

# CUDA PROGRAMMING CHEAT SHEET

CUDA Application 의 기본적인 구성을 살펴보고, 오늘 배운 내용을 정리해보세요.

## CUDA 프로그래밍 기초

### GPU 메모리 할당

```
cudaMallocManaged((void**)&ptr, size_t byte_buffer);
```

### GPU 로 처리할 데이터 전송

```
cudaMemPrefetchAsync(ptr, byte_size, deviceId);
```

### CUDA KERNEL CODE 작성

```
__global__  
void foo_kernel(ptr) {  
  
    ...  
}
```

### CUDA KERNEL 호출

```
int block_size = _____;  
  
int grid_size = N / block_size;  
  
foo_kernel<<< grid_size, block_size >>>>(ptr, N);
```

### CPU 로 연산 결과 전송 및 동기화

```
cudaMemPrefetchAsync(ptr, byte_size, cudaCpuDeviceId);  
  
cudaDeviceSynchronize();
```

## GPU 메모리 해제

```
cudaFree(ptr);
```

## CUDA BUILT-IN KEYWORDS

CUDA Kernel에서는 CUDA thread로 하여금 자신의 index를 알 수 있도록 다음과 같은 Keyword들을 제공합니다.

<b>GridDim</b>	CUDA block의 개수
<b>BlockDim</b>	CUDA thread block의 개수
<b>BlockIdx</b>	CUDA thread block의 index
<b>ThreadId</b>	CUDA thread block 내의 CUDA thread의 index

## THREAD INDEXING

이를 활용하여 CUDA thread를 다음과 같이 할 수 있습니다.

---

### 1D INDEX

```
int idx = blockDim.x * blockIdx.x + threadIdx.x;
```

---


### 2D INDEX

```
int row = blockDim.y * blockIdx.y + threadIdx.y;
```

```
int row = blockDim.x * blockIdx.x + threadIdx.x;
```

## LOOP REMOVAL

위 indexing을 활용하여 CPU 코드 상에서 loop으로 짜인 코드에서 loop을 하나 없애는 방식으로 병렬화를 할 수 있습니다.

<pre>for (int i = 0; i &lt; N; i++) {     C[i] = A[i] + B[i]; }</pre>		<pre>int i = blockDim.x * blockIdx.x       + threadIdx.x; C[i] = A[i] + B[i];</pre>
---	---	---