

```
Week5 > C q1_1.cu > ...
1  #include <stdio.h>
2  #include <cuda_runtime.h>
3
4  __global__ void vectorAdd(float *A, float *B, float *C, int N) {
5      int i = threadIdx.x;
6      if (i < N) {
7          C[i] = A[i] + B[i];
8      }
9  }
10
11 int main() {
12     int N = 8;
13     size_t size = N * sizeof(float);
14
15     float *h_A, *h_B, *h_C;
16     float *d_A, *d_B, *d_C;
17
18     h_A = (float*)malloc(size);
19     h_B = (float*)malloc(size);
20     h_C = (float*)malloc(size);
21
22     for (int i = 0; i < N; i++) {
23         h_A[i] = i;
24         h_B[i] = i * 2;
25     }
26
27     cudaMalloc(&d_A, size);
28     cudaMalloc(&d_B, size);
29     cudaMalloc(&d_C, size);
30
31     cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
32     cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);
33
34     vectorAdd<<<1, N>>>(d_A, d_B, d_C, N);
35
36     cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost);
37
38     for (int i = 0; i < N; i++) {
39         printf("%f ", h_C[i]);
40     }
41
42     cudaFree(d_A);
43     cudaFree(d_B);
44     cudaFree(d_C);
45     free(h_A);
46     free(h_B);
47     free(h_C);
48
49     return 0;
50 }
51
```

```
Week5 > C q1_2.cu > ...
1 #include <stdio.h>
2 #include <cuda_runtime.h>
3
4 __global__ void vectorAdd(float *A, float *B, float *C, int N) {
5     int i = blockIdx.x * blockDim.x + threadIdx.x;
6     if (i < N) {
7         C[i] = A[i] + B[i];
8     }
9 }
10
11 int main() {
12     int N = 1024;
13     size_t size = N * sizeof(float);
14
15     float *h_A, *h_B, *h_C;
16     float *d_A, *d_B, *d_C;
17
18     h_A = (float*)malloc(size);
19     h_B = (float*)malloc(size);
20     h_C = (float*)malloc(size);
21
22     for (int i = 0; i < N; i++) {
23         h_A[i] = i;
24         h_B[i] = i * 2;
25     }
26
27     cudaMalloc(&d_A, size);
28     cudaMalloc(&d_B, size);
29     cudaMalloc(&d_C, size);
30
31     cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
32     cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);
33
34     int threadsPerBlock = 256;
35     int blocks = (N + threadsPerBlock - 1) / threadsPerBlock;
36
37     vectorAdd<<<blocks, threadsPerBlock>>>(d_A, d_B, d_C, N);
38
39     cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost);
40
41     for (int i = 0; i < 10; i++) {
42         printf("%f ", h_C[i]);
43     }
44
45     cudaFree(d_A);
46     cudaFree(d_B);
47     cudaFree(d_C);
48     free(h_A);
49     free(h_B);
50     free(h_C);
51
52     return 0;
53 }
```

```
● (dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ nvcc ql_1.cu -o ql
● (dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ ./ql
0.000000 3.000000 6.000000 9.000000 12.000000 15.000000 18.000000 21.000000 (dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ nvcc ql_2.cu -o ql_2
● (dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ ./ql_2
0.000000 3.000000 6.000000 9.000000 12.000000 15.000000 18.000000 21.000000 24.000000 27.000000 (dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ ^C
○ (dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ █
```

```
q2.cu      X  q3.cu

Week5 > q2.cu > ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <cuda_runtime.h>
4
5 __global__ void vectorAdd(float *A, float *B, float *C, int N) {
6     int i = blockIdx.x * blockDim.x + threadIdx.x;
7     if (i < N) {
8         C[i] = A[i] + B[i];
9     }
10}
11
12 int main() {
13     int N;
14     scanf("%d", &N);
15
16     size_t size = N * sizeof(float);
17
18     float *h_A = (float*)malloc(size);
19     float *h_B = (float*)malloc(size);
20     float *h_C = (float*)malloc(size);
21
22     for (int i = 0; i < N; i++) {
23         h_A[i] = i;
24         h_B[i] = i * 2;
25     }
26
27     float *d_A, *d_B, *d_C;
28
29     cudaMalloc(&d_A, size);
30     cudaMalloc(&d_B, size);
31     cudaMalloc(&d_C, size);
32
33     cudaMemcpy(d_A, h_A, size, cudaMemcpyHostToDevice);
34     cudaMemcpy(d_B, h_B, size, cudaMemcpyHostToDevice);
35
36     int threadsPerBlock = 256;
37     int blocks = (N + threadsPerBlock - 1) / threadsPerBlock;
38
39     vectorAdd<<<blocks, threadsPerBlock>>>(d_A, d_B, d_C, N);
40
41     cudaMemcpy(h_C, d_C, size, cudaMemcpyDeviceToHost);
42
43     for (int i = 0; i < N; i++) {
44         printf("%f ", h_C[i]);
45     }
46
47     cudaFree(d_A);
48     cudaFree(d_B);
49     cudaFree(d_C);
50
51     free(h_A);
52     free(h_B);
53     free(h_C);
54
55     return 0;
56 }
57
```

```
(dse) mca@computinglab25:~/Desktop/PPL_230962278/Week5$ nvcc q2.cu -o q2
(dse) mca@computinglab25:~/Desktop/PPL_230962278/Week5$ ./q2
20
0.000000 3.000000 6.000000 9.000000 12.000000 15.000000 18.000000 21.000000 24.000000 27.000000 30.000000 33.000000 36.000000 39.000000 42.000000 45.000000 48.000000 51.00000
```

```
q3.cu      x
Week5 > q3.cu > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <cuda_runtime.h>
5
6  __global__ void computeSine(float *angles, float *output, int N) {
7      int i = blockIdx.x * blockDim.x + threadIdx.x;
8      if (i < N) {
9          output[i] = sinf(angles[i]);
10     }
11 }
12
13 int main() {
14     int N;
15     scanf("%d", &N);
16
17     size_t size = N * sizeof(float);
18
19     float *h_angles = (float*)malloc(size);
20     float *h_output = (float*)malloc(size);
21
22     for (int i = 0; i < N; i++) {
23         scanf("%f", &h_angles[i]);
24     }
25
26     float *d_angles, *d_output;
27
28     cudaMalloc(&d_angles, size);
29     cudaMalloc(&d_output, size);
30
31     cudaMemcpy(d_angles, h_angles, size, cudaMemcpyHostToDevice);
32
33     int threadsPerBlock = 256;
34     int blocks = (N + threadsPerBlock - 1) / threadsPerBlock;
35
36     computeSine<<<blocks, threadsPerBlock>>>(d_angles, d_output, N);
37
38     cudaMemcpy(h_output, d_output, size, cudaMemcpyDeviceToHost);
39
40     for (int i = 0; i < N; i++) {
41         printf("%f ", h_output[i]);
42     }
43
44     cudaFree(d_angles);
45     cudaFree(d_output);
46
47     free(h_angles);
48     free(h_output);
49
50     return 0;
51 }
52
```

```
● (dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ nvcc q3.cu -o q3
● (dse) mca@computinglab25-22:~/Desktop/PPL_230962278/Week5$ ./q3
4
0
1.5708
3.1416
4.7124
0.000000 1.000000 -0.000007 -1.000000 (dse) mca@computinglab25-22:~/D
```