Задание 23. Многопоточная программа «Hello World!» в CUDA.

Напишите CUDA-программу, в которой создаётся 2 блока, содержащих по 2 нити, и каждая нить выводит на экран строку «Hello World!».

Входные данные: нет.

Выходные данные: 4 строки «Hello World!».

Пример входных и выходных данных

Входные данные	Выходные данные
	`Hello World!` `Hello World!` `Hello World!` `Hello World!`

Указания к заданию 23. Многопоточная программа «Hello World!» в CUDA.

Внимание! Для выполнения этого задания требуется графический ускоритель NVIDIA с поддержкой технологии CUDA (присутствует во всех графических ускорителях NVIDIA начиная с серии G8x, включая GeForce, Quadro и Tesla).

В данном примере используется ускоритель NVIDIA GeForce MX130, CUDA 11.4 и Visual Studio 2019.

1. Скачайте последнюю версию CUDA Toolkit с официального сайта NVIDIA (https://developer.nvidia.com/cuda-downloads).
    a. В случае, если последняя версия CUDA Toolkit не поддерживает ваш графический ускоритель, скачайте одну из предыдущих версий (https://developer.nvidia.com/cuda-toolkit-archive).
2. Установите CUDA Toolkit, включая интеграцию с Visual Studio.
3. Создайте пустой проект C++ в среде разработки Visual Studio 2019.
4. Измените разрядность вашего проекта на x64, как показано на рисунке 1.

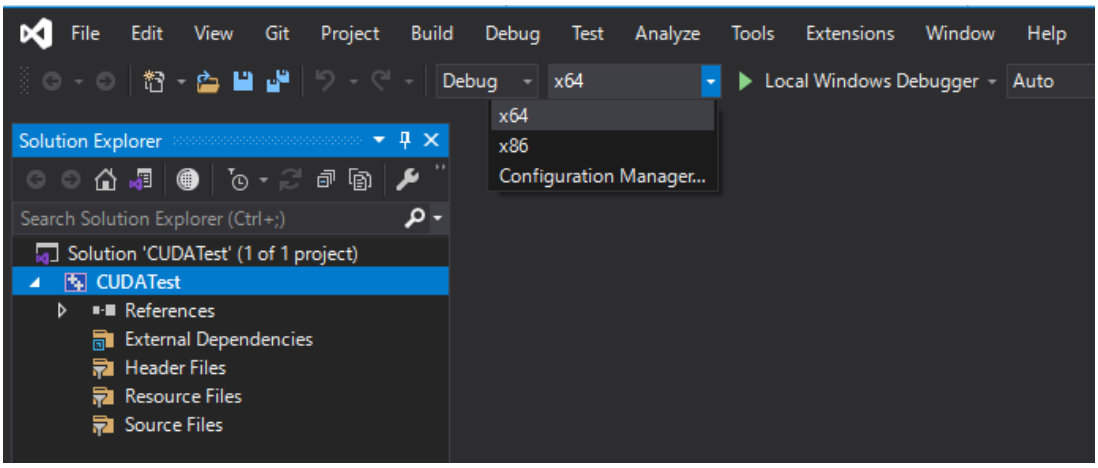

Рис. 1

5. Щёлкните правой кнопкой мыши по проекту в Solution Explorer, выберите Build Dependencies – Build Customizations (рисунок 2).

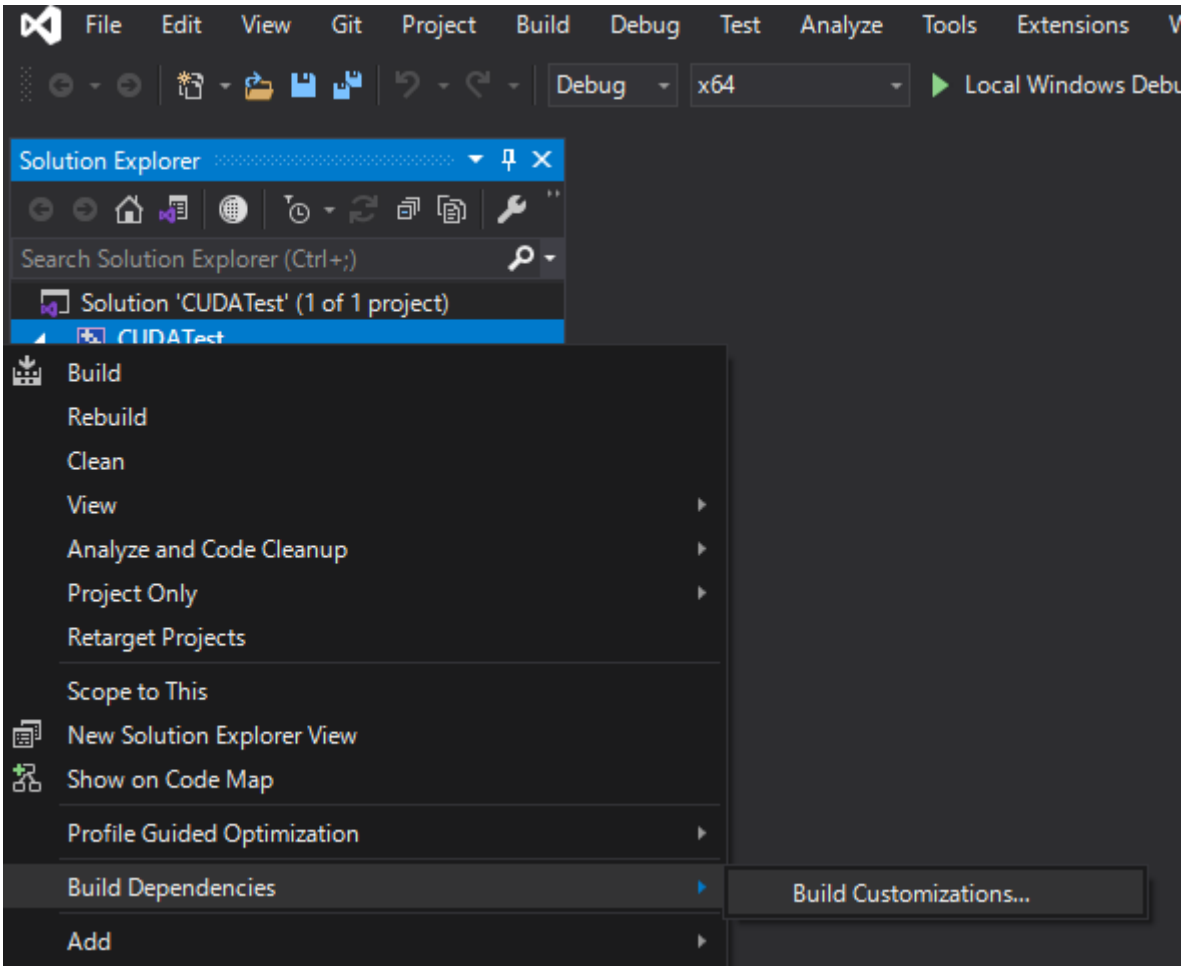

Рис. 2

6. В появившемся меню выберите установленную ранее версию CUDA и нажмите ОК (рисунок 3).

    a. Если в списке отсутствует CUDA, скопируйте содержимое директории MSBuildExtensions:

    C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v11.4\extras\visual_studio_integration\MSBuildExtensions

    В директорию BuildCustomizations:

    C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\MSBuild\Microsoft\VC\v160\BuildCustomizations

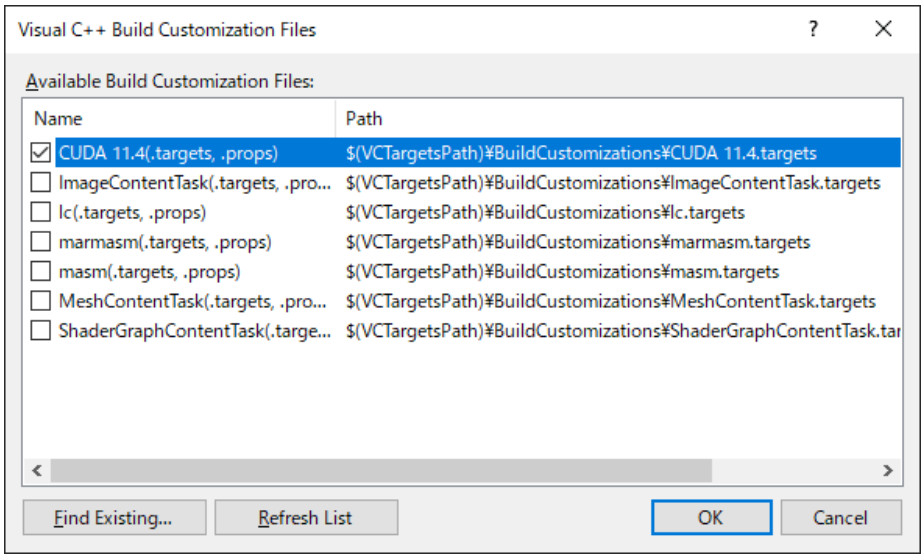

Рис. 3

7. Откройте свойства проекта (Project – Properties), выберите раздел CUDA C/C++ – Command Line. Введите в текстовое поле: -arch *(код архитектуры вашего ускорителя)* (рисунок 4).

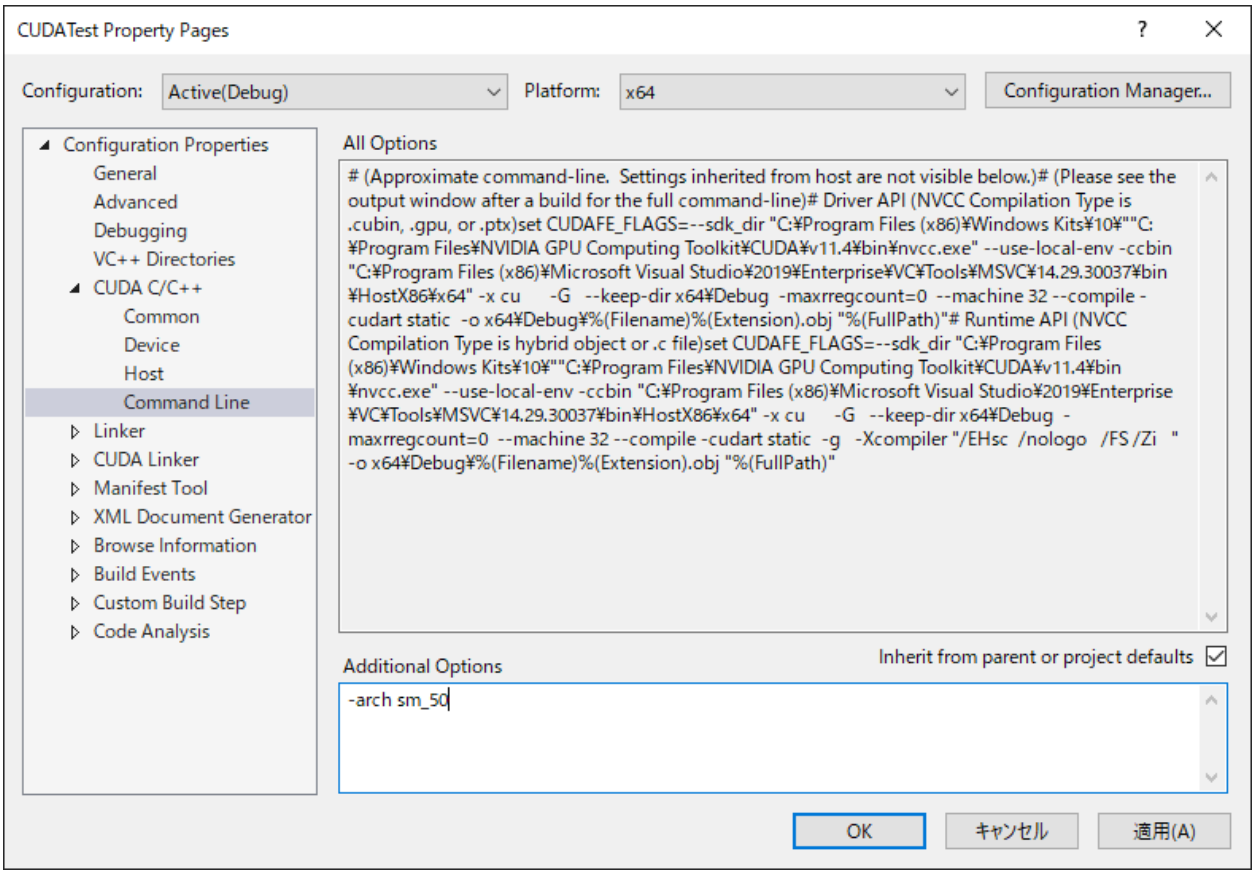

Рис. 4

8. Добавьте в ваш проект файл CUDA 11.4 C/C++ File (рисунок 5).

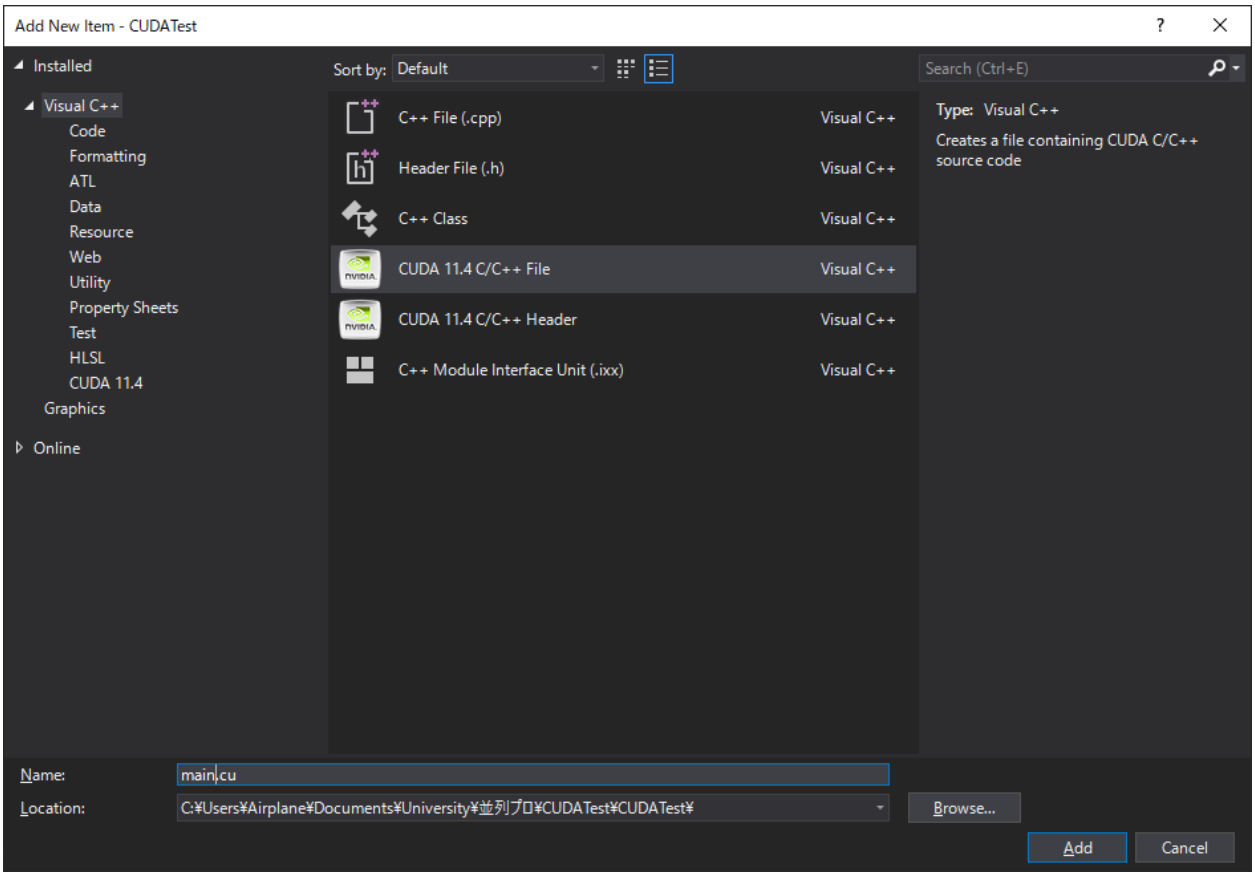

Рис. 5

9. Импортируйте библиотеки iostream и cuda_runtime.h.
10. Создайте функцию-ядро cHello со следующим кодом:

```
__global__ void cHello()
{
	printf("Hello World!\n");
}
```

11. Ознакомьтесь с командой запуска ядра:

kernel<<<execution configuration>>>(params);

execution configuration – Dg, Db, Ns, S

dim3 **Dg** – размеры грида в блоках

Dg.x * **Dg**.y * **Dg**.z – число блоков

dim3 **Db** – размер каждого блока

Db.x * **Db**.y * **Db**.z – число нитей в блоке

size_t **Ns** – размер динамически выделяемой общей памяти (опционально)

cudaStream_t **S** – поток, в котором следует запустить ядро (опционально)

struct **dim3** – структура определенная в CUDA Toolkit

Поля: unsigned **x, y, z**

Конструктор: dim3(unsigned x=1, unsigned y=1, unsigned z=1)

12. Запустите созданное ядро в теле функции main. Используйте два блока и две нити. После запуска ядра используйте следующий код для синхронизации нитей:

```
cudaError_t ce = cudaThreadSynchronize();
if (ce != cudaSuccess)
{
	printf("Error: %s\n", cudaGetErrorString(ce));
}
```

13. Скомпилируйте и запустите приложение средствами Visual Studio 2019 (Debug – Start Debugging или клавиша F5). Убедитесь, что выводится верный результат.