



Parallel Processing - CUDA

2016. 3. 17



What is CUDA?

- Compute Unified Device Architecture
- General Purpose Programming Model
- Programmable in C with CUDA tools
- Multithreaded SPMD models uses application data parallelism and thread parallelism

Installation

■ <https://developer.nvidia.com/cuda-toolkit-42-archive>

The screenshot shows the NVIDIA Developer Zone website. The main navigation bar includes links for Log In, Feedback, and New Account. Below the navigation bar, the 'CUDA Downloads' section is highlighted. The 'CUDA TOOLKIT 4.2' section is outlined in red. It contains three main download paths: 1. Download Toolkit (64bit, 32bit), 2. Download Drivers (V 301.32 Or 301.27) for Win7/Vista Desktop, Win7/Vista Notebook, and Win XP Desktop (64bit, 32bit), and 3. Download SDK (64bit, 32bit). Below this, the 'CUDA FOR LINUX' section is shown, with download links for various Linux distributions (Fedora, Redhat, Ubuntu, OpenSUSE, SUSE) and their respective versions (14, 5.5, 6.0, 11.04, 10.04, 11.2, Server 11 SP1). The 'Download Drivers (ver 295.41)' and 'Download SDK' links are also present. The right sidebar contains 'QUICKLINKS' and 'FEATURED ARTICLES'.

CUDA Downloads
CUDA TOOLKIT 4.2

CUDA FOR WINDOWS

- 1 Download Toolkit
 - 64bit
 - 32bit
- 2 Download Drivers (V 301.32 Or 301.27)
 - Win7/Vista Desktop
 - 64bit
 - 32bit
 - Win7/Vista Notebook
 - 64bit
 - 32bit
 - Win XP Desktop
 - 64bit
 - 32bit
- 3 Download SDK
 - 64bit
 - 32bit

CUDA FOR LINUX

- 1 Download Toolkit
 - Fedora 14
 - 64bit
 - 32bit
 - Redhat 5.5
 - 64bit
 - 32bit
 - 6.0
 - 64bit
 - Ubuntu 11.04
 - 64bit
 - 32bit
 - 10.04
 - 64bit
 - 32bit
 - OpenSUSE 11.2
 - 64bit
 - 32bit
 - SUSE Server 11 SP1
 - 64bit
 - 32bit
- 2 Download Drivers(ver 295.41)
 - 64bit
 - 32bit
- 3 Download SDK
 - DOWNLOAD

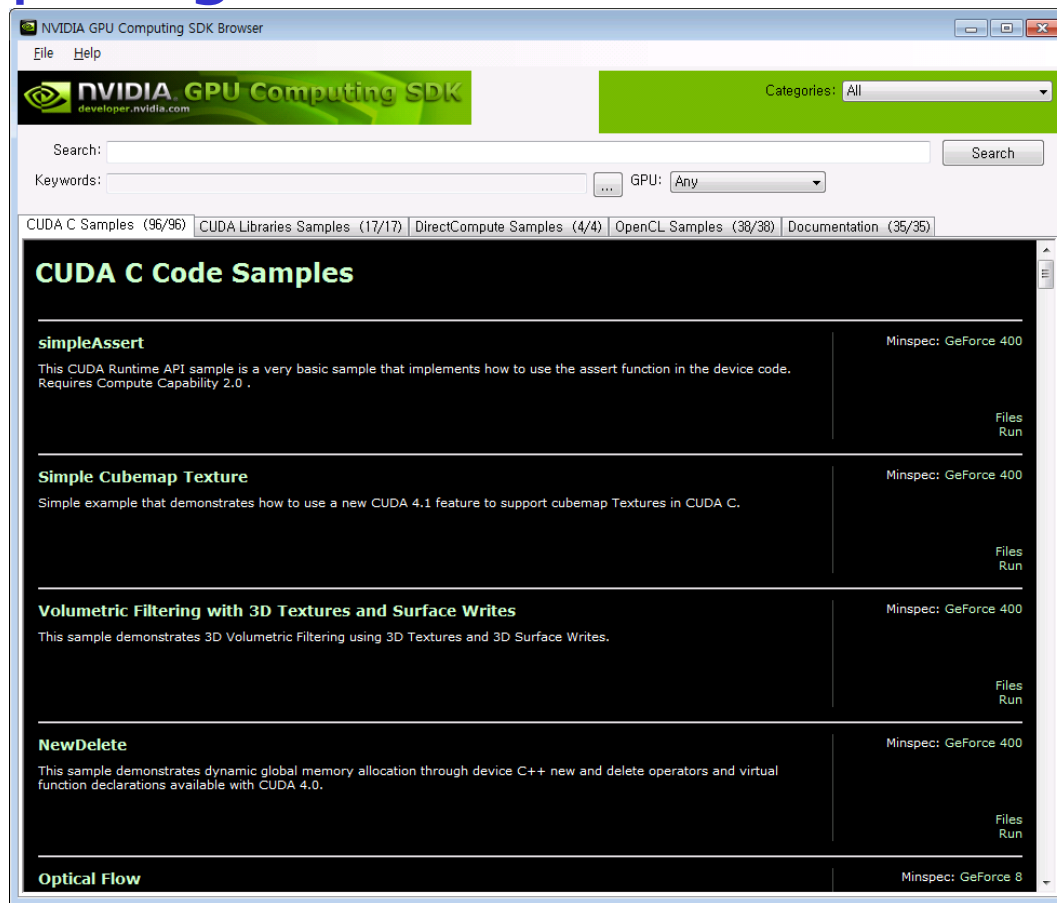


Installation

- 3개의 파일을 다운받아 설치
- Developer Driver 먼저 설치
- CUDA toolkit 설치
- SDK 설치

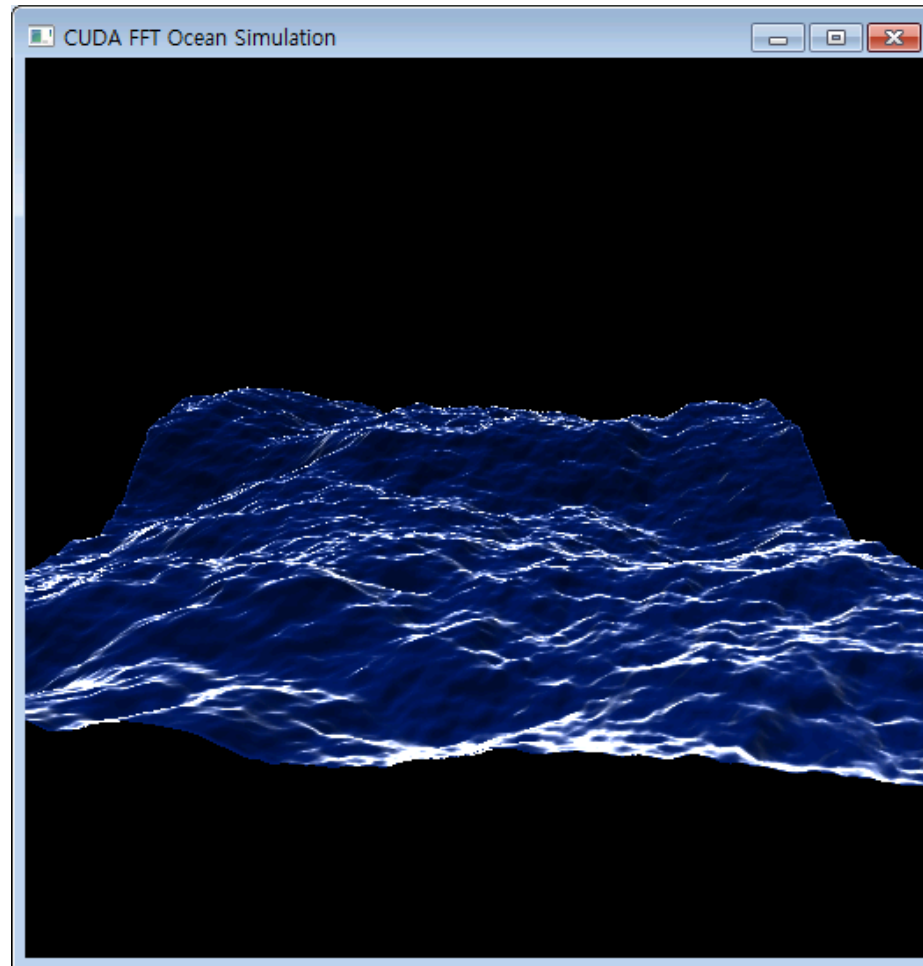
Installation

■ 3개의 파일 설치 완료 후 NVIDIA GPU Computing SDK Browser 실행



Installation

■ CUDA FFT Ocean Simulation 실행

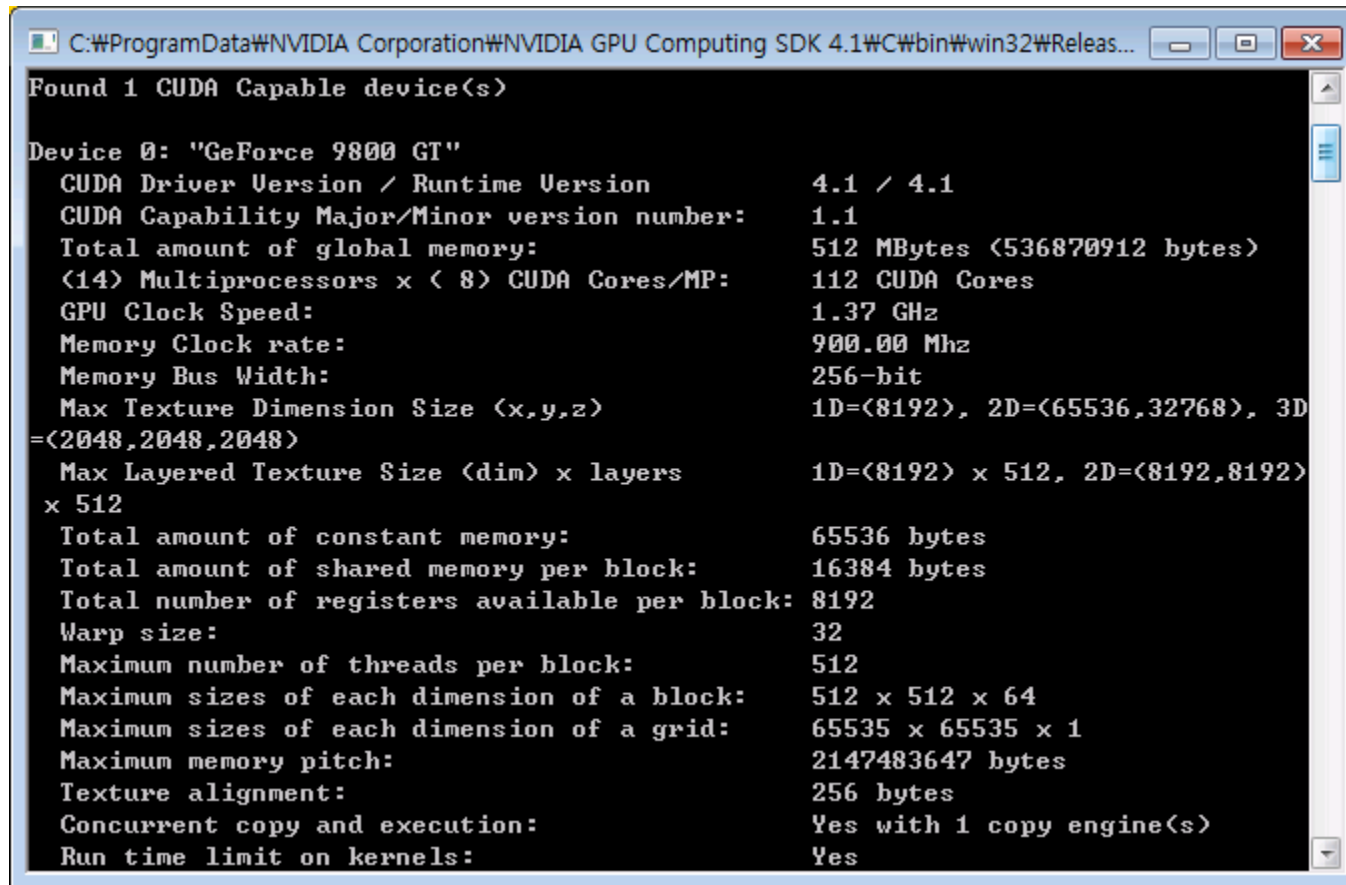


Parallel Processing



Installation

■ Device Query 실행

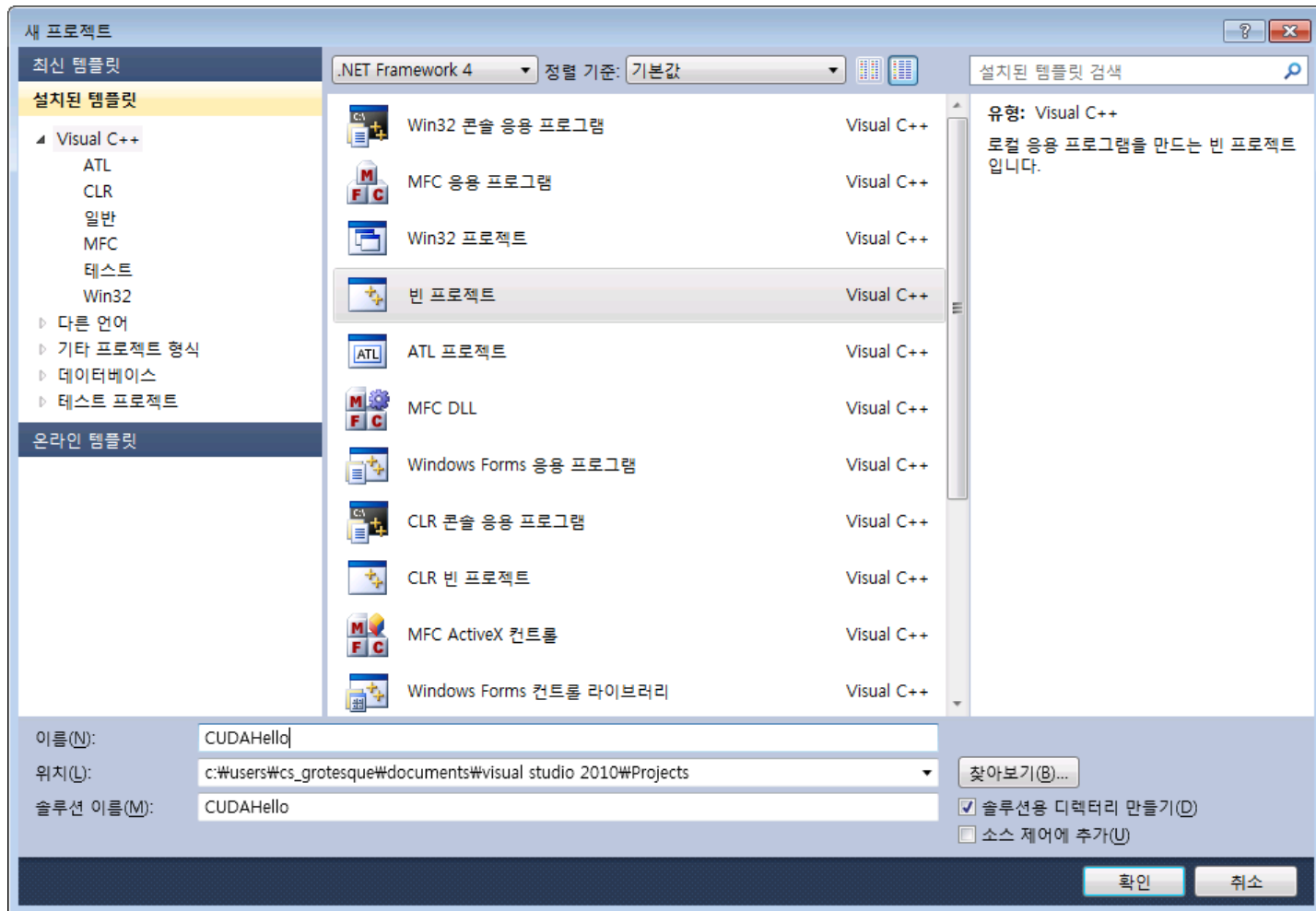


```
C:\ProgramData\NVIDIA Corporation\NVIDIA GPU Computing SDK 4.1\bin\win32\Releas...
Found 1 CUDA Capable device(s)

Device 0: "GeForce 9800 GT"
  CUDA Driver Version / Runtime Version      4.1 / 4.1
  CUDA Capability Major/Minor version number: 1.1
  Total amount of global memory:              512 MBytes (536870912 bytes)
  (14) Multiprocessors x (8) CUDA Cores/MP:  112 CUDA Cores
  GPU Clock Speed:                           1.37 GHz
  Memory Clock rate:                          900.00 Mhz
  Memory Bus Width:                           256-bit
  Max Texture Dimension Size (x,y,z)          1D=(8192), 2D=(65536,32768), 3D
=(2048,2048,2048)
  Max Layered Texture Size (dim) x layers      1D=(8192) x 512, 2D=(8192,8192)
x 512
  Total amount of constant memory:             65536 bytes
  Total amount of shared memory per block:     16384 bytes
  Total number of registers available per block: 8192
  Warp size:                                   32
  Maximum number of threads per block:         512
  Maximum sizes of each dimension of a block:  512 x 512 x 64
  Maximum sizes of each dimension of a grid:    65535 x 65535 x 1
  Maximum memory pitch:                       2147483647 bytes
  Texture alignment:                           256 bytes
  Concurrent copy and execution:              Yes with 1 copy engine(s)
  Run time limit on kernels:                   Yes
```

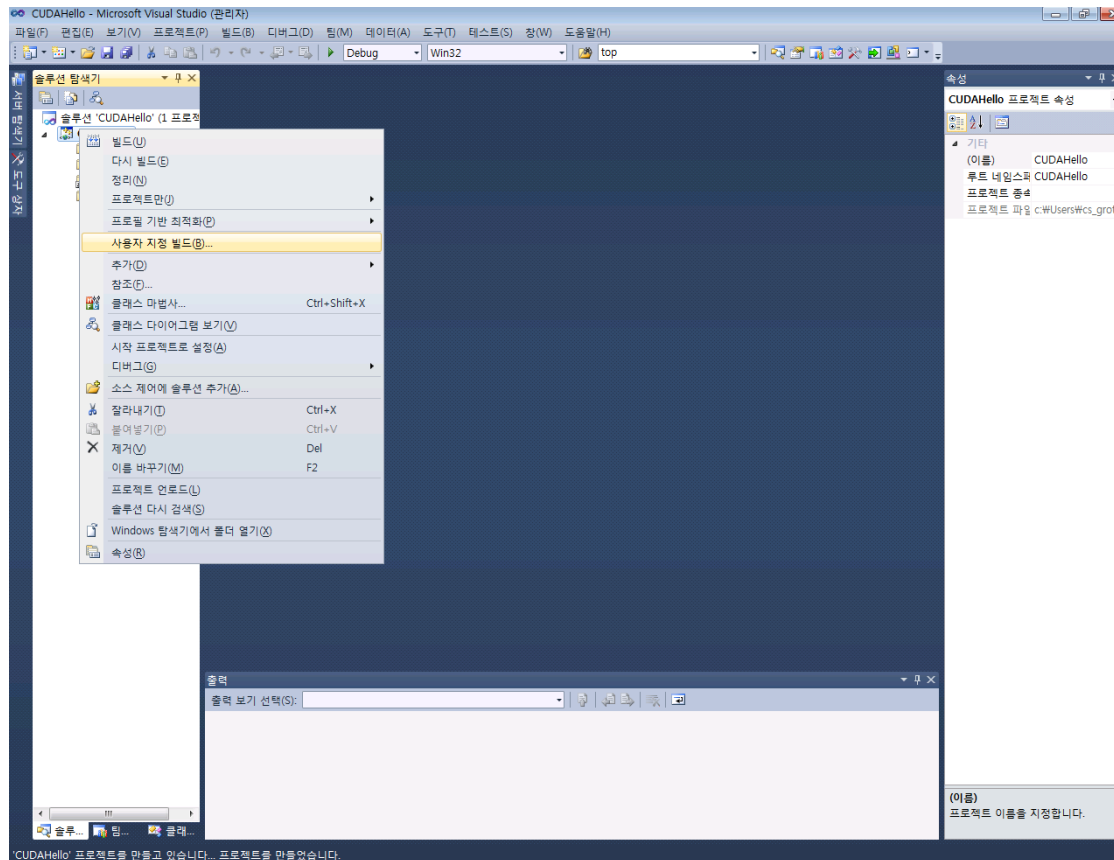
Setting at VS2010

■ VS2010에서 새 프로젝트 생성



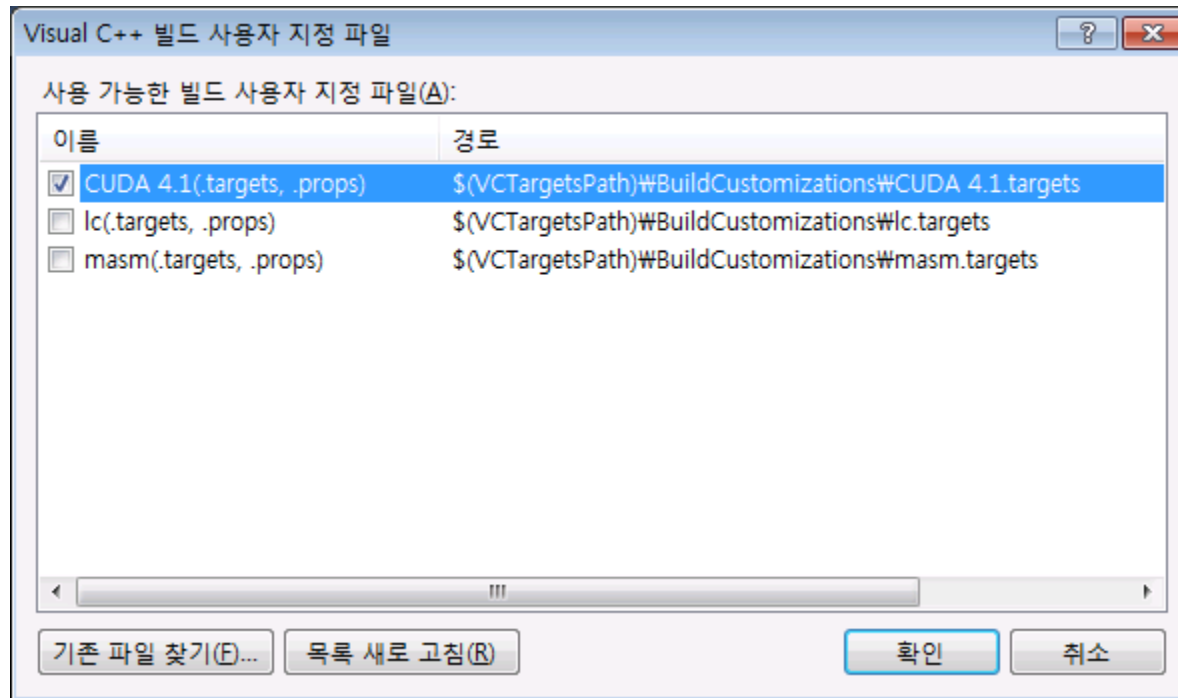
Setting at VS2010

- 솔루션 탐색기 -> 프로젝트 이름(우클릭) -> 사용자 지정 빌드



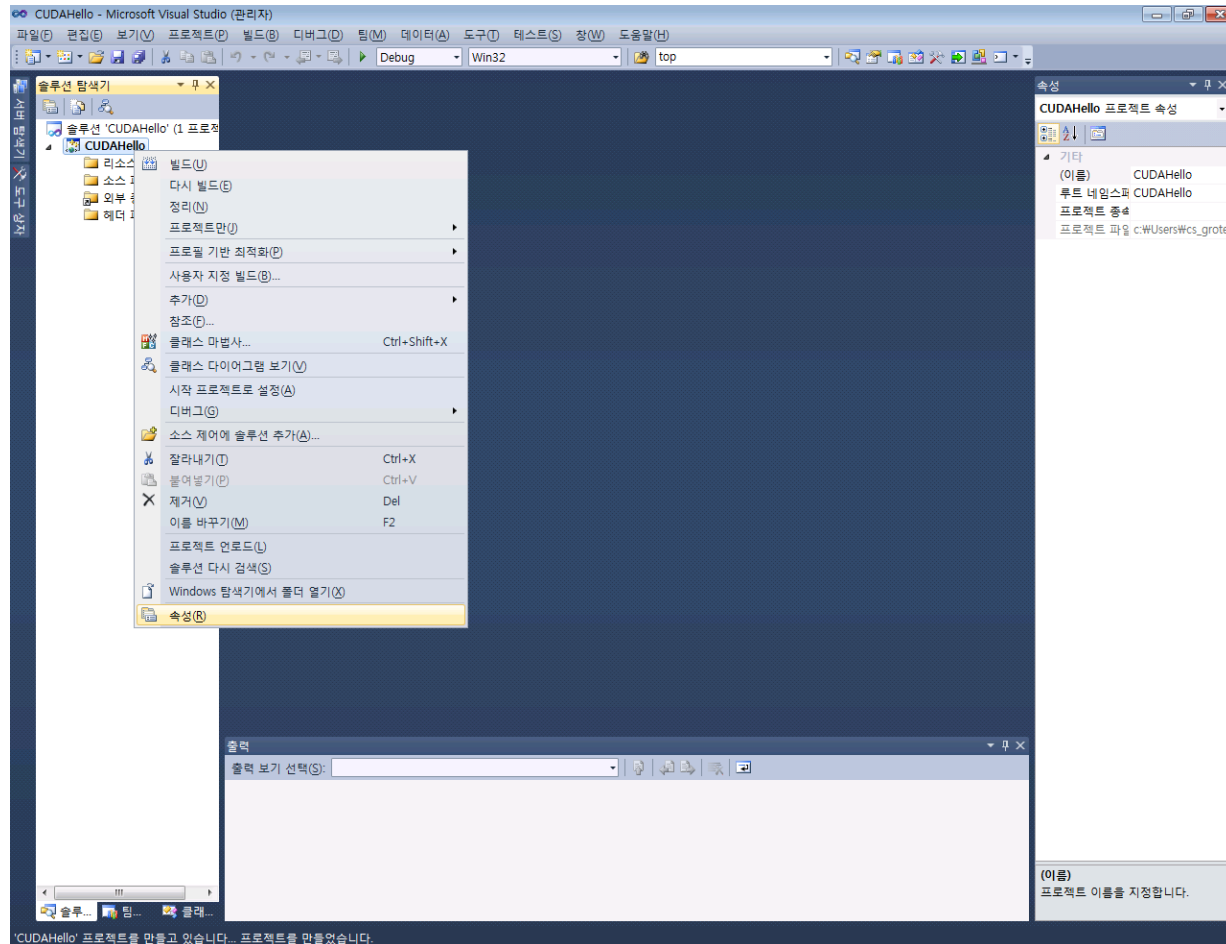
Setting at VS2010

■ CUDA 선택



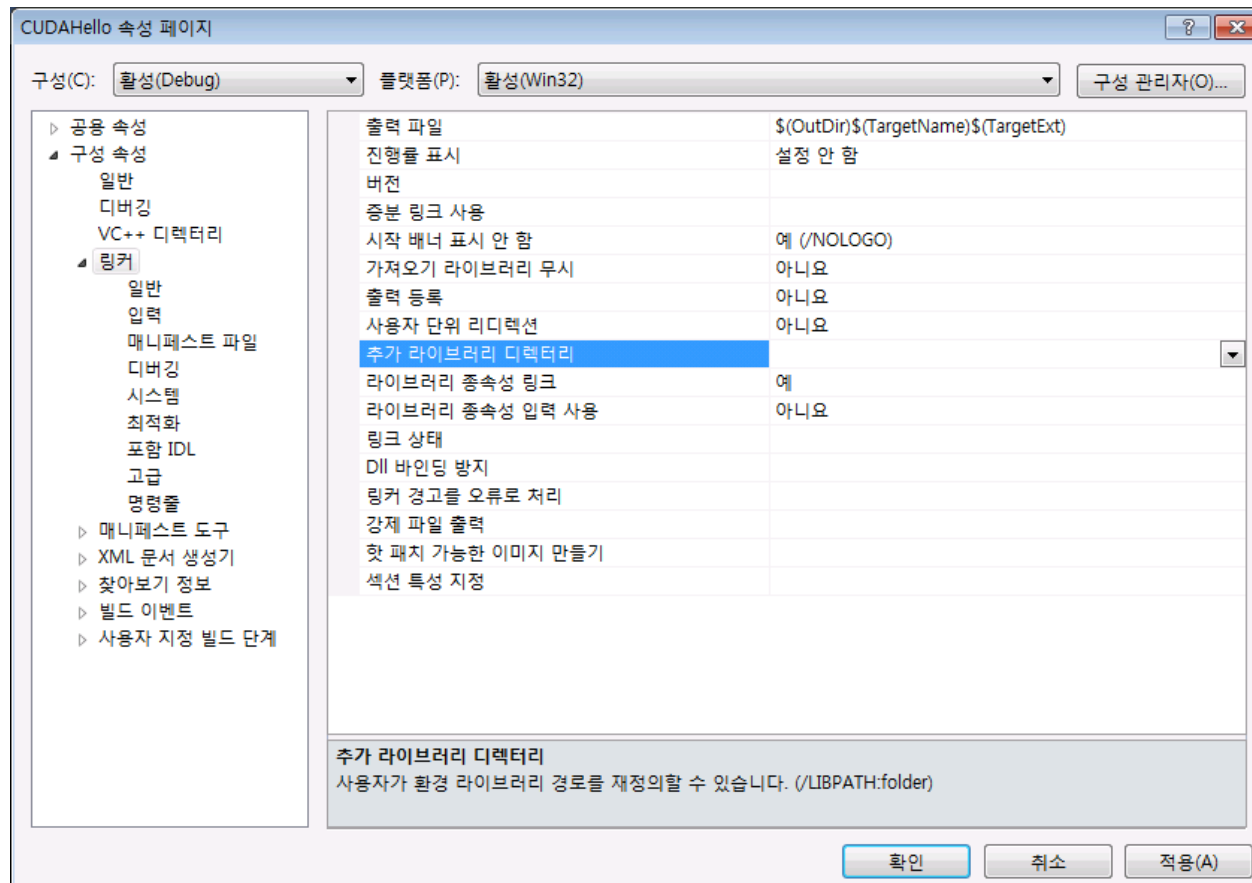
Setting at VS2010

■ 다시 프로젝트 이름(우클릭) -> 속성



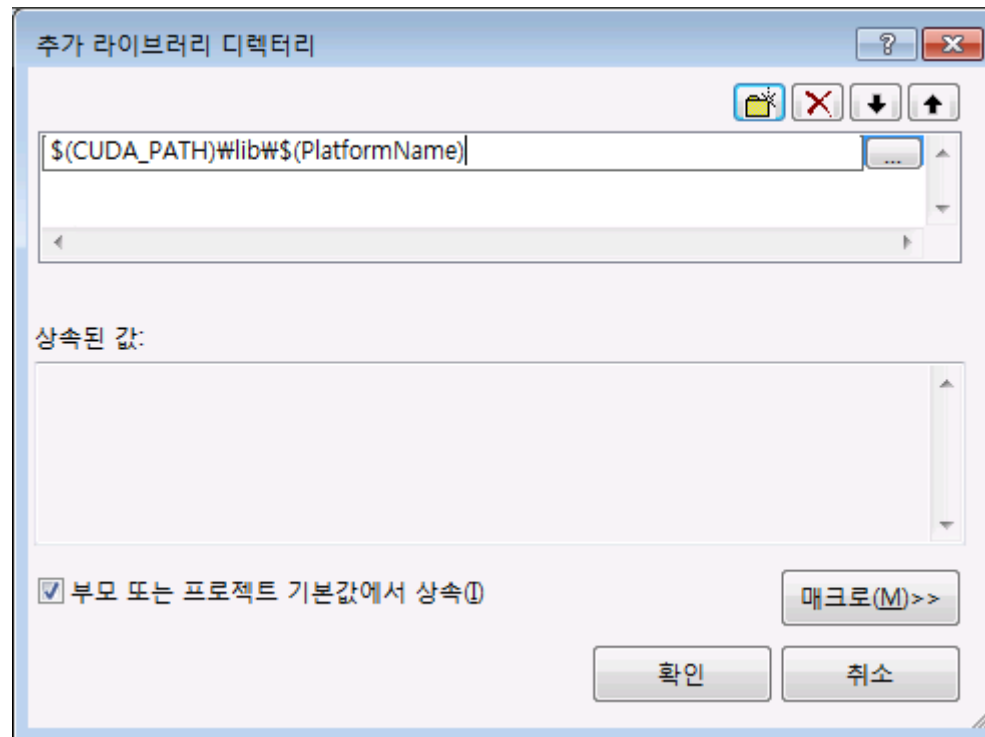
Setting at VS2010

- 구성 속성 -> 링커 -> 추가 라이브러리 디렉터리 -> 편집



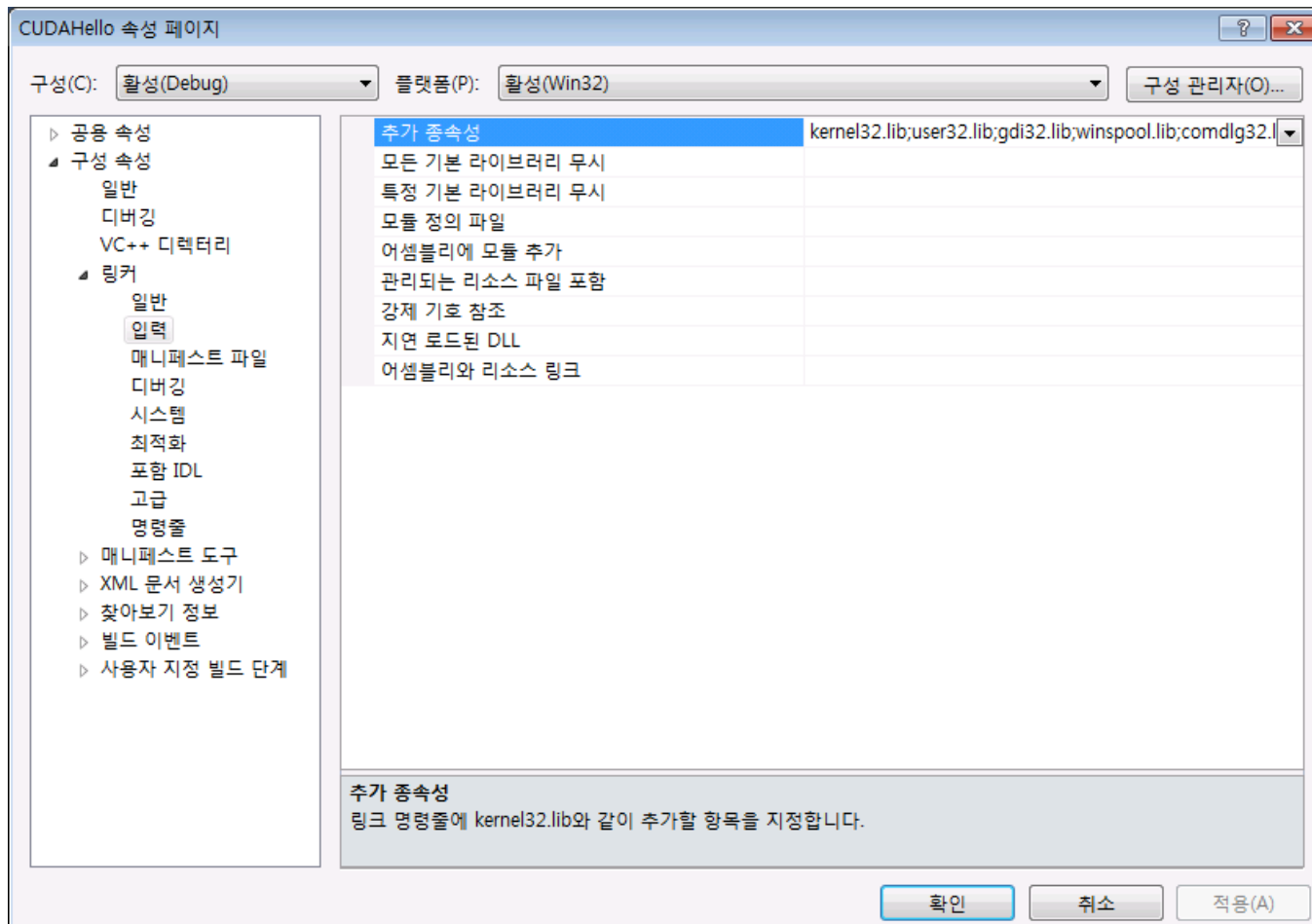
Setting at VS2010

- `$(CUDA_PATH)\lib\$(PlatformName)` 추가



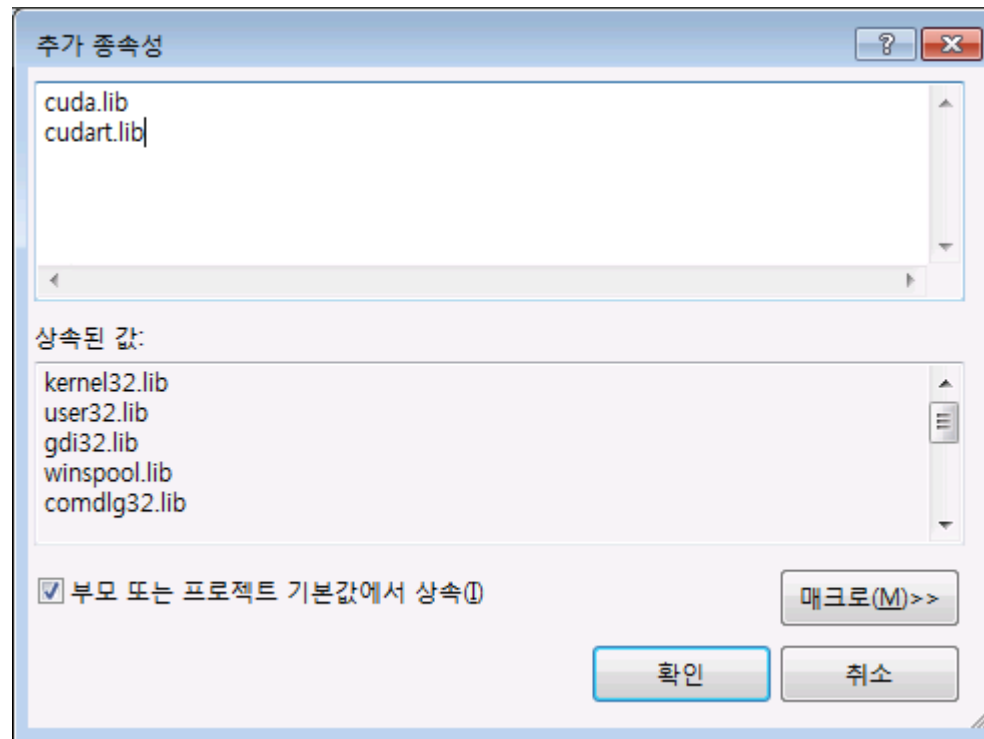
Setting at VS2010

■ 링커 -> 입력 -> 추가 종속성 -> 편집



Setting at VS2010

■ cuda.lib cudart.lib 등록

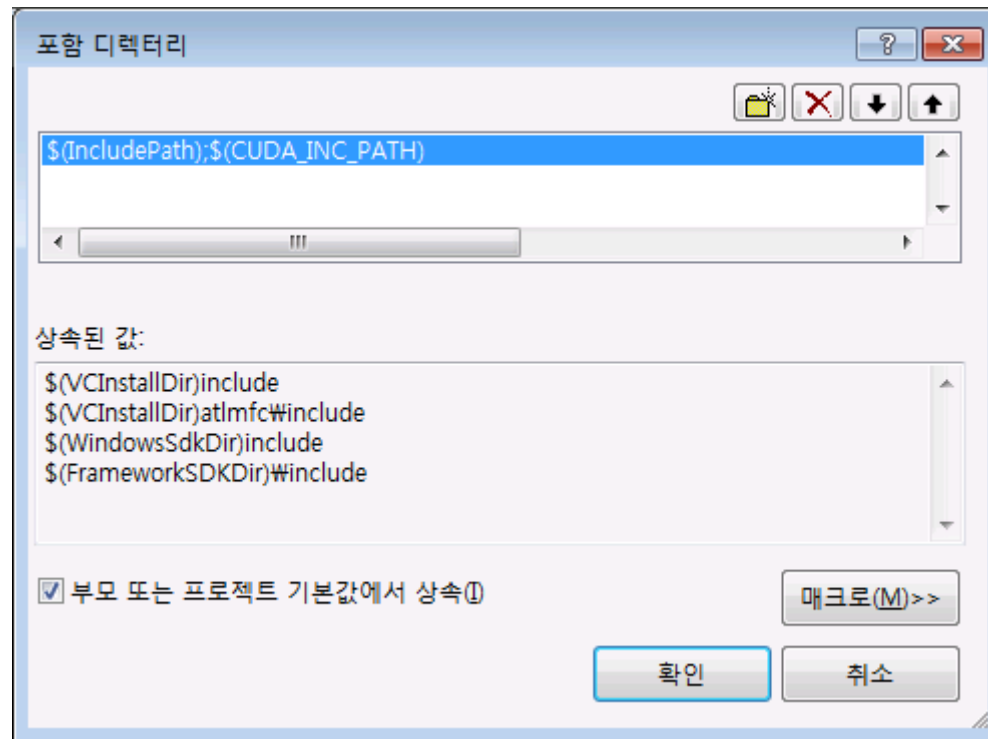


■ VC++ 디렉터리 -> 포함 디렉터리



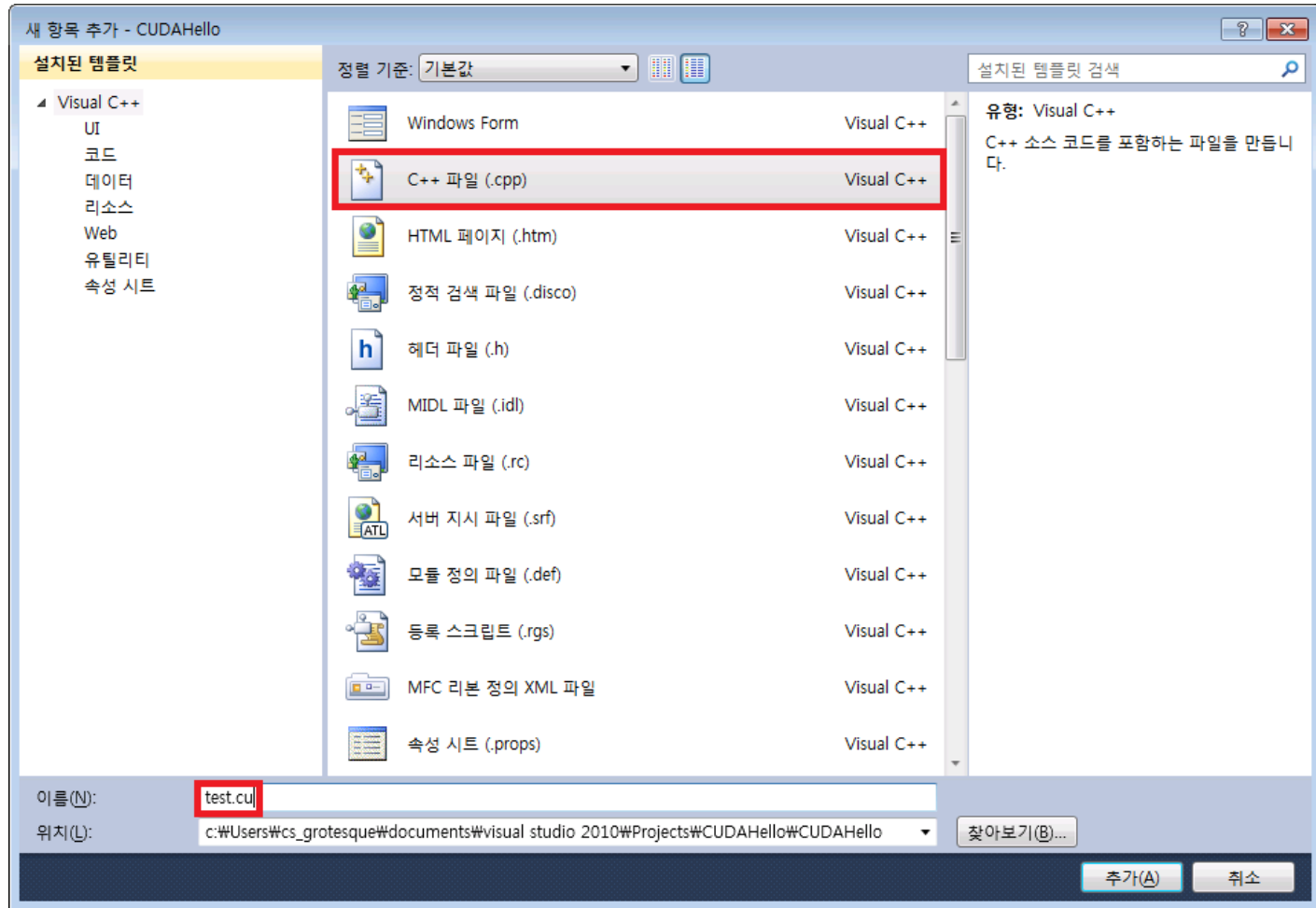
Setting at VS2010

■ \$(IncludePath);\$(CUDA_INC_PATH)



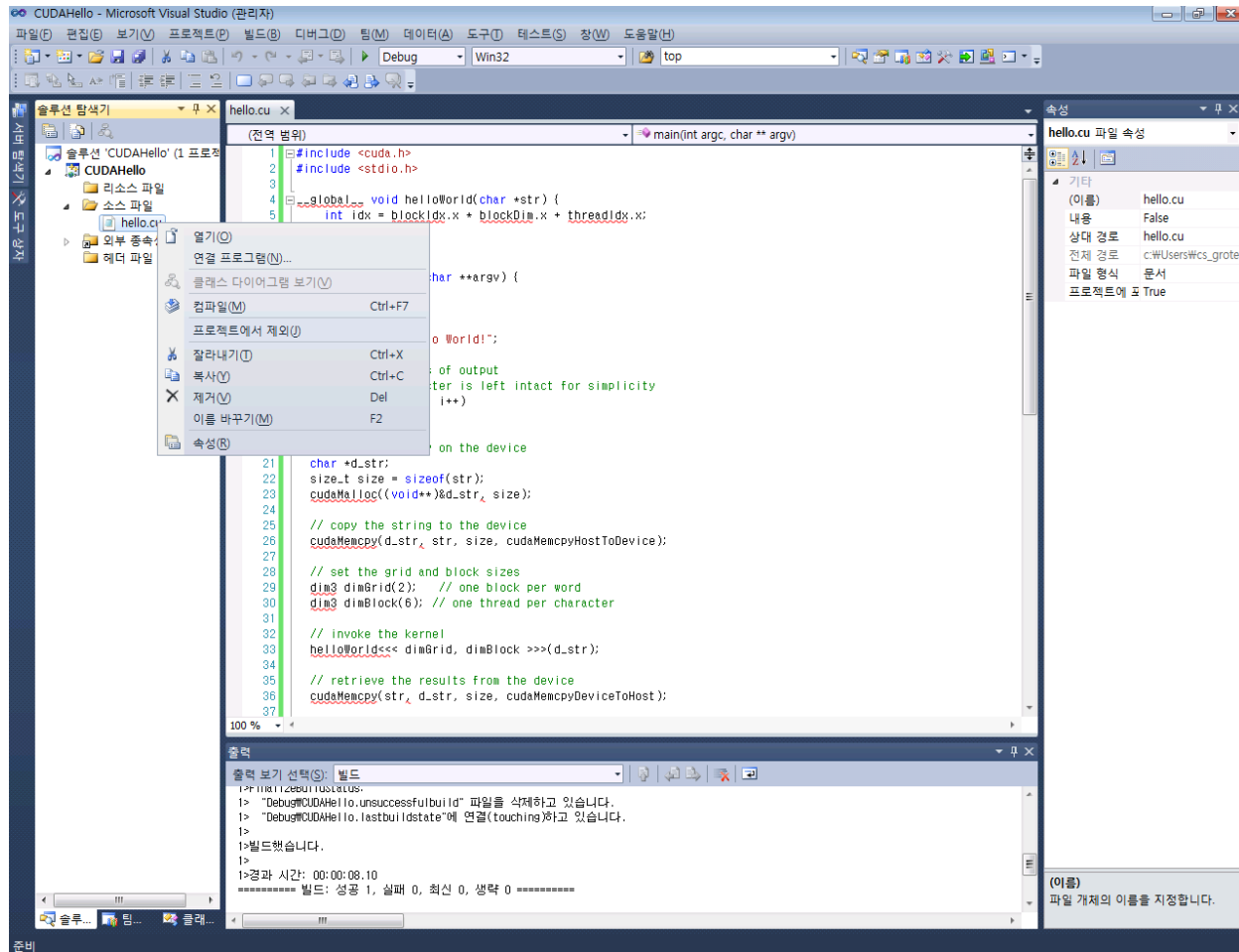
Setting at VS2010

■ 소스 코드 파일 추가



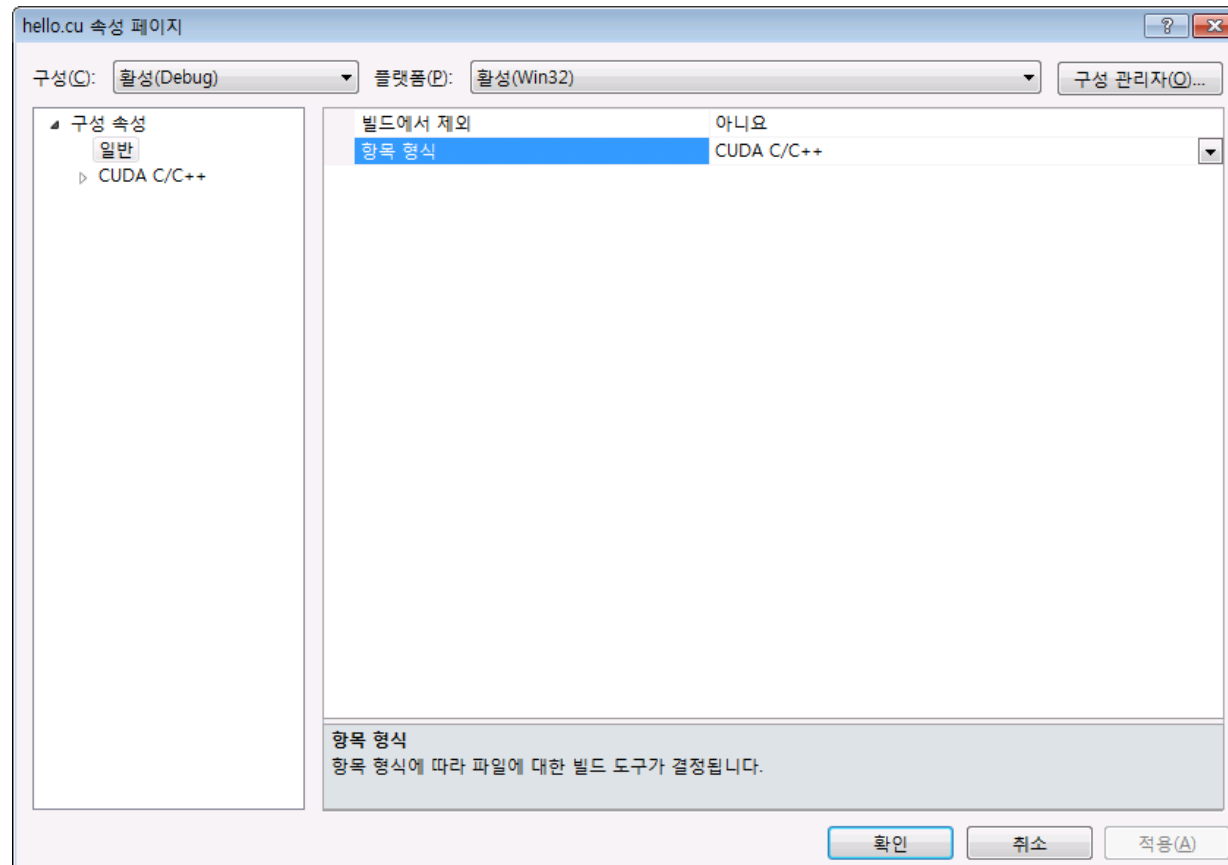
Setting at VS2010

■ 추가 후 소스 파일 우클릭 -> 속성

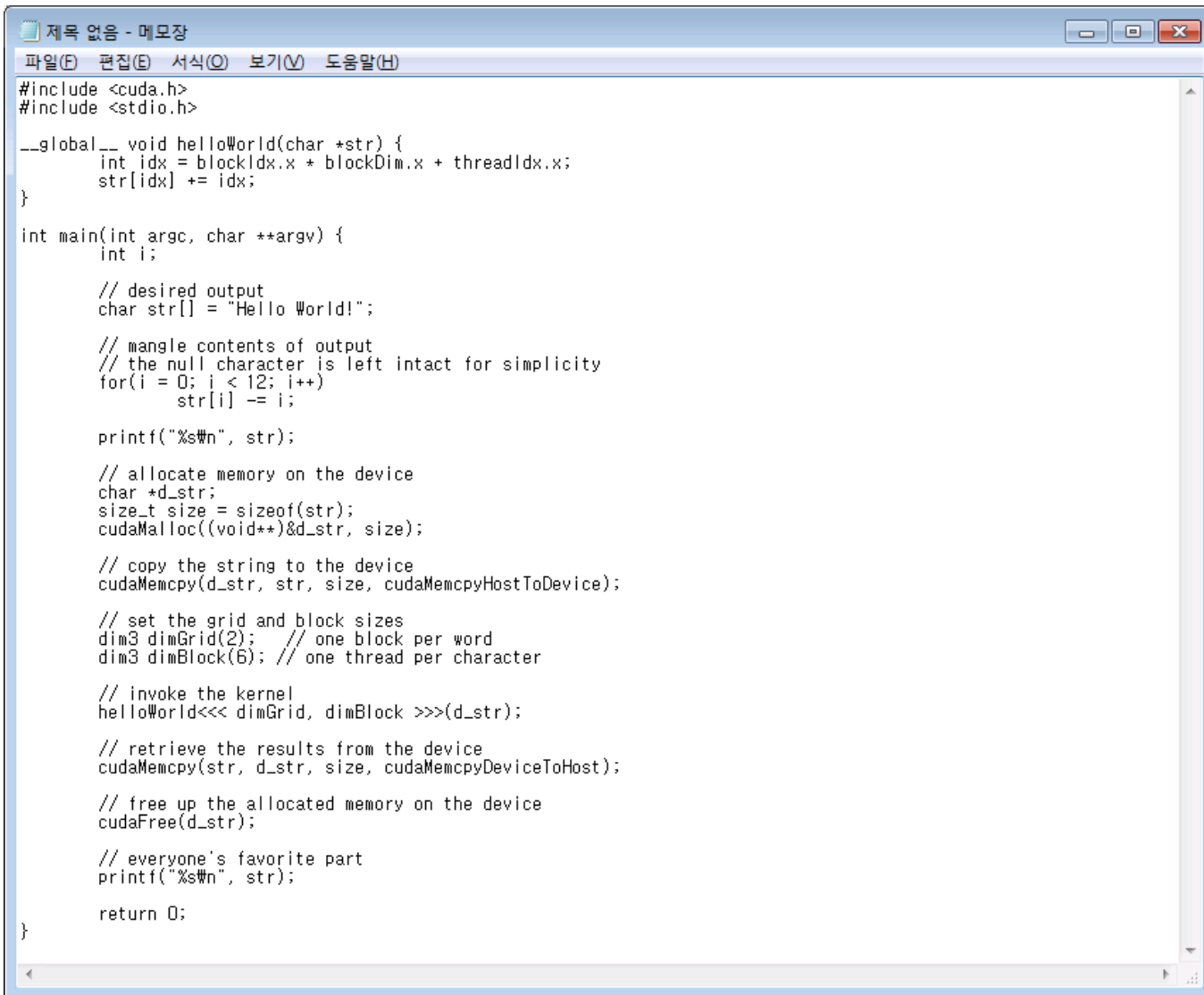


Setting at VS2010

■ 일반 -> 항목 형식 -> CUDA C/C++



실습 예제 코드



```
제목 없음 - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

#include <cuda.h>
#include <stdio.h>

__global__ void helloWorld(char *str) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    str[idx] += idx;
}

int main(int argc, char **argv) {
    int i;

    // desired output
    char str[] = "Hello World!";

    // mangle contents of output
    // the null character is left intact for simplicity
    for(i = 0; i < 12; i++)
        str[i] -= i;

    printf("%s\n", str);

    // allocate memory on the device
    char *d_str;
    size_t size = sizeof(str);
    cudaMalloc((void**)&d_str, size);

    // copy the string to the device
    cudaMemcpy(d_str, str, size, cudaMemcpyHostToDevice);

    // set the grid and block sizes
    dim3 dimGrid(2); // one block per word
    dim3 dimBlock(6); // one thread per character

    // invoke the kernel
    helloWorld<<< dimGrid, dimBlock >>>(d_str);

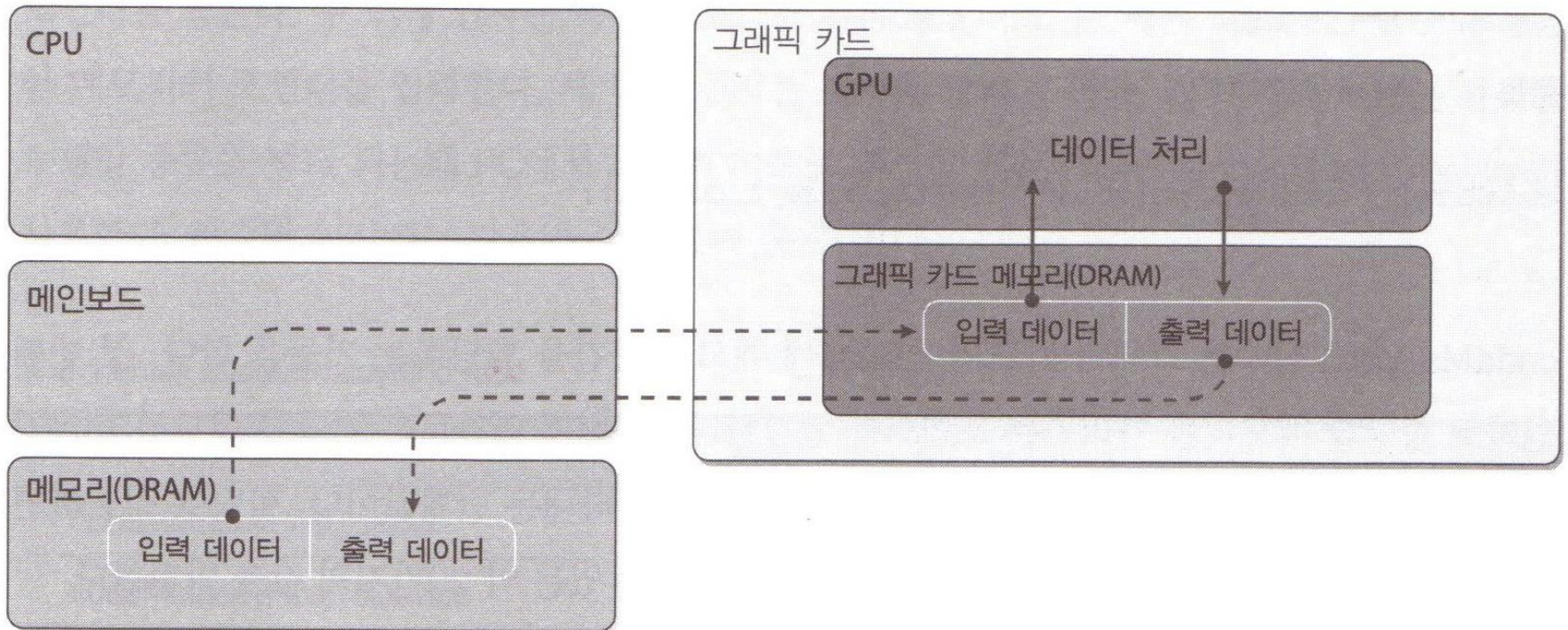
    // retrieve the results from the device
    cudaMemcpy(str, d_str, size, cudaMemcpyDeviceToHost);

    // free up the allocated memory on the device
    cudaFree(d_str);

    // everyone's favorite part
    printf("%s\n", str);

    return 0;
}
```

CUDA의 데이터 처리 과정





CUDA Memory

- Allocation
- `cudaError_t cudaMalloc(void **devPtr, size_t count);`
- 그래픽 카드의 DRAM에 메모리 공간 할당
- 할당할 메모리의 포인터 : `devPtr`
- 메모리의 크기 : `count`



CUDA Memory

■ Example

```
char *d_str;  
size_t size = sizeof(str);  
cudaMalloc((void**)&d_str, size);
```




CUDA Memory

- Deallocation

- `cudaError_t cudaFree(void *devPtr);`

- 그래픽 카드의 DRAM에 할당된 메모리 해제

- 해제하고자 하는 포인터 : `devPtr`

- Example

```
cudaFree(d_str);
```



CUDA Memory

- Data Copy
- `cudaError_t cudaMemcpy(void *dst, const void* src, size_t count, enum cudaMemcpyKind kind);`
- 메모리에서 데이터 복사
- `src`에서 `dst`로 `count`만큼 복사

CUDA Memory

■ Enum cudaMemcpyKind kind

종류	동작
cudaMemcpyHostToHost	PC 메모리에서 PC 메모리로
cudaMemcpyHostToDevice	PC 메모리에서 그래픽 카드로
cudaMemcpyDeviceToHost	그래픽 카드 메모리에서 PC 메모리로
cudaMemcpyDeviceToDevice	그래픽 카드 메모리에서 그래픽 카드 메모리로

■ Example

```
cudaMemcpy(d_str, str, size, cudaMemcpyHostToDevice);  
cudaMemcpy(str, d_str, size, cudaMemcpyDeviceToHost);
```



Kernel Function

- `__global__ void kernelFunc(type param, ...)`
- 선언 및 정의 시에는 위의 코드 사용
- `main`에서 수행 시에는 아래의 코드 사용
- `kernelFunc<<<Block, Thread>>>(type param, ...);`

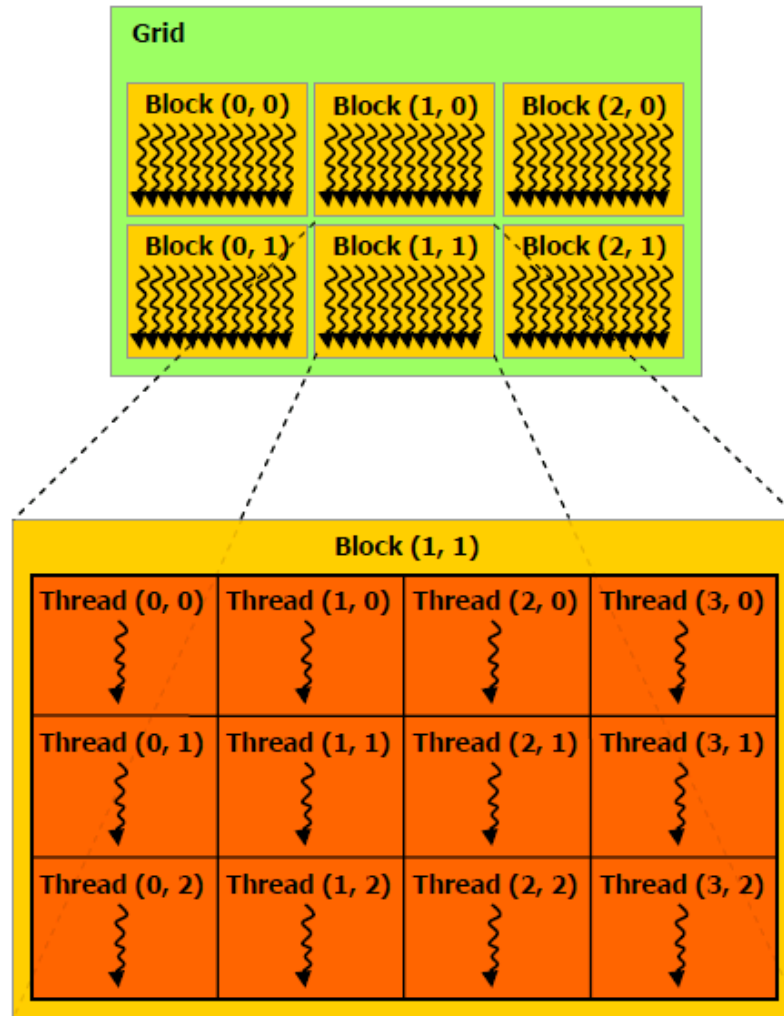


Kernel Function

■ Example

```
__global__ void  
helloWorld(char* str)  
{  
    // determine where in the thread grid we are  
    int idx = blockIdx.x * blockDim.x + threadIdx.x;  
  
    // unmangle output  
    str[idx] += idx;  
}  
  
helloWorld<<< dimGrid, dimBlock >>>(d_str);
```

Grid, Block, Thread



Parallel Processing



Grid, Block, Thread와 Kernel Function

- Kernel Function 수행 시 <<<a, b>>>는 a개의 block에서 block당 b개의 thread 생성
- Block과 Thread는 다차원으로 생성 가능
 - dim3 변수를 이용

```
dim3 grid, block;  
grid.x = 2; grid.y = 4;  
block.x = 8; block.y = 16;  
kernel<<<grid, block>>>(...);
```

```
dim3 grid(2, 4), block(8, 16);  
kernel<<<grid, block>>>(...);
```

← Equivalent assignment using
constructor functions

```
kernel<<<32, 512>>>(...);
```



Grid, Block, Thread와 Kernel Function

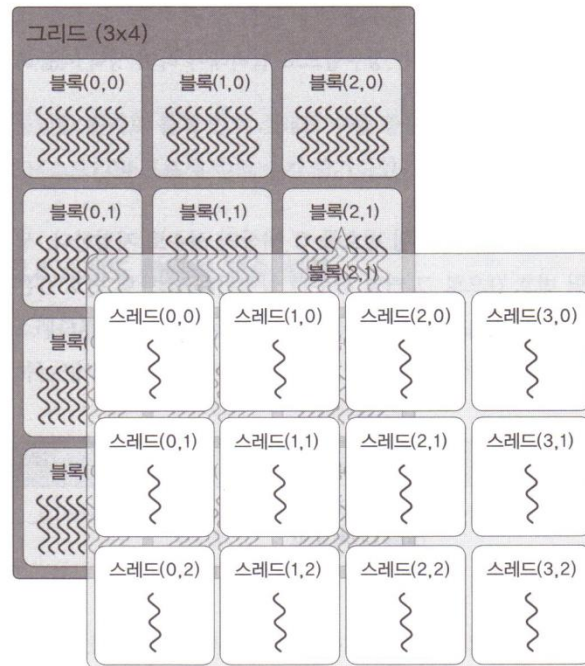
```
dim3 dimGrid(2);  
dim3 dimBlock(6);  
  
helloWorld<<< dimGrid, dimBlock >>>(d_str);  
  
__global__ void  
helloWorld(char* str)  
{  
    // determine where in the thread grid we are  
    int idx = blockIdx.x * blockDim.x + threadIdx.x;  
  
    // unmangle output  
    str[idx] += idx;  
}
```

- idx?
- blockIdx : block의 Index
- blockDim : 몇 번째의 block
- threadIdx : thread의 인덱스



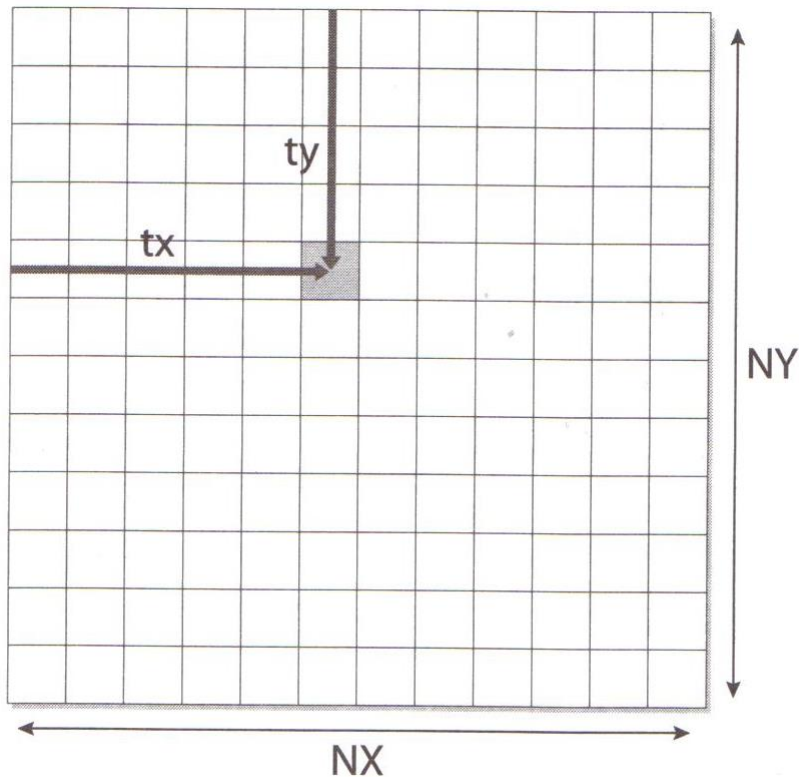
2차원 Kernel Index

- dim3 Dg(3, 4, 1);
- dim3 Db(4, 3, 1);
- Kernel<<<Dg, Db>>>(a, b, c);

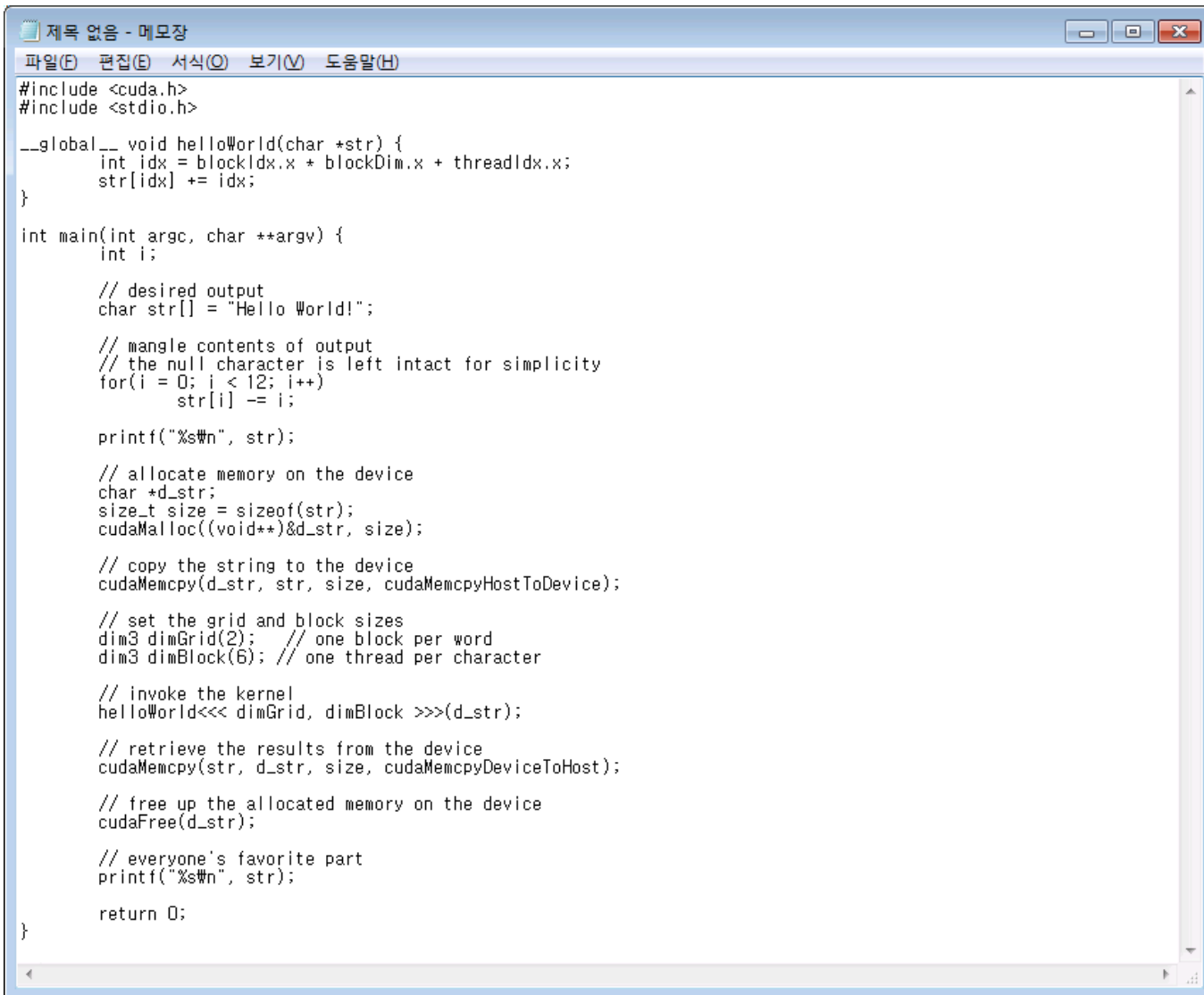


2차원 Kernel Index

```
int tid, tx, ty;  
tx = blockDim.x * blockIdx.x + threadIdx.x;  
ty = blockDim.y * blockIdx.y + threadIdx.y;  
tid = NX * ty + tx;
```



실습 예제 코드



```
제목 없음 - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

#include <cuda.h>
#include <stdio.h>

__global__ void helloWorld(char *str) {
    int idx = blockIdx.x * blockDim.x + threadIdx.x;
    str[idx] += idx;
}

int main(int argc, char **argv) {
    int i;

    // desired output
    char str[] = "Hello World!";

    // mangle contents of output
    // the null character is left intact for simplicity
    for(i = 0; i < 12; i++)
        str[i] -= i;

    printf("%s\n", str);

    // allocate memory on the device
    char *d_str;
    size_t size = sizeof(str);
    cudaMalloc((void**)&d_str, size);

    // copy the string to the device
    cudaMemcpy(d_str, str, size, cudaMemcpyHostToDevice);

    // set the grid and block sizes
    dim3 dimGrid(2); // one block per word
    dim3 dimBlock(6); // one thread per character

    // invoke the kernel
    helloWorld<<< dimGrid, dimBlock >>>(d_str);

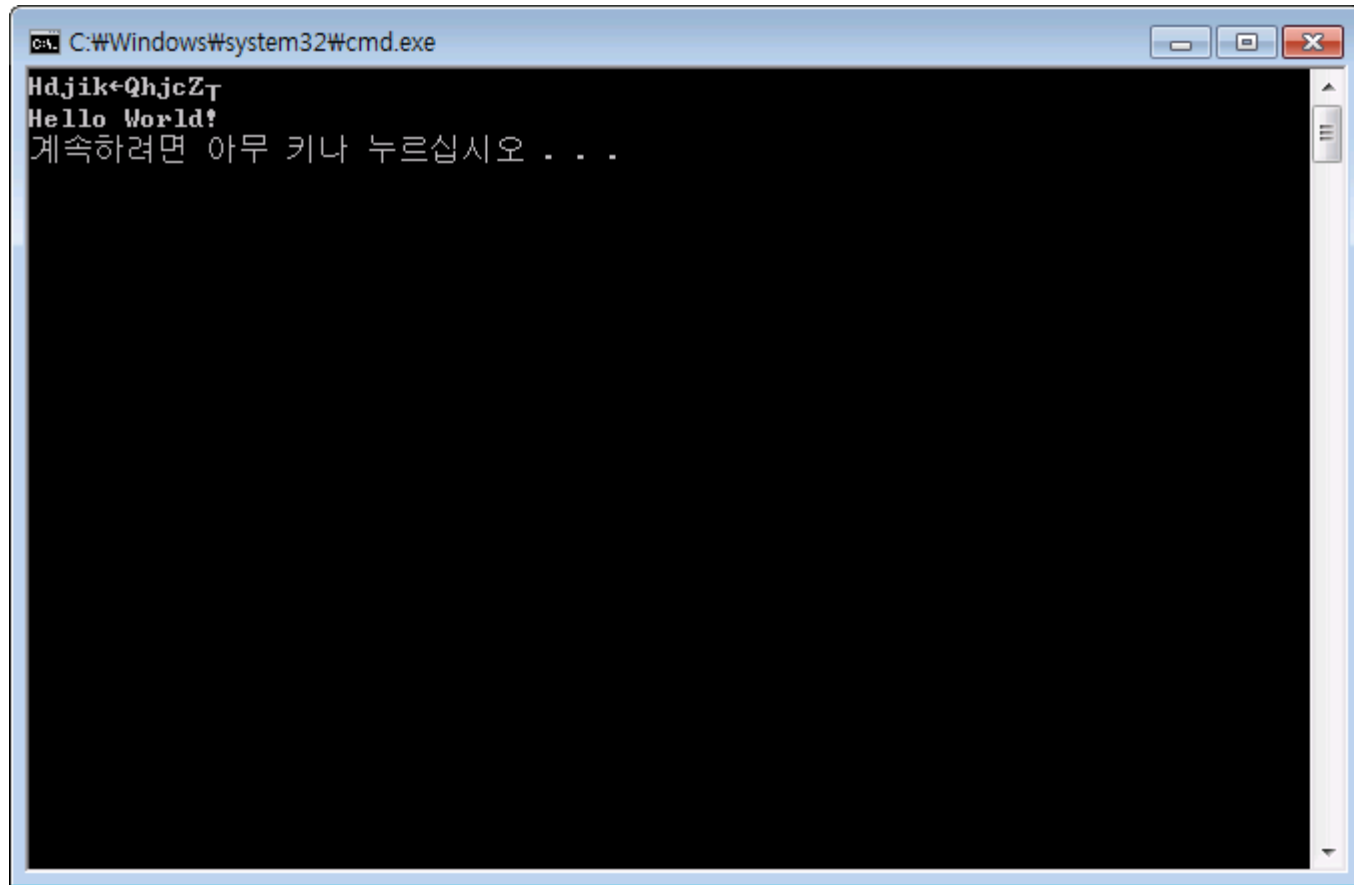
    // retrieve the results from the device
    cudaMemcpy(str, d_str, size, cudaMemcpyDeviceToHost);

    // free up the allocated memory on the device
    cudaFree(d_str);

    // everyone's favorite part
    printf("%s\n", str);

    return 0;
}
```

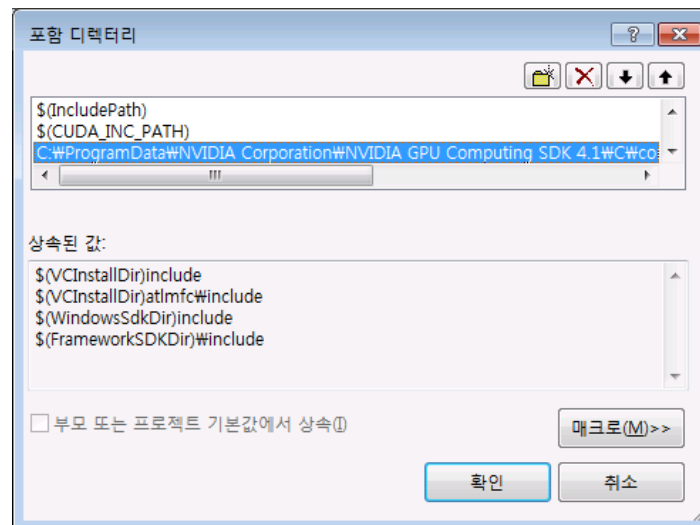
실행 결과



A screenshot of a Windows command prompt window. The title bar shows the path "C:\Windows\system32\cmd.exe". The command prompt displays the following text: "Hdjik←QhjcZ_T", "Hello World!", and "계속하려면 아무 키나 누르십시오 . . .".

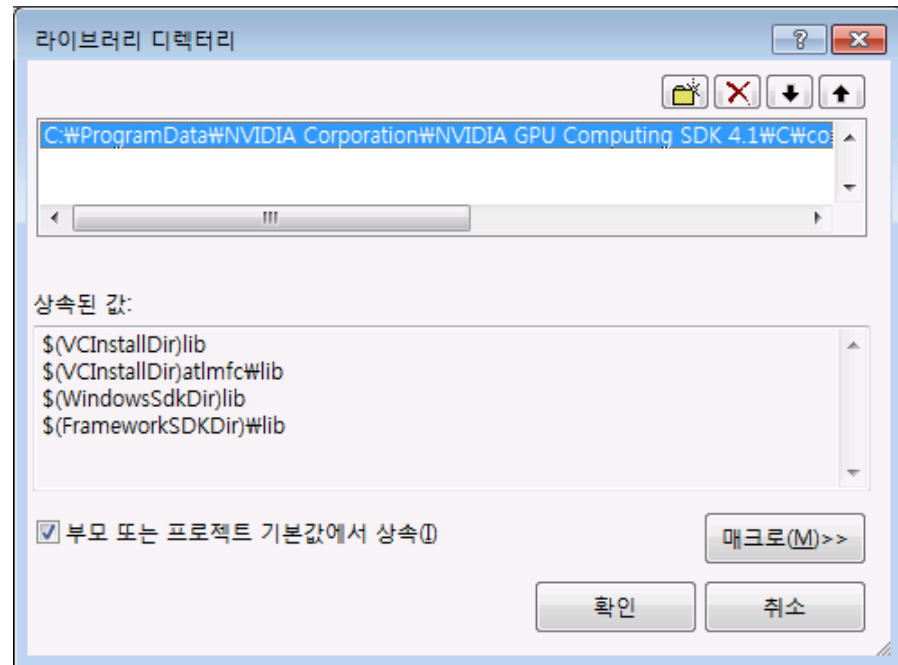
cutil.h 사용

- 프로젝트 속성 -> 링커 -> VC++ 디렉터리 -> 포함 디렉터리
- C:\ProgramData\NVIDIA Corporation\NVIDIA GPU Computing SDK 4.2\C\common\inc 등록



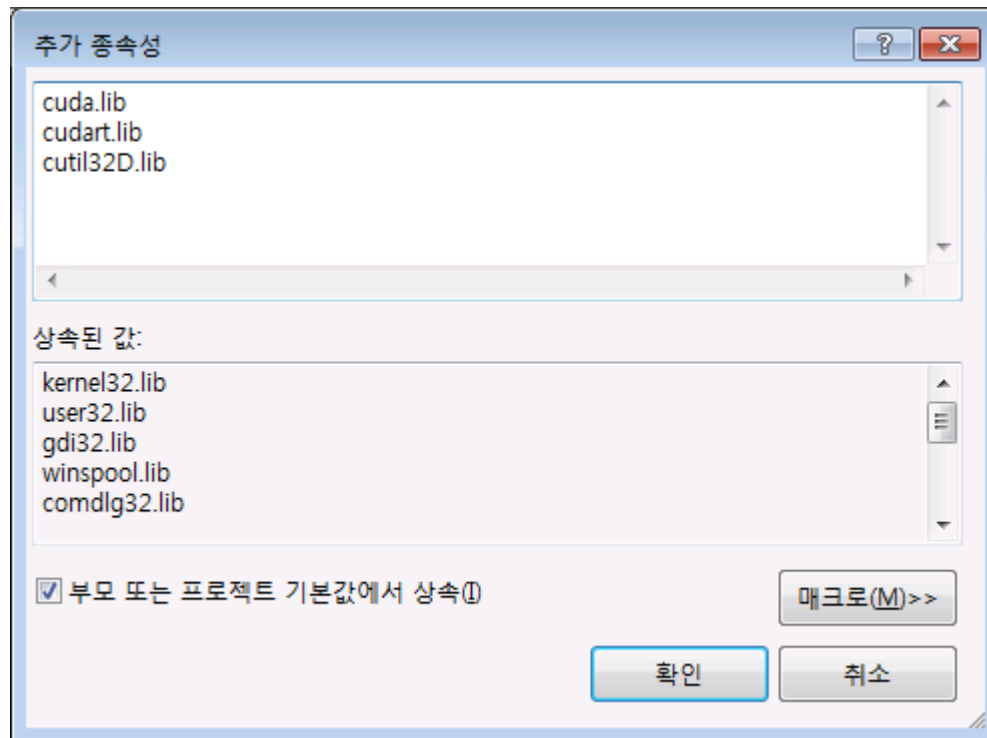
cutil.h 사용

- 라이브러리 디렉터리에다가도 추가
- C:\ProgramData\NVIDIA Corporation\NVIDIA GPU Computing SDK 4.2\C\common\lib\Win32(64비트 일 경우 x64)



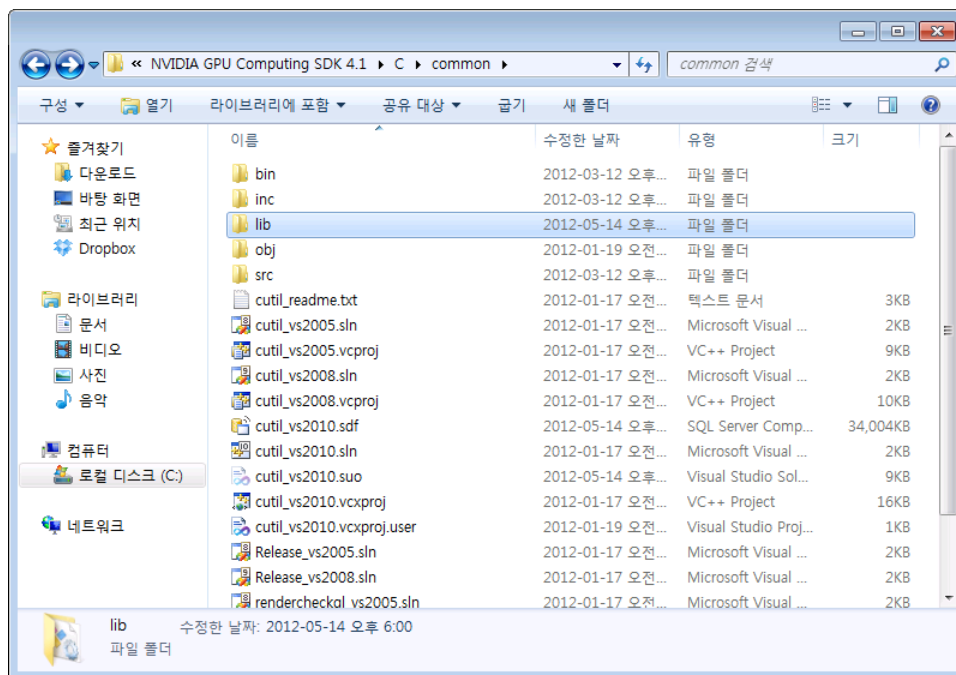
cutil.h 사용

- 링커 -> 입력 -> 추가 종속성 -> cutil32D.lib 등록(64비트일 경우 cutil64D.lib)



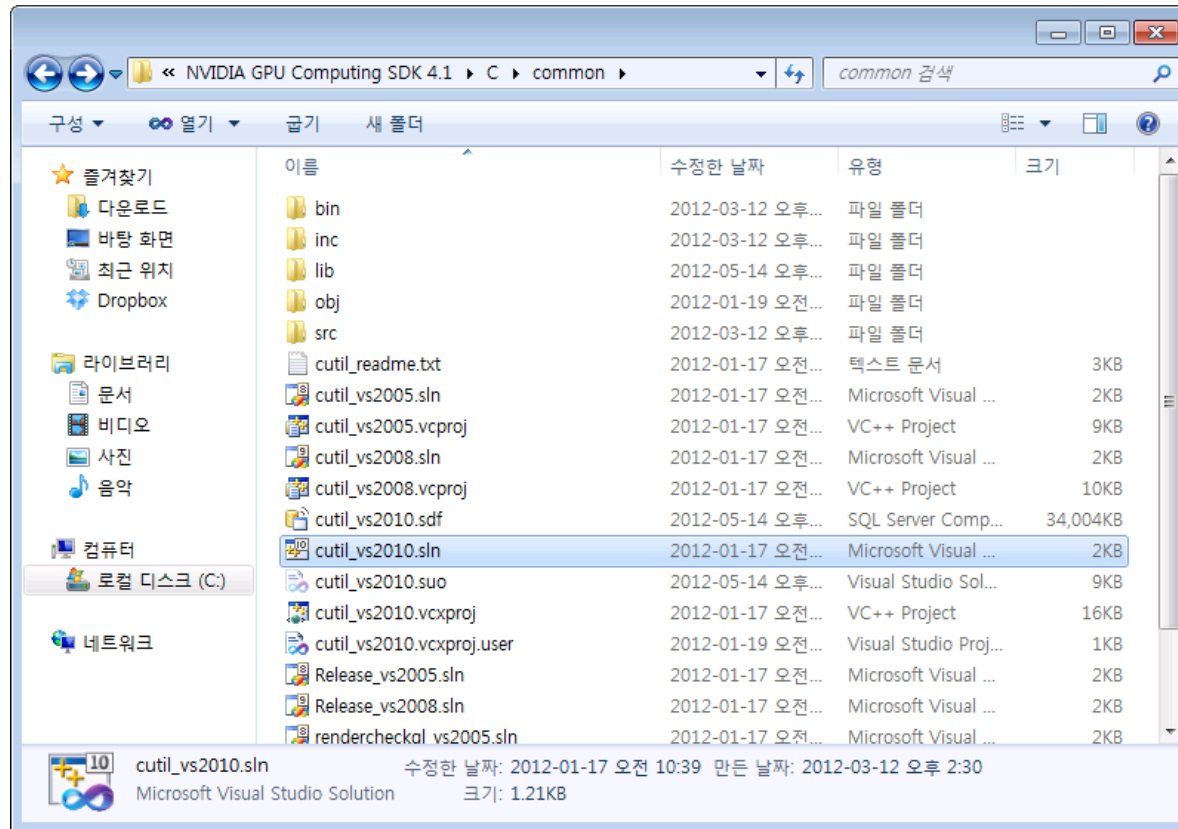
cutil.h 사용

- 탐색기를 열어 주소 창에
C:\ProgramData\NVIDIA
Corporation\NVIDIA GPU Computing SDK
4.2\C\common 입력



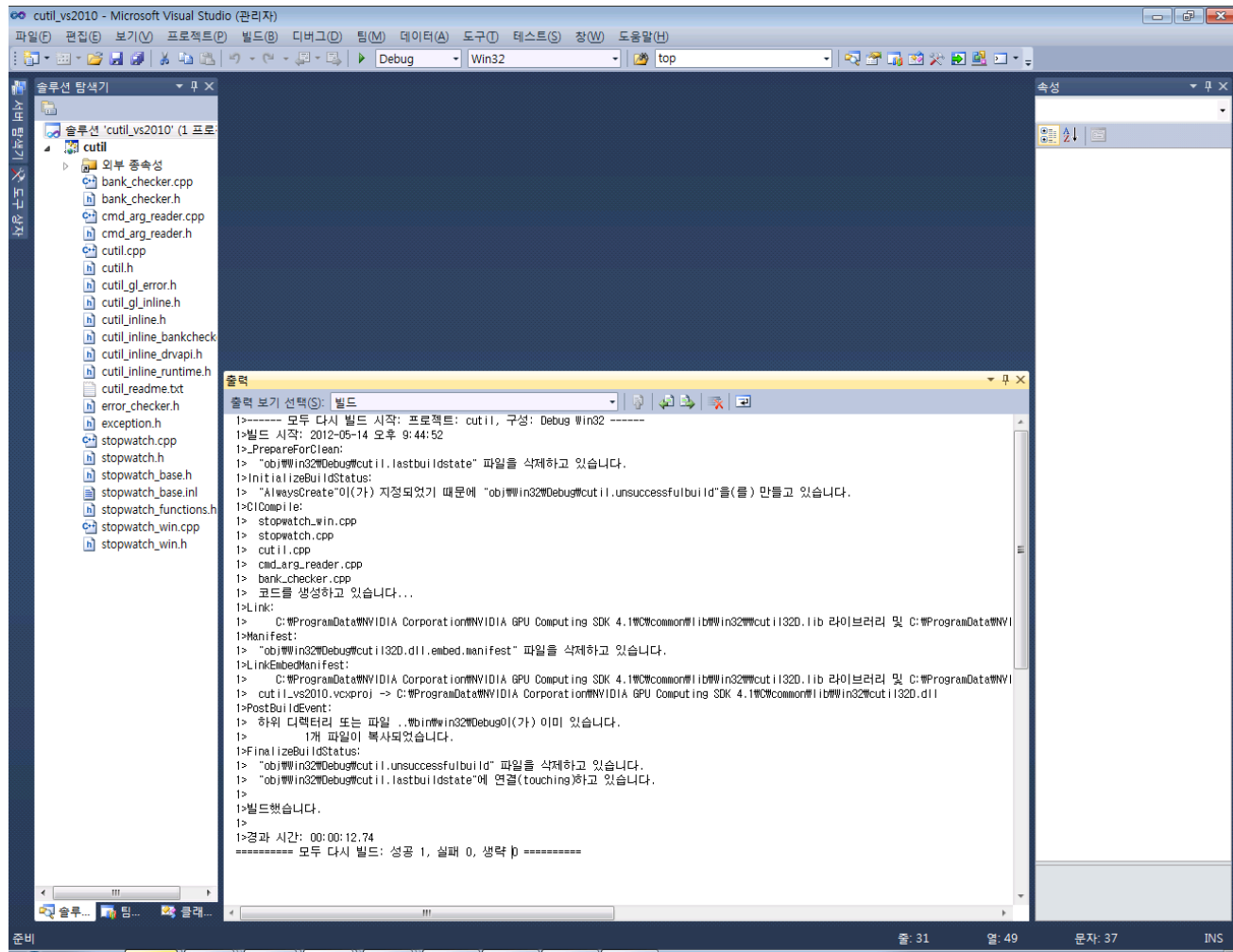
cutil.h 사용

■ Cutil_vs2010.sln 열기



cutil.h 사용

프로젝트 빌드

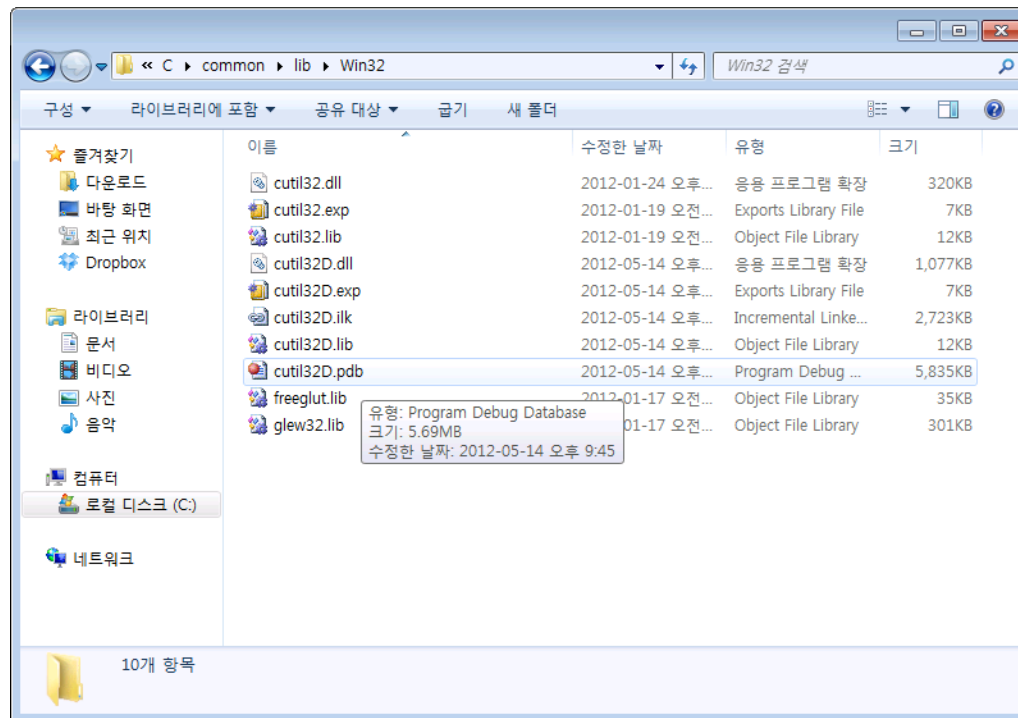


The screenshot shows the Microsoft Visual Studio IDE with the 'cutil' project selected in the Solution Explorer. The 'Output' window displays the build log for the 'Debug Win32' configuration. The log shows the successful compilation and linking of the project, resulting in the 'cutil32.dll' file.

```
>----- 모두 다시 빌드 시작: 프로젝트: cutil, 구성: Debug Win32 -----
>빌드 시작: 2012-05-14 오후 9:44:52
>_PrepareForClean:
> "obj\Win32\Debug\cutil.lastbuildstate" 파일을 삭제하고 있습니다.
>InitializeBuildStatus:
> "AlwaysCreate"이(가) 지정되었기 때문에 "obj\Win32\Debug\cutil.unsuccessfulbuild"을(를) 만들고 있습니다.
>ClCompile:
> stopwatch_win.cpp
> stopwatch.cpp
> cutil.cpp
> cmd_arg_reader.cpp
> bank_checker.cpp
> 코드를 생성하고 있습니다...
>Link:
> C:\ProgramData\NVIDIA Corporation\NVIDIA GPU Computing SDK 4.1\WC\common\lib\Win32\cutil32.lib 라이브러리 및 C:\ProgramData\NVIDIA Corporation\NVIDIA GPU Computing SDK 4.1\WC\common\lib\Win32\cutil32.dll
>Manifest:
> "obj\Win32\Debug\cutil32.dll.embed.manifest" 파일을 삭제하고 있습니다.
>LinkEmbedManifest:
> C:\ProgramData\NVIDIA Corporation\NVIDIA GPU Computing SDK 4.1\WC\common\lib\Win32\cutil32.lib 라이브러리 및 C:\ProgramData\NVIDIA Corporation\NVIDIA GPU Computing SDK 4.1\WC\common\lib\Win32\cutil32.dll
>PostBuildEvent:
> 하위 디렉터리 또는 파일 ..\bin\Win32\Debug이(가) 이미 있습니다.
> 1개 파일이 복사되었습니다.
>FinalizeBuildStatus:
> "obj\Win32\Debug\cutil.unsuccessfulbuild" 파일을 삭제하고 있습니다.
> "obj\Win32\Debug\cutil.lastbuildstate"에 연결(touching)하고 있습니다.
>
>빌드했습니다.
>
>경과 시간: 00:00:12.74
>----- 모두 다시 빌드: 성공 1, 실패 0, 생략 0 -----
```

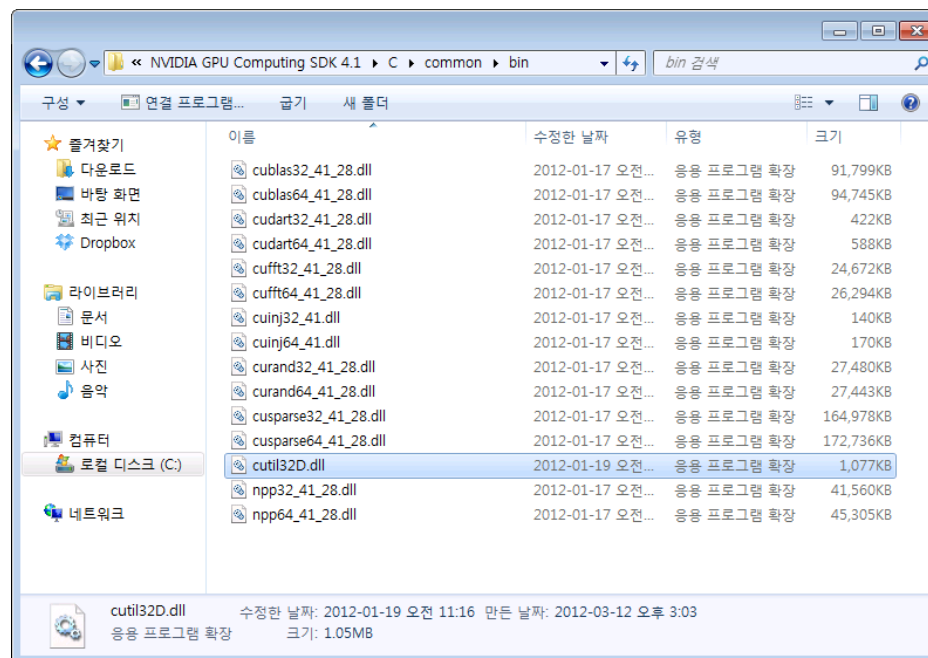
cutil.h 사용

- 빌드가 완료되면 솔루션 폴더의 lib\win32(64비트 경우 x64)에 가면 여러가지 파일 생성됨



cutil.h 사용

- 여기서 cutil32D.dll을
C:\ProgramData\NVIDIA
Corporation\NVIDIA GPU Computing SDK
4.2\C\common\bin에 복사





cutil.h 사용

- cutil.h에서 사용할 수 있는 함수들 중 timer 함수 사용
- cutCreateTimer(unsigned int* name);
 - 새로운 Timer 생성
- cutDeleteTimer(unsigned int name);
 - Timer 삭제

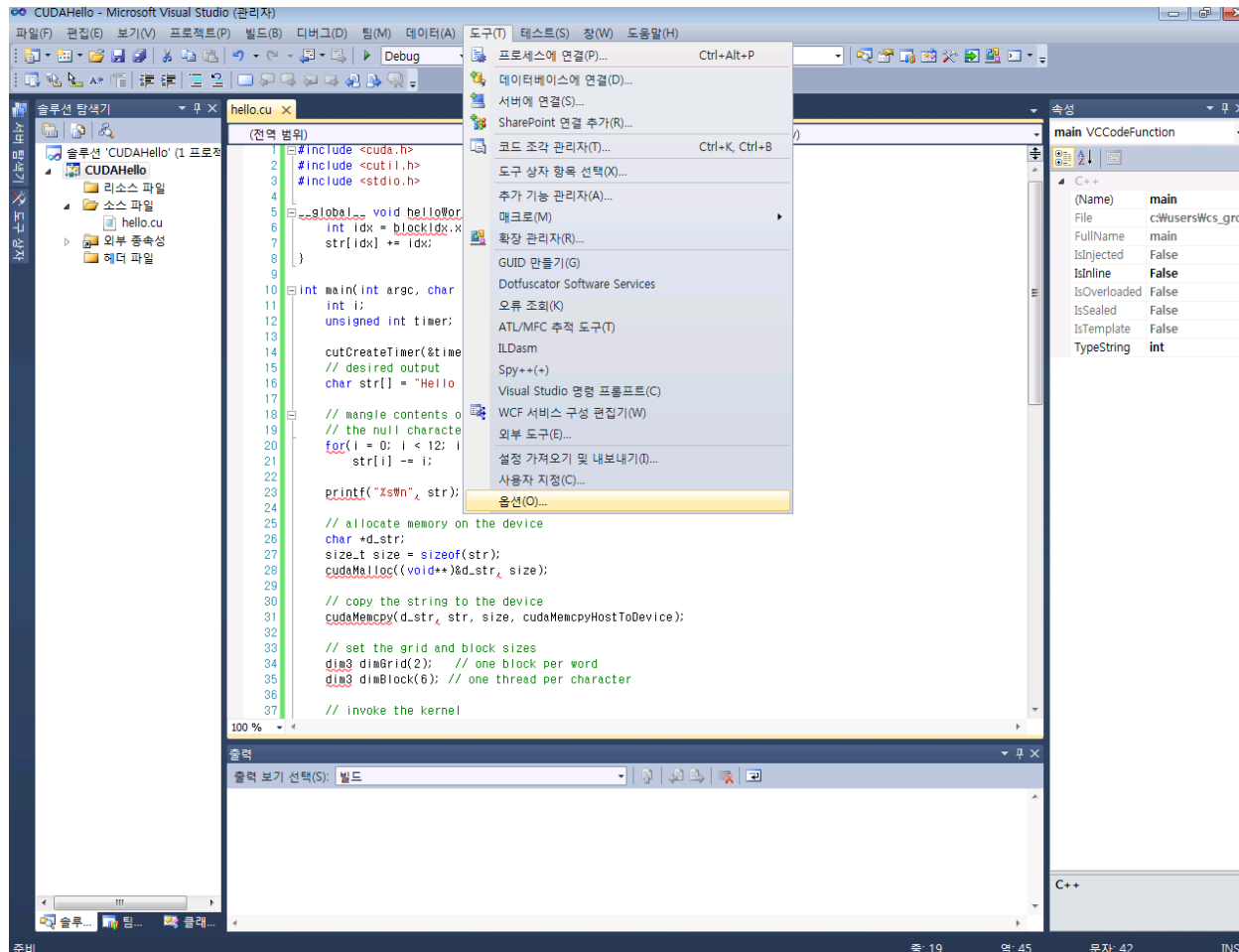


cutil.h 사용

- `cutStartTimer(const unsigned int name);`
 - Timer 시작
- `cutStopTimer(const unsigned int name);`
 - Timer 종료
- `Float cutGetTimerValue(const unsigned int name);`
 - Timer의 값 반환

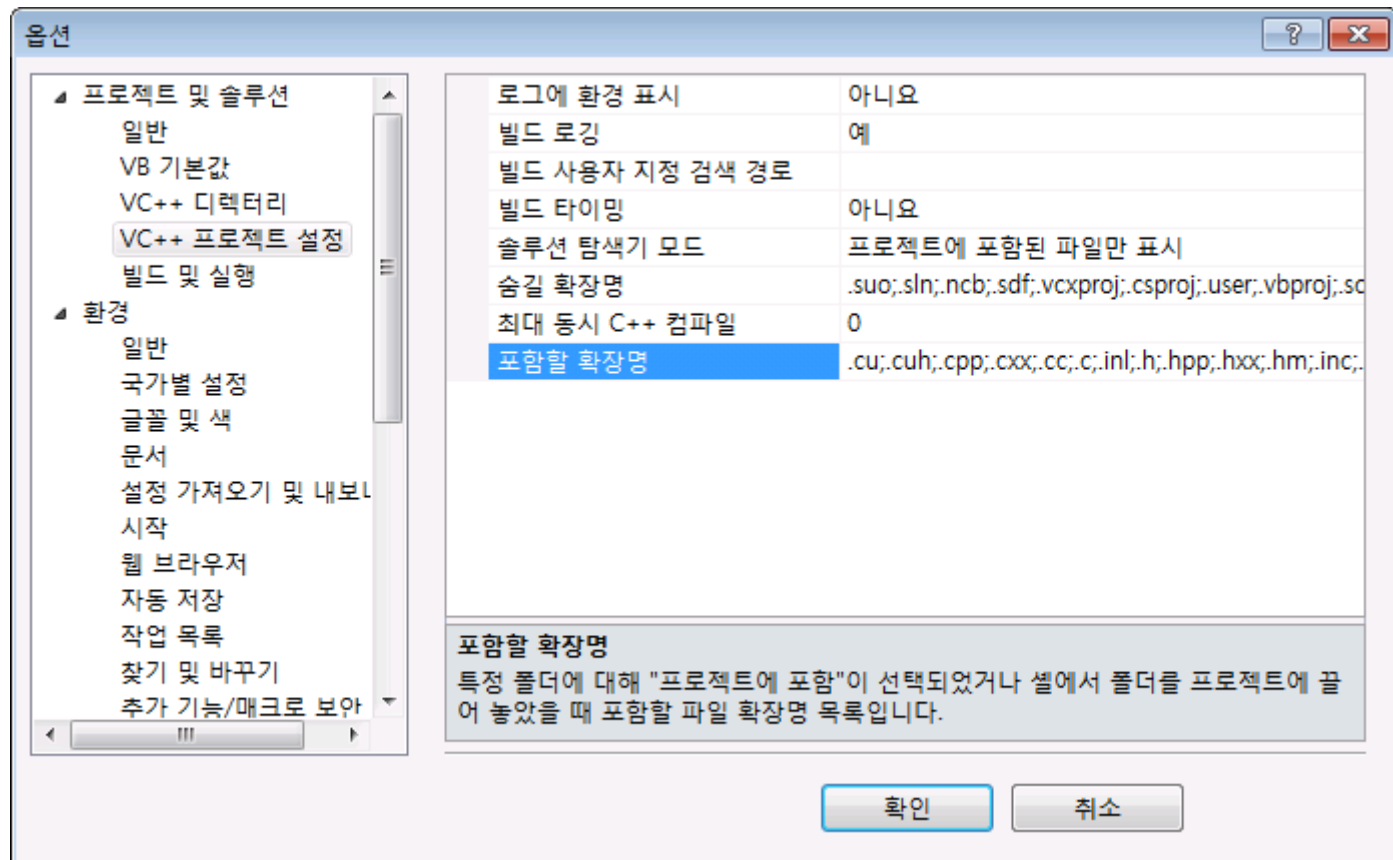
Syntax highlighting

■ 위의 메뉴 -> 도구 -> 옵션 선택



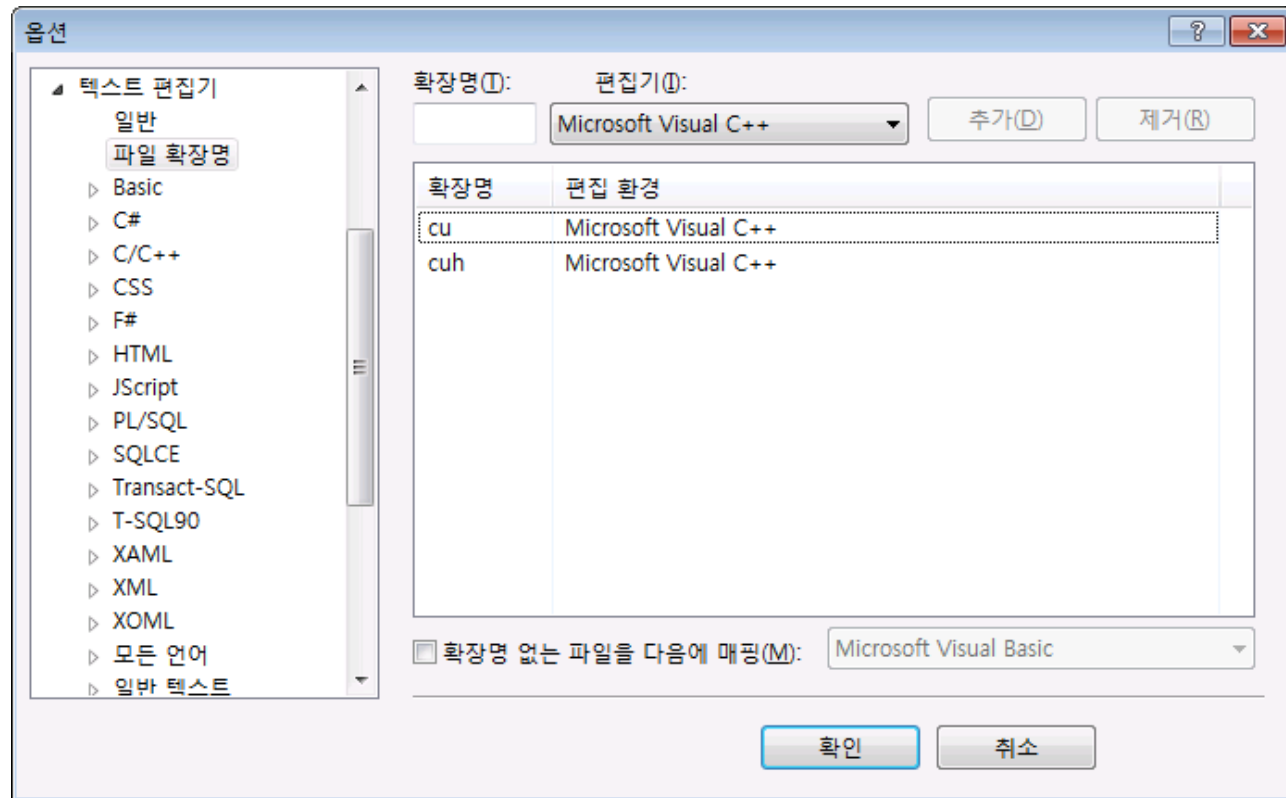
Syntax highlighting

- 프로젝트 및 솔루션 -> VC++ 프로젝트 설정 -> 포함할 확장명에 .cu;.cuh; 추가



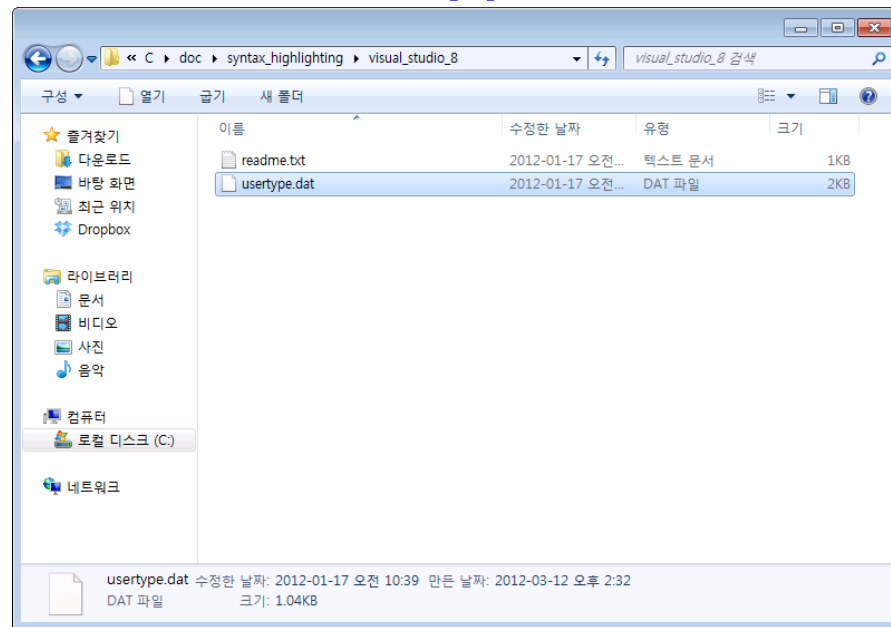
Syntax Highlighting

- 텍스트 편집기 -> 파일 확장명에서 확장명 .cu, .cuh를 편집기 Microsoft Visual C++로 추가



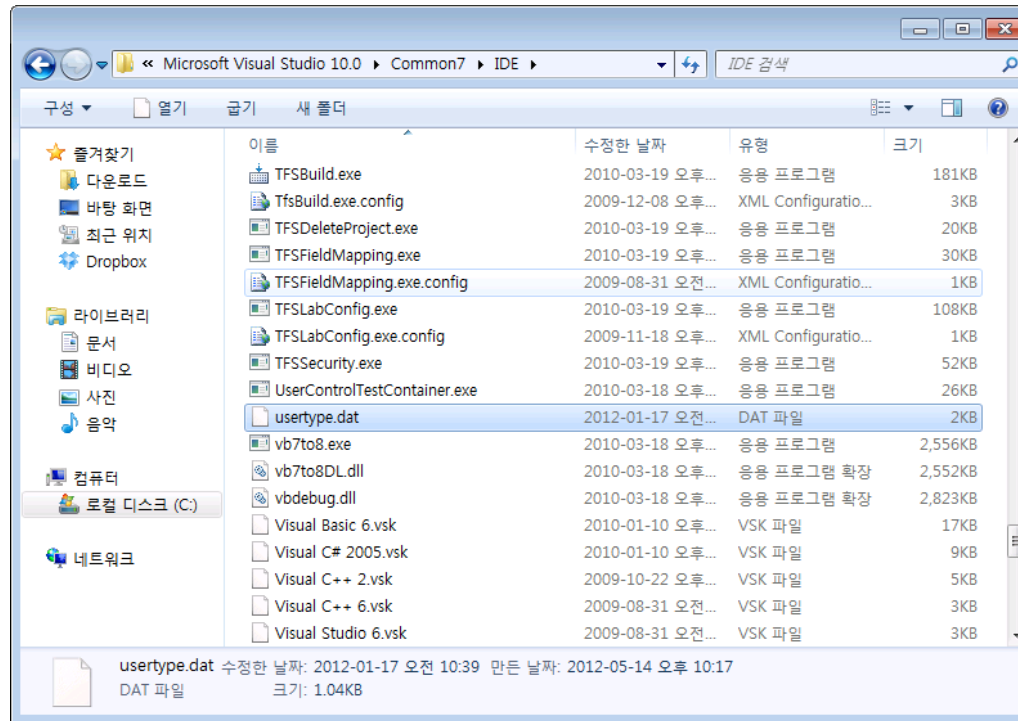
Syntax highlighting

- C:\ProgramData\NVIDIA Corporation\NVIDIA GPU Computing SDK 4.2\C\doc\syntax_highlighting\visual_studio_8에 가면 usertype.dat라는 파일 존재



Syntax highlighting

- 이 usertype.dat 파일을 C:\Program Files\Microsoft Visual Studio 10.0\Common7\IDE에 복사



■ Matrix multiplication

- 정방형의 행렬 곱셈
- $P = M * N$

■ 행렬의 크기를 변화시켜 처리 시간 측정 및 속도 향상 계산

■ CPU로 처리했을 때와 GPU로 함께 처리했을 때를 비교



수행 과제

- CUDA 소스 코드, 결과 Screenshot, 코드에 대한 이해 및 전반적인 지식 내용 보고서에 포함.
- Zip파일
(프로젝트 소스, 결과 Screenshot, 보고서)
- 추가) 메모리의 활용 또는 CUBLAS를 사용하여 속도 향상이 더 일어나는가에 대한 내용 작성