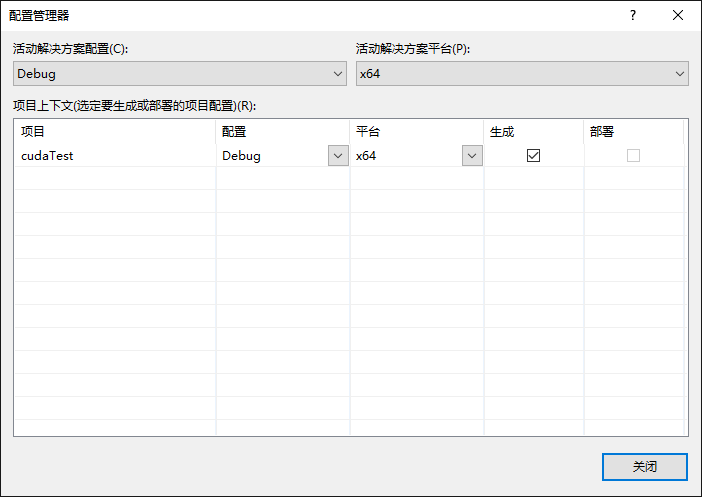
一、新建项目

　　打开VS2017 → 新建项目 → Win32控制台应用程序 → “空项目”打钩

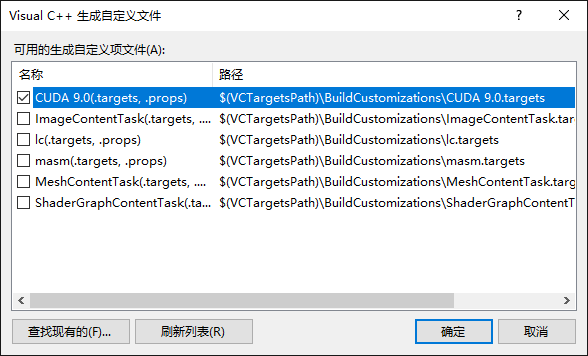
二、调整配置管理器平台类型

　　右键项目 → 属性 → 配置管理器 → 全改为“x64”



三、配置生成属性

　　右键项目 → 生成依赖项 → 生成自定义 → 勾选“CUDA 9.0XXX”



三、配置基本库目录

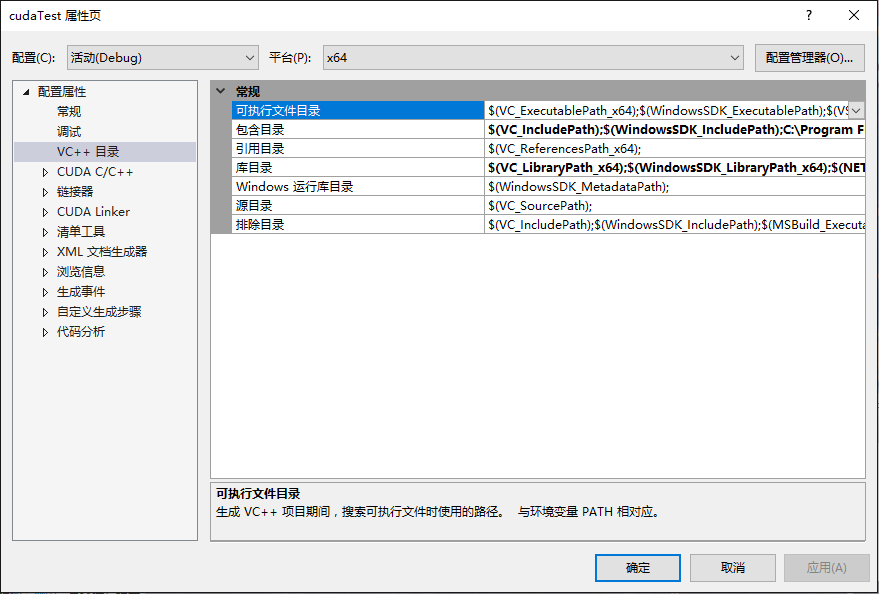
　　注意：后续步骤中出现的目录地址需取决于你当前的CUDA版本及安装路径

　　右键项目 → 属性 → 配置属性 → VC++目录 → 包含目录，添加以下目录：

* C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\include
* C:\ProgramData\NVIDIA Corporation\CUDA Samples\v9.0\common\inc

　　…… → 库目录，添加以下目录：

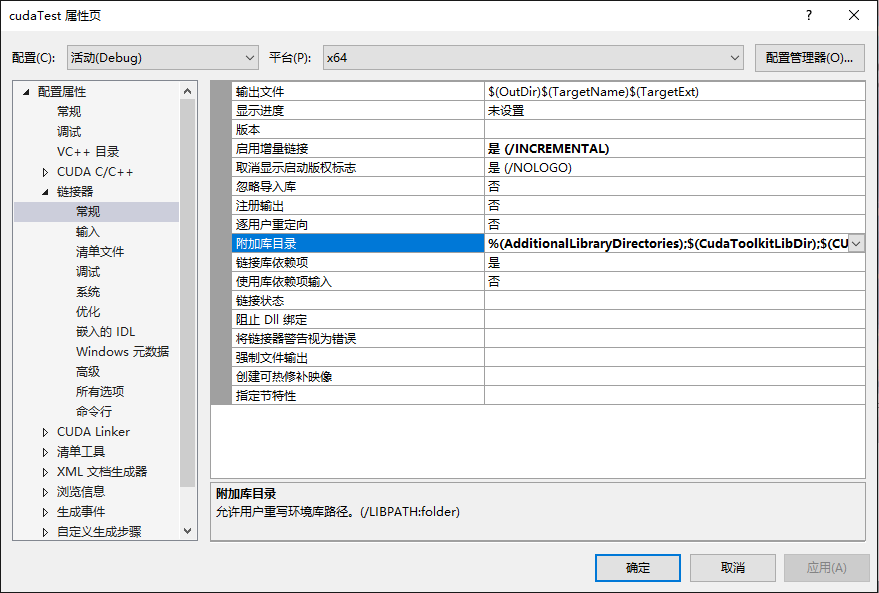
* C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\lib\x64
* C:\ProgramData\NVIDIA Corporation\CUDA Samples\v9.0\common\lib\x64



四、配置CUDA静态链接库路径

　　右键项目 → 属性 → 配置属性 → 链接器 → 常规 → 附加库目录，添加以下目录：

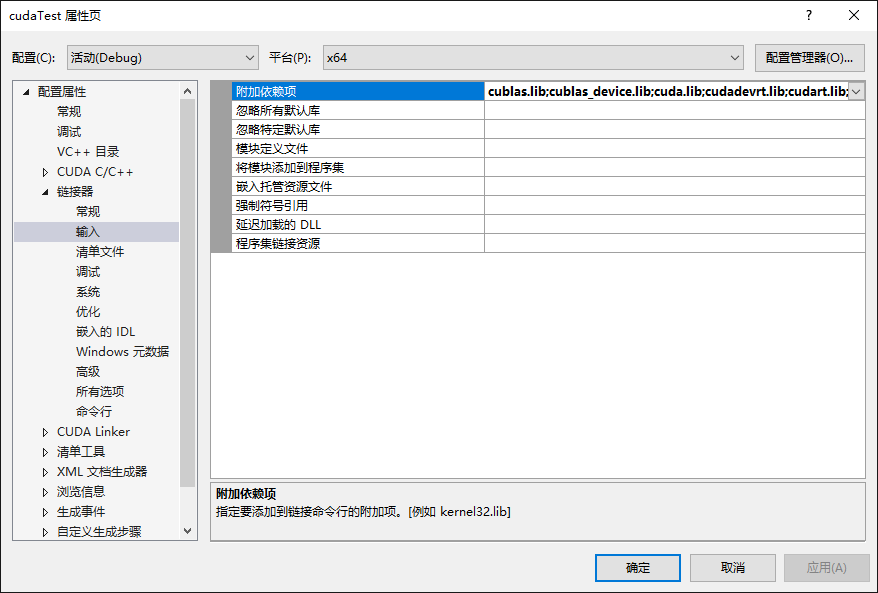
* $(CUDA\_PATH\_V9\_0)\lib\$(Platform)



五、选用CUDA静态链接库

　　右键项目 → 属性 → 配置属性 → 链接器 → 输入 → 附加依赖项，添加以下库：

* cublas.lib;cublas\_device.lib;cuda.lib;cudadevrt.lib;cudart.lib;cudart\_static.lib;cufft.lib;cufftw.lib;curand.lib;cusolver.lib;cusparse.lib;nppc.lib;nppial.lib;nppicc.lib;nppicom.lib;nppidei.lib;nppif.lib;nppig.lib;nppim.lib;nppist.lib;nppisu.lib;nppitc.lib;npps.lib;nvblas.lib;nvcuvid.lib;nvgraph.lib;nvml.lib;nvrtc.lib;OpenCL.lib;  
  以上为 “第三步” 中添加的库目录 “C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v9.0\lib\x64” 中的库！

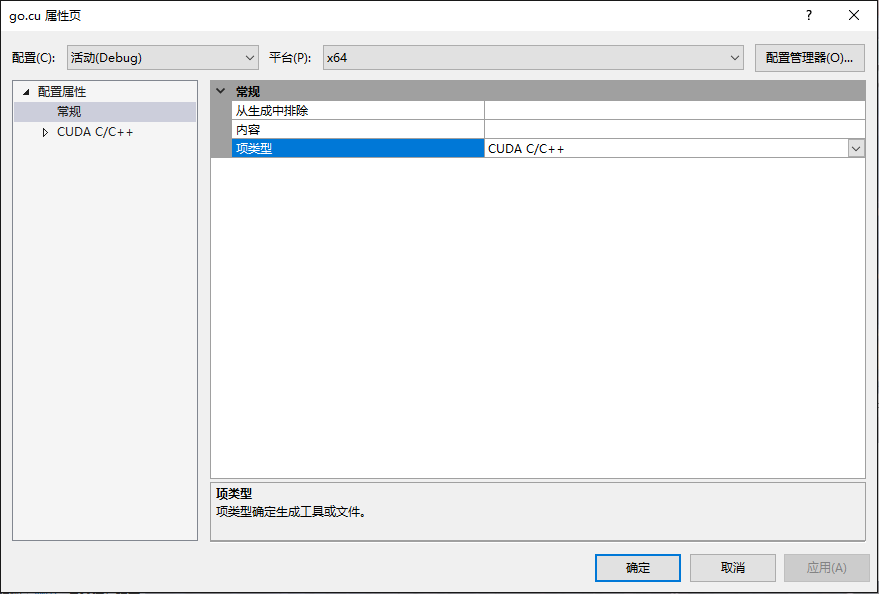


* 注意：kernel32.lib;user32.lib;gdi32.lib;winspool.lib;comdlg32.lib;advapi32.lib;shell32.lib;ole32.lib;oleaut32.lib;uuid.lib;odbc32.lib;odbccp32.lib;%(AdditionalDependencies)  
  这些库为原有！

六、配置源码文件风格

　　右键源文件 → 添加 → 新建项 → 选择 “CUDA C/C++ File”

　　右键 “xxx.cu" 源文件 → 属性 → 配置属性 → 常规 → 项类型 → 设置为“CUDA C/C++”



//警告4819

-Xcompiler "/wd4819"

七、编写测试程序

[复制代码](javascript:void(0);)

1 #include "cuda\_runtime.h"

2 #include "device\_launch\_parameters.h"

3 #include <stdio.h>

4

5 int main() {

6 int deviceCount;

7 cudaGetDeviceCount(&deviceCount);

8

9 int dev;

10 for (dev = 0; dev < deviceCount; dev++)

11 {

12 int driver\_version(0), runtime\_version(0);

13 cudaDeviceProp deviceProp;

14 cudaGetDeviceProperties(&deviceProp, dev);

15 if (dev == 0)

16 if (deviceProp.minor = 9999 && deviceProp.major == 9999)

17 printf("\n");

18 printf("\nDevice%d:\"%s\"\n", dev, deviceProp.name);

19 cudaDriverGetVersion(&driver\_version);

20 printf("CUDA驱动版本: %d.%d\n", driver\_version / 1000, (driver\_version % 1000) / 10);

21 cudaRuntimeGetVersion(&runtime\_version);

22 printf("CUDA运行时版本: %d.%d\n", runtime\_version / 1000, (runtime\_version % 1000) / 10);

23 printf("设备计算能力: %d.%d\n", deviceProp.major, deviceProp.minor);

24 printf("Total amount of Global Memory: %u bytes\n", deviceProp.totalGlobalMem);

25 printf("Number of SMs: %d\n", deviceProp.multiProcessorCount);

26 printf("Total amount of Constant Memory: %u bytes\n", deviceProp.totalConstMem);

27 printf("Total amount of Shared Memory per block: %u bytes\n", deviceProp.sharedMemPerBlock);

28 printf("Total number of registers available per block: %d\n", deviceProp.regsPerBlock);

29 printf("Warp size: %d\n", deviceProp.warpSize);

30 printf("Maximum number of threads per SM: %d\n", deviceProp.maxThreadsPerMultiProcessor);

31 printf("Maximum number of threads per block: %d\n", deviceProp.maxThreadsPerBlock);

32 printf("Maximum size of each dimension of a block: %d x %d x %d\n", deviceProp.maxThreadsDim[0],

33 deviceProp.maxThreadsDim[1],

34 deviceProp.maxThreadsDim[2]);

35 printf("Maximum size of each dimension of a grid: %d x %d x %d\n", deviceProp.maxGridSize[0], deviceProp.maxGridSize[1], deviceProp.maxGridSize[2]);

36 printf("Maximum memory pitch: %u bytes\n", deviceProp.memPitch);

37 printf("Texture alignmemt: %u bytes\n", deviceProp.texturePitchAlignment);

38 printf("Clock rate: %.2f GHz\n", deviceProp.clockRate \* 1e-6f);

39 printf("Memory Clock rate: %.0f MHz\n", deviceProp.memoryClockRate \* 1e-3f);

40 printf("Memory Bus Width: %d-bit\n", deviceProp.memoryBusWidth);

41 }

42

43 return 0;

44 }

