Задание 25. Распараллеливание циклов в CUDA: программа «Число π».

Напишите CUDA-программу, которая вычисляет число π с точностью до N знаков после запятой. Используйте следующую формулу:

$$\pi = \left(\frac{4}{1+x_0^2} + \frac{4}{1+x_1^2} + \cdots + \frac{4}{1+x_{N-1}^2}\right) * \frac{1}{N}, \text{где } x_i = (i+0.5) * \frac{1}{N}, i = \overline{0, N-1}$$

Входные данные: одно целое число N (точность вычисления).

Выходные данные: одно вещественное число pi.

Пример входных и выходных данных

Входные данные	Выходные данные
10000000	3.14159265

Указания к заданию 25. Распараллеливание циклов в CUDA: программа «Число π».

1. Создайте проект с поддержкой CUDA (см. задание 1).
2. Напишите функцию для вычисления числа π, используя за основу следующую функцию:

```
double CalcPi(const int n)
{
		double pi = 0;
		const double coef = 1.0 / n;

		for (int i = 0; i < n; ++i)
		{
			const double xi = (i + 0.5) * coef;
			pi += 4.0 / (1.0 + xi * xi);
		}
		return pi * coef;
}
```

3. Для хранения вычисленных каждой нитью значений используйте shared memory (общая память для блока нитей):

```
extern __shared__ double sdata[];
```

4. Для более эффективной редукции значений, вычисленных каждой нитью в рамках блока, используйте метод Interleaved addressing:

```
for (int i = 1; i < blockDim.x; i *= 2)
{
	int index = 2 * i * threadIdx.x;
	if (index < blockDim.x)
	{
```

```
            sdata[index] += sdata[index + i];
        }
        __syncthreads();
}
```

5. Для редукции значений, вычисленных блоками, используйте глобальную память.
6. Выполните запуск ядра, скопируйте результат выполнения на хост и выведите его на экран.
7. Скомпилируйте и запустите ваше приложение. Убедитесь, что выводится верный результат.