МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Кафедра вычислительной математики и программирования

**спецкурс «Параллельное программирование»**

**ОТЧЕТ**

**Лабораторная работа № 1**

**«Освоение программного обеспечения среды программирования NVIDIA»**

Выполнил: Сапетный В.Г.

Группа: 8-114M, **вариант 19**

Преподаватель: Семенов С. А.

Москва, 2022

Оглавление

[1. Постановка задачи 2](#_Toc114078323)

[2. Описание решения 2](#_Toc114078324)

[3. Основные моменты кода 2](#_Toc114078325)

[4. Результат работы программы 4](#_Toc114078326)

[5. Выводы 4](#_Toc114078327)

# 1. Постановка задачи

Вариант задания предполагает расписать геометрическую прогрессию.

# 2. Описание решения

Происходит вычисление n-ого элемента геометрической прогрессии по формуле , где – члены прогрессии, а – знаменатель прогрессии.

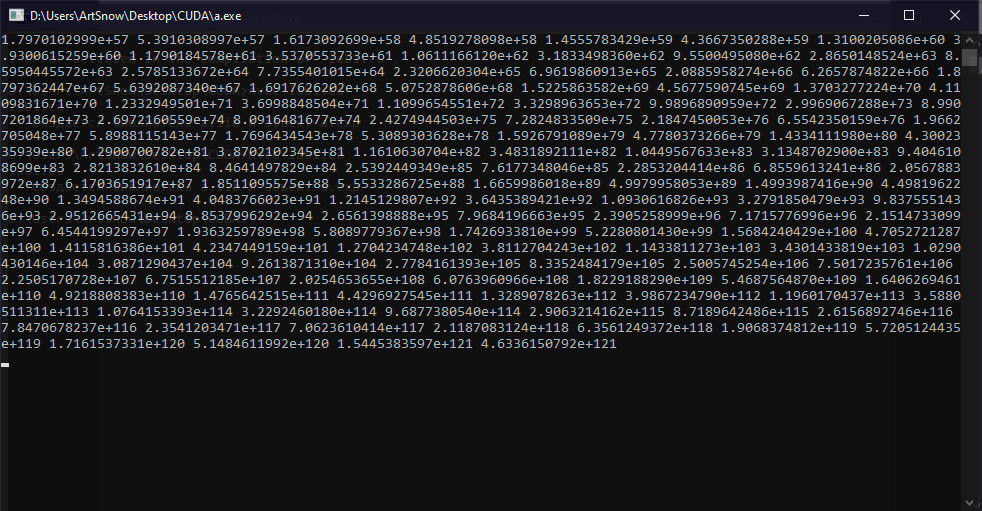
# 3. Основные моменты кода

Мы объявляем две переменные – первый элемент геометрической прогрессии и знаменатель прогрессии, а также массив для остальных элементов. Далее мы выделяем память для массива ответов в Device. Затем мы вычисляем значение всех элементов геометрической прогрессии с помощью первого элемента прогрессии и знаменателя прогрессии и возвращаем на Host полученный результат.

Код на GitHub находится по ссылке https://github.com/ArtSnow/Univ\_PGP

|  |
| --- |
| **Файл lab1.cu, проект Univ\_PGP** |
| #include "cuda\_runtime.h"  #include "device\_launch\_parameters.h"  #include <iostream>  #include <math.h>  #include <cstdlib>  using namespace std;  \_\_global\_\_ void kernel(double b, double q, double\* ans, long long n) { //отличное от C++ (\_\_global\_\_)  long long i, idx = blockDim.x \* blockIdx.x + threadIdx.x; // Абсолютный номер потока  long long offset = blockDim.x \* gridDim.x; // Общее кол-во потоков  for (i = idx; i < n; i += offset) // Для всех требование - внутри цикла for()  ans[i] = b\*pow(q, i);  }  int main() {  ios::sync\_with\_stdio(false);  cin.tie(nullptr);  long long n;  double b = 1;  double q = 3;  cin >> n;  double\* answer = (double\*)malloc(sizeof(double) \* n); //выделение массива ответа  double\* result;  cudaMalloc(&result, sizeof(double) \* n); //выделение массива на устройстве  cudaMemcpy(result, answer, sizeof(double) \* n, cudaMemcpyHostToDevice);  kernel <<<256, 256>>> (b, q, result, n); //отличное от C++ (<<<>>>), стандартная функция  // Многопоточное  // 256 блоков и 256 потоков(Thread)  cudaMemcpy(answer, result, sizeof(double) \* n, cudaMemcpyDeviceToHost);  cudaFree(result);  cout.precision(10);  cout.setf(ios::scientific);  for (long long i = 0; i < n; i++)  cout << answer[i] << ' ';  cout << endl;  free(answer);  cin >> n;  return 0;  } |

# 4. Результат работы программы



# 5. Выводы

Во время выполнения лабораторной работы №1 были изучены основы технологии CUDA а также применены для решения простых задач на примере вычисления элементов геометрической прогрессии.