```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>

#include <cuda_runtime.h>

__global__ void squareKernel(float *d_in, float *d_out)
{
    const unsigned int tid = threadIdx.x; // access thread id
    d_out[tid] = d_in[tid]*d_in[tid];     // do computation
}

int main(int argc, char **argv)
{
    unsigned int num_threads = 32;
    unsigned int mem_size = sizeof(float) * num_threads;

    // allocate host memory
    float *h_in = (float *)malloc(mem_size);
    float *h_out = (float *) malloc(mem_size);

    // initalize the memory
    for (unsigned int i = 0; i < num_threads; ++i){
        h_in[i] = (float) i;
    }

    // allocate device memory
    float *d_in;
    float *d_out;
    cudaMalloc((void **) &d_in, mem_size);
    cudaMalloc((void **) &d_out, mem_size);

    // copy host memory to device
    cudaMemcpy(d_in, h_in, mem_size, cudaMemcpyHostToDevice);

    // execute the kernel
    squareKernel<<< 1, num_threads >>>(d_in, d_out);

    // copy result from device to host
    cudaMemcpy(h_out, d_out, sizeof(float) * num_threads, cudaMemcpyDeviceToHost);

    for (unsigned int i=0;i<num_threads; ++i){
        printf("%.1f\n",h_out[i]);
    }

    // cleanup memory
    free(h_in);
    free(h_out);
    cudaFree(d_in);
    cudaFree(d_out);

    return 0;
}
```