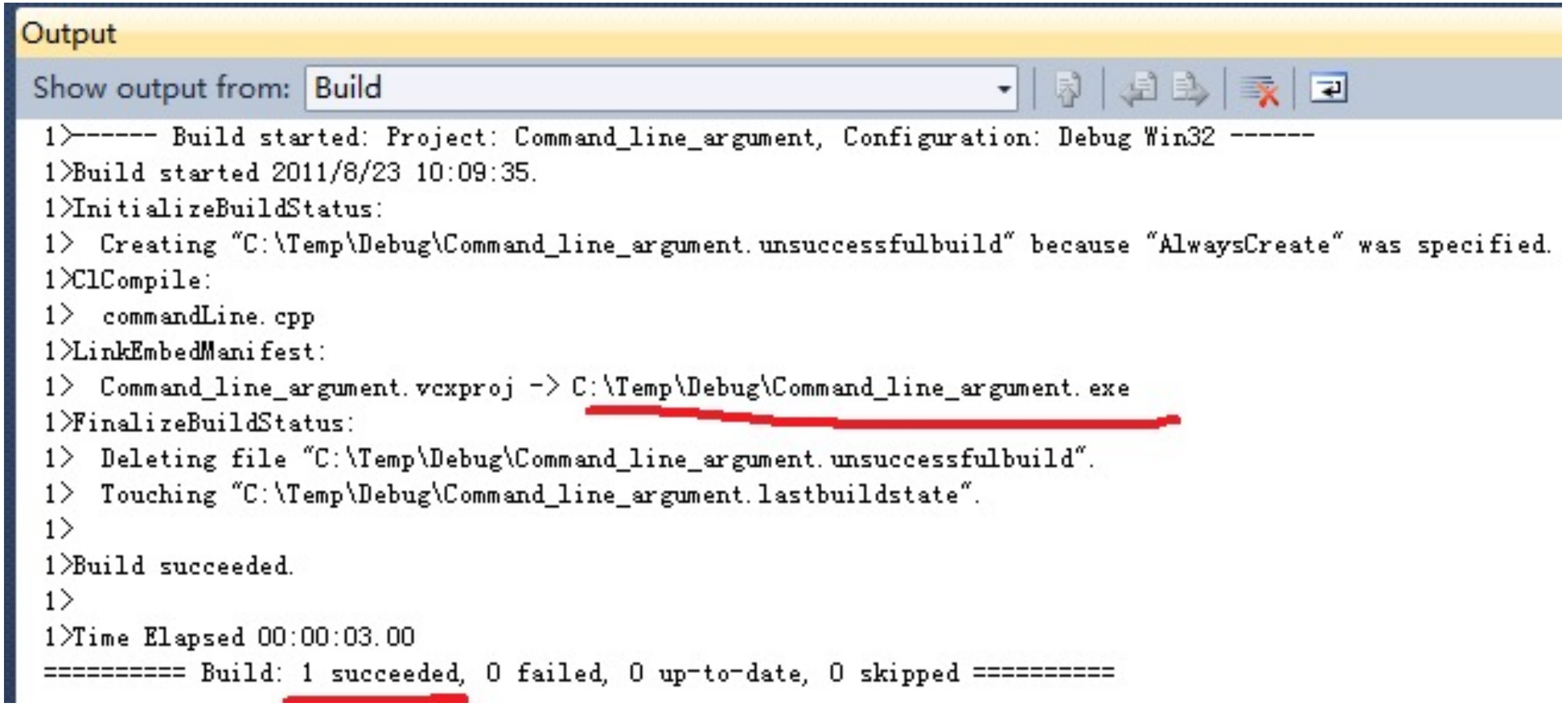
Step 2: Modify Project's Command Arguments Setting

Open the **Class View** window, right click on our project (**Command_line_argument**), and choose **Properties**. Choose **Debugging** from the left hand pane. Add **My name is JJCAO** to the **Command Arguments.** Click **OK**



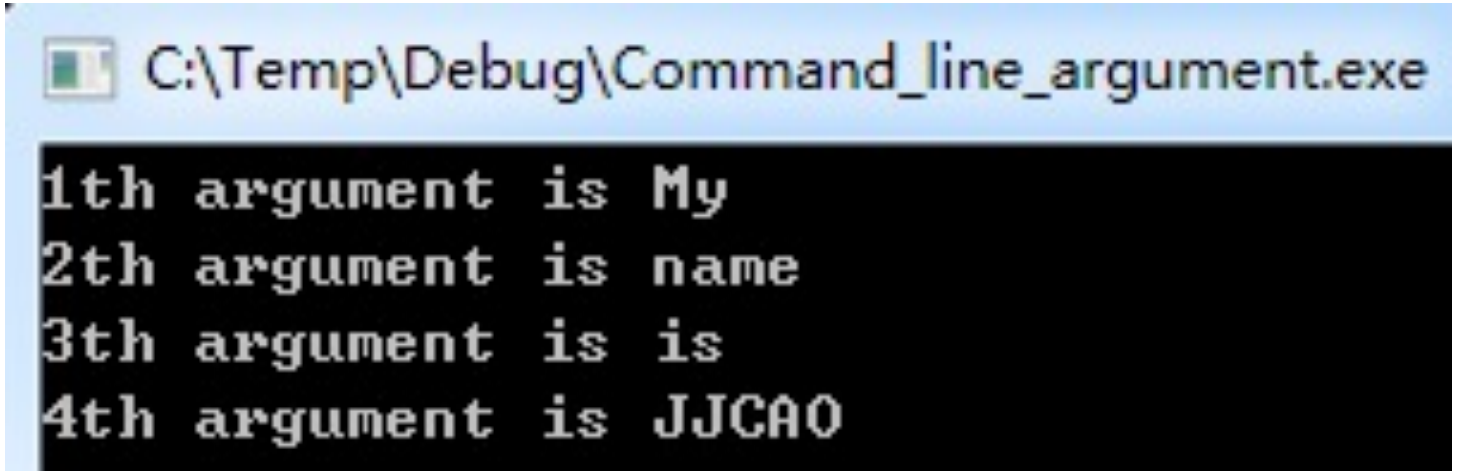
Step 3: Build Project

- Press **F7** to build the solution.



Step 4: Run the Program

- Press **F5** to run the program..
- Congratulations! You've got your 2nd successful VS2010 project



- Imagine that you wish to read a txt file: someinfo.txt, can your program find the file correctly?
  - When debugging a c++ program from the IDE (F5), the **current path is the path where the project file (*.vcproj) is located**.
  - When running a c++ program from the IDE (Ctrl+F5), the current path is the path where the excutable file (*.exe) is located (commonly is the path **Debug**).
- Relative file path vs absolute path

Questions

Input

- Let user input values to the program (line 6)

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5    int x;
6    cin >> x;
7
8    cout << x / 3 << ' ' << x * 2;
9
10   return 0;
11 }
```

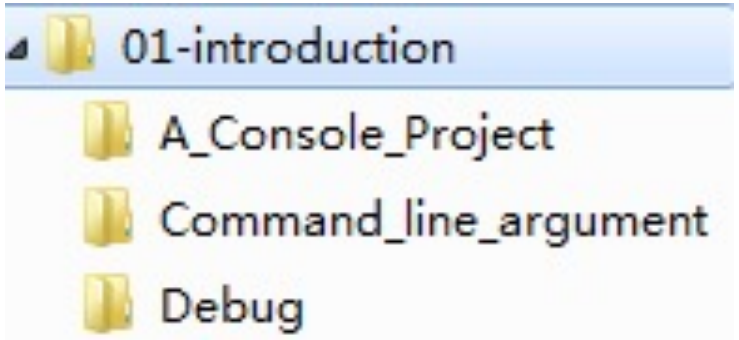
- ctrl+z: cancel input from cin

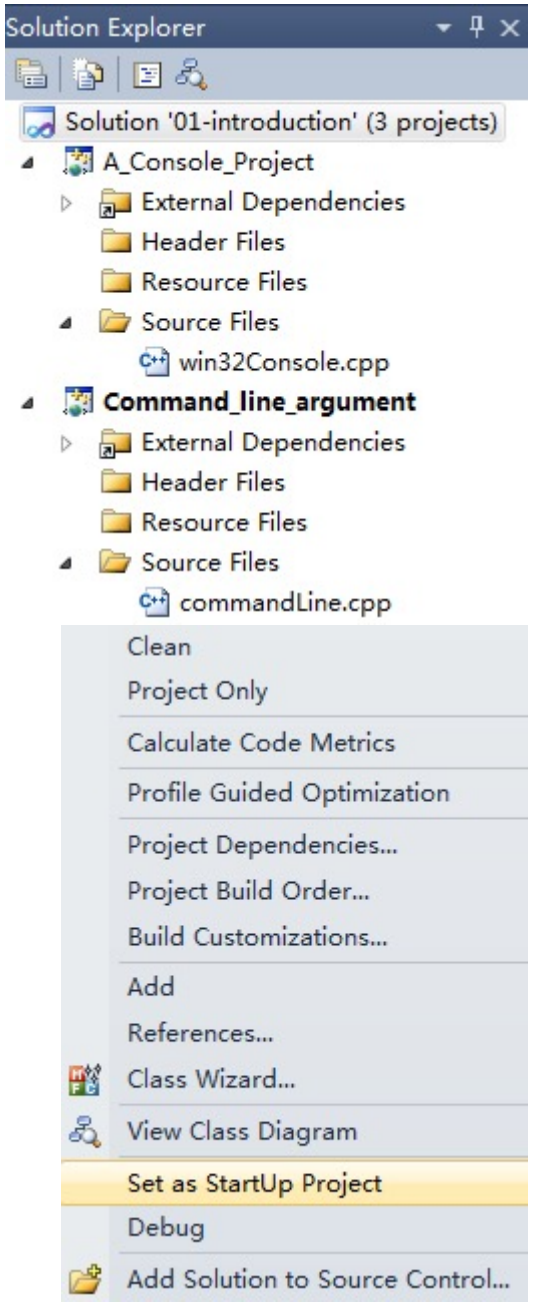
iostream

- cin
- cout
- cerr
- clog
- Ordinarily, sys associates them with the console window.
- They can be redirected to files.

Default current directory of VC

- **Solution**: 01-Introduction
  - Project: A_Console_Project
  - Project: Command_line_argument





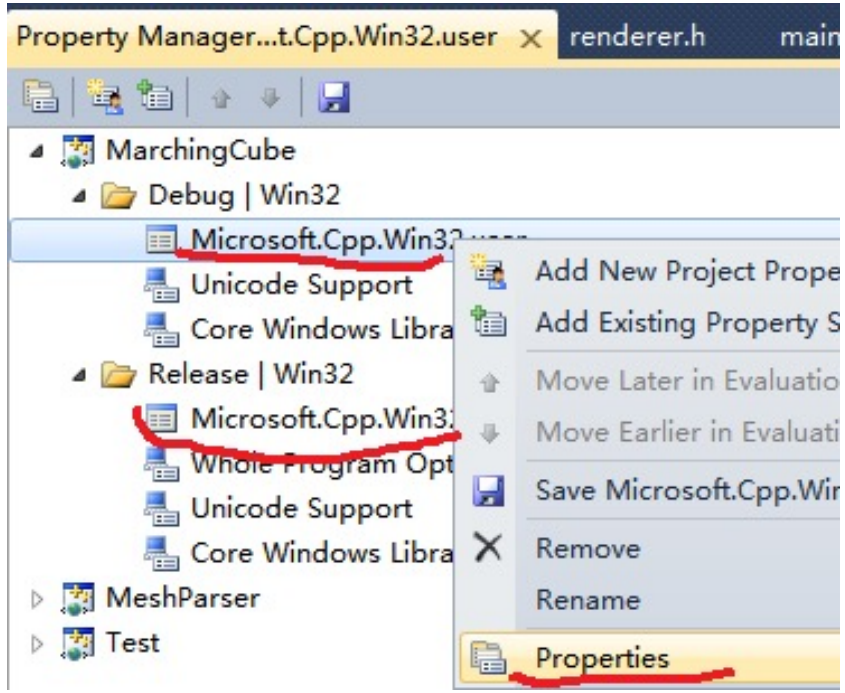
- **Current Project**: Command_line_argument
- The **current directory** of the current project
  - The dir where the Command_line_argument.vcxproj is
  - Where is win32Console.cpp?
    ../ A_Console_Project/

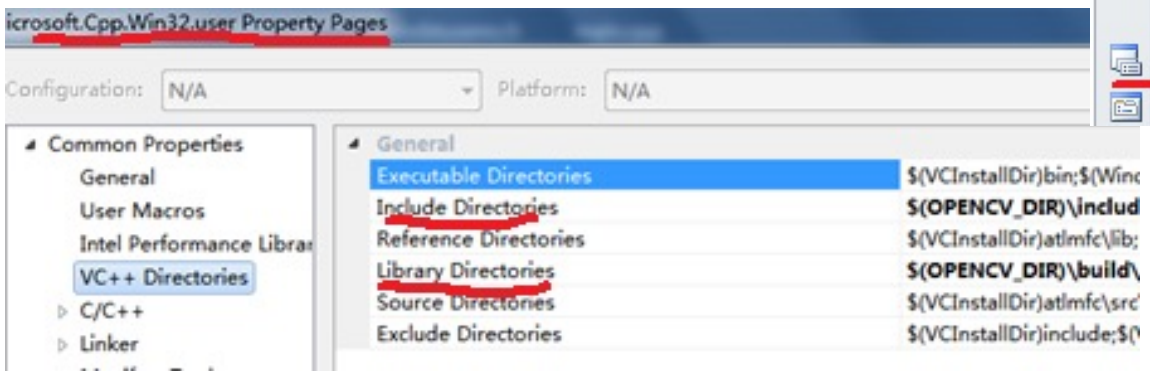
Debugging

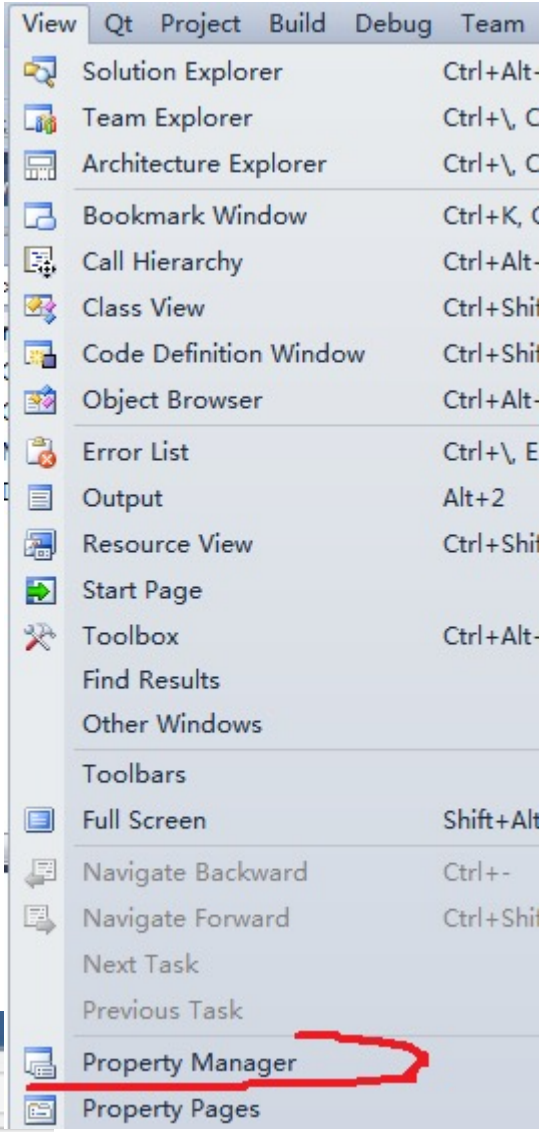
- Compilation error
  - Violations of the syntax rules
  - Misuse of types
- Runtime error
  - Need debugging

Set include & lib path independent with solutions

- Set it in Property Manager (You have to open a project first.)
- If you set it in the Context Menu of a solution or project, it will be dependent on specified projections.

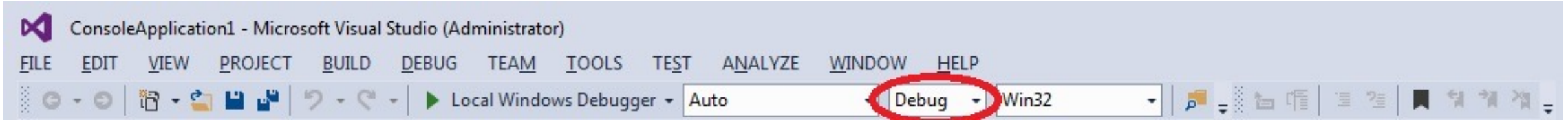






Build configurations

- Switching between debug and release in Visual Studio



Pack Your Solution
-- before sending it to others

1. Delete *.ncb || *.sdf
2. Delete debug, release && ipch directories
3. Compress all into a *.rar or *.zip
4. Sent it by email || …

代码打包工具packing.bat

Why vscode?

As usual, everyone was using the CodeBlocks IDE and Visual Studio IDE. But I was already used to Visual Studio Code

- A lightweight editor
- Versatile: c++, python, html, markdown, latex, …
- Powerful
- Cross-platform

其它C++环境设置参考

- https://www.runoob.com/cplusplus/cpp-environment-setup.html