**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ   
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

**Домашнее задание 3**

**Программа определения чисел порядка от 104 до 109, сохраняющих набр своих цифр при умножении на данное число**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

Исполнитель

Студент группы БПИ 199

\_\_\_\_\_\_\_\_/Мостачев А.О./

«\_\_\_»\_\_\_\_\_\_\_\_\_\_2020 г.

Москва

2020

# Содержание

[Содержание 2](#_Toc56549692)

[1. Постановка задачи 3](#_Toc56549693)

[2. Применяемые расчетные методы и хранение данных 4](#_Toc56549694)

[3. Входные и выходные данные 5](#_Toc56549695)

[4. Использованные источники 6](#_Toc56549696)

[5. Приложение 1 7](#_Toc56549697)

[6. Приложение 2 8](#_Toc56549698)

# Постановка задачи

Вывести список всех целых чисел, содержащих от 4 до 9 значащих цифр, которые после умножения на n, будут содержать все те же самые цифры в произвольной последовательности и в произвольном количестве. Входные данные: целое положительное число n, больше единицы и меньше десяти. Количество потоков является входным параметром. Программа должна быть реализована с помощью методов параллельного программирования.

# Применяемые расчетные методы и хранение данных

Для работы с потоками в программе используются инструментов хэдера thread. В программе используется модель итеративного параллелизма. Определение сохранения чисел обеспечивается методом SameDigits и реализовано с помощью двух unordered\_set, содержащих в себе цифры, которые уже встретились в числе и те, которые еще должны в нем появиться. Поиск в unordered\_set происходит за константу, поэтому была выбрана именно эта структура данных. Вывод происходит через перенаправленный стандартный поток. Этот способ был выбран эмпирически, так как оказался самым быстрым.

# Входные и выходные данные

Входные данные подаются программе через аргументы командной строки. Программа принимает 2 аргумента: число от 2 до 9, на которое умножаются все числа рассматриваемой области, и количество потоков – число от 1 до 1000.  
Вывод программы идет в файл output.txt. В выходном файле на каждой строке пишется одно подходящее под условие число. В последней строке отражено время работы программы.

# Использованные источники

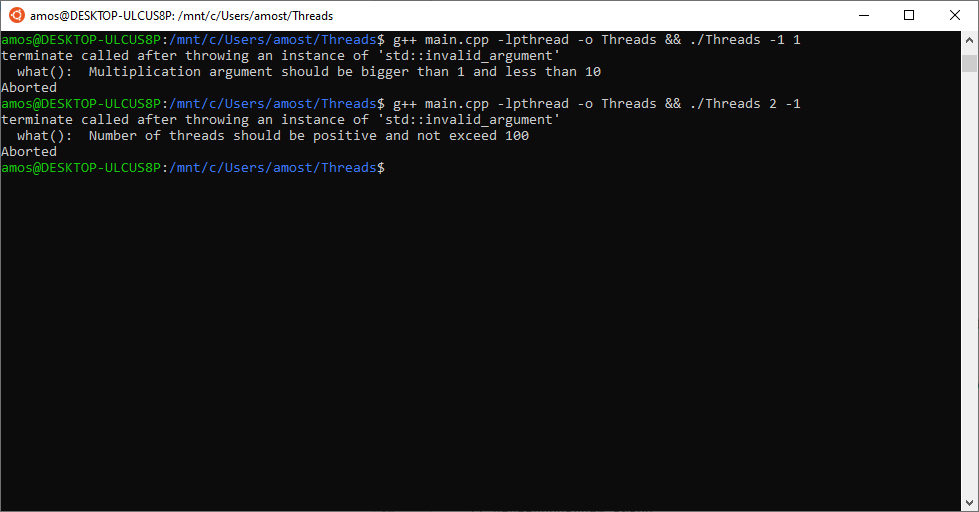
1. Написание многопоточных приложений на C++

<https://eax.me/cpp-multithreading/>

# Приложение 1

**Тестирование программы**

Файлы с результатами работы программы представлены в папке с файлом с кодом. Их названия – outputn, где n - число из условия. Все файлы из этой папки были запущены с 10 потоками. Результаты с 1 и 5 потоками представлены в папках 1\_thread и 5\_threads.

Некорректные входные данные:  


# Приложение 2

**Текст программы**

#include <stdexcept>  
#include <string>  
#include <thread>  
#include <unordered\_set>  
#include <iostream>  
#include <vector>  
#include <utility>  
#include <mutex>  
#include <chrono>  
  
std::mutex mtx;  
size\_t lower = 1e3;  
size\_t upper = 1e9;  
  
void SameDigits(size\_t multiplicant, size\_t lower, size\_t upper);  
  
int main(int argc, char\* argv[]) {  
 if (argc != 3) throw std::runtime\_error("Two console parameters expected");  
  
 size\_t multiplicant = std::stoi(argv[1]);  
 if (multiplicant < 2 || multiplicant > 9) throw std::invalid\_argument("Multiplication argument should be bigger than 1 and less than 10");  
  
 size\_t num\_of\_threads = std::stoi(argv[2]);  
 if (num\_of\_threads < 1 || num\_of\_threads >= 1000) throw std::invalid\_argument("Number of threads should be positive and not exceed 100");  
 size\_t nums\_per\_thread = (upper - lower - 1) / num\_of\_threads;  
  
 *// Засекаем время* auto start = std::chrono::high\_resolution\_clock::now();  
  
 *// Перенаправляем вывод* freopen("./output.txt", "w", **stdout**);  
  
 std::vector<std::thread> threads;  
  
 *// Инициализируем потоки* for (size\_t i = 0; i < num\_of\_threads - 1; i++)  
 {  
 std::thread thr(SameDigits, multiplicant, lower + i \* nums\_per\_thread, lower + (i + 1) \* nums\_per\_thread);  
 threads.emplace\_back(std::move(thr));  
 }  
  
 std::thread thr(SameDigits, multiplicant, lower + num\_of\_threads \* nums\_per\_thread, upper);  
 threads.emplace\_back(std::move(thr));  
  
 *// Джоин* for (auto& thr : threads) {  
 thr.join();  
 }  
  
 *// Отсечка времени* auto finish = std::chrono::high\_resolution\_clock::now();  
 std::chrono::duration<double> elapsed = finish - start;  
 std::cout << "Elapsed time: " << elapsed.count() << std::endl;  
}  
  
void SameDigits(size\_t multiplicant, size\_t lower, size\_t upper) {  
 for (size\_t i = lower; i < upper; i++)  
 {  
 *// Получаем цифры текущего числа* size\_t cur = i;  
 std::unordered\_set<uint8\_t> digits;  
 while (cur != 0) {  
 digits.insert(cur % 10);  
 cur /= 10;  
 }  
  
 *// Цифры, которые уже использованы в произведении* std::unordered\_set<size\_t> used;  
  
 *// Умножаем* cur = i \* multiplicant;  
 bool nice = true;  
  
 *// Если цифра есть в исходном числе, убираем оттуда, добавляем в использованные  
 // Если нет ни в исходных, ни в использованных, значит число неподходящее* while (cur != 0) {  
 size\_t rem = cur % 10;  
 if (digits.count(rem)) {  
 digits.erase(rem);  
 used.insert(rem);  
 }  
 else if (!used.count(rem)) {  
 nice = false;  
 break;  
 }  
 cur /= 10;  
 }  
  
 if (!(nice && digits.size() == 0)) continue;  
  
 printf("%lu\n", i);  
 }  
}