Правительство Российской Федерации

Национальный исследовательский университет

«Высшая школа экономики»

Факультет компьютерных наук

Департамент программной инженерии

Домашнее задание 4

Вариант 15

Выполнил

Студент группы БПИ193

Минец Максим

mvminets@edu.hse.ru

Москва 2020

Оглавление

1. Задание

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.

2. Решение задачи

2.1 Формулировка задания

Другими словами, задача заключается в том, чтобы пройти все числа от 1000 до 999999999, которые при умножении на число n (лежащее в промежутке от 1 до 10), содержит все числа, лежащие в промежутке от 1000 до 999999999 и вывести эти числа в консоль.

2.2 Решение задачи

Прочесть входные данные (см. пункт 2.3), запустить метод выбора, подходящих по условию чисел.Весь массив (автоматически) равномерно распределяется между всеми потоками.

2.3 Формат ввода данных

Ввод входных данных осуществлен через консоль (не через командную строку). Для запуска необходимо ввести в консоль два аргумента: число n (больше единицы и меньше десяти – ограничения заданы в условии задачи) и количество потоков (больше нуля и меньше тысячи – данные границы установлены на усмотрение разработчика). Пример входных данных продемонстрирован на рис. 1.

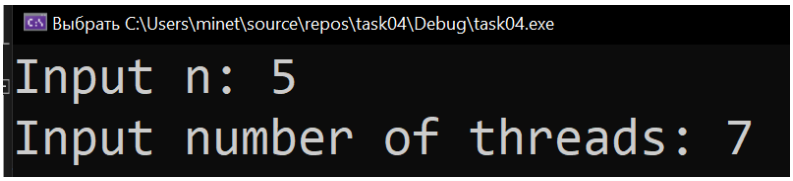


рис. 1

2.4 Формат выводы данных

Результат программы – три числа:

1) num – число, которое при умножении на число n (лежащее в промежутке от 1 до 10), содержит все числа, лежащие в промежутке от 1000 до 999999999;

2) n*num – число, которое является результатом произведения чисел num (см. пункт выше) и n.

3) current thread – номер потока, который выполнил проверку для числа num.

Каждая итерация выводится с новой строки. Пример выходных данных продемонстрирован на рис. 2 и рис. 3.

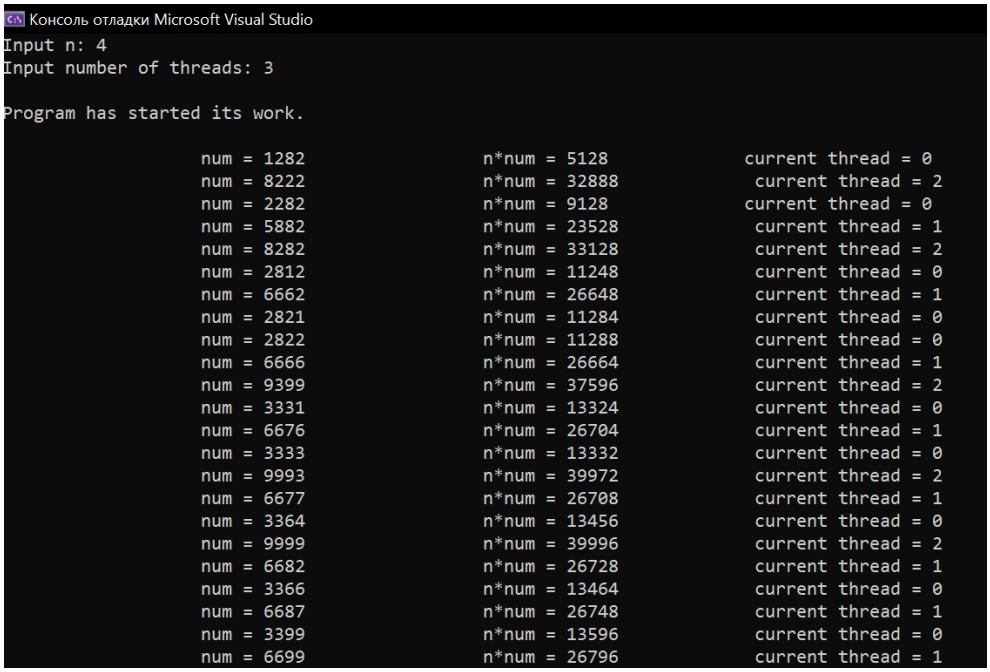


рис. 2

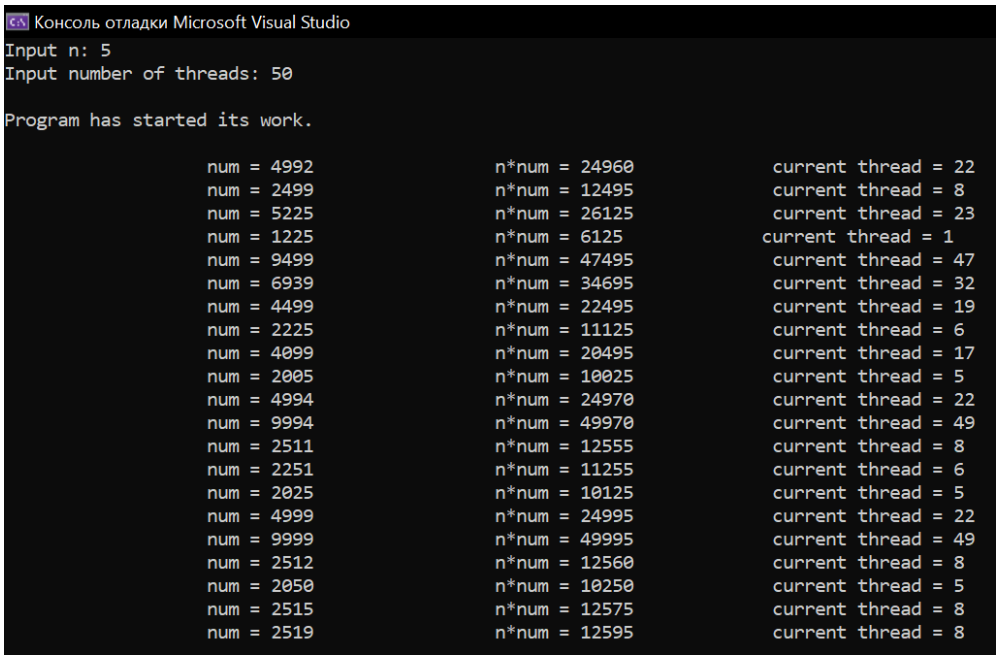


рис. 3

Также программа уведомляет о начале (см. рис. 4) и о завершении (см. рис. 5) своей работы.

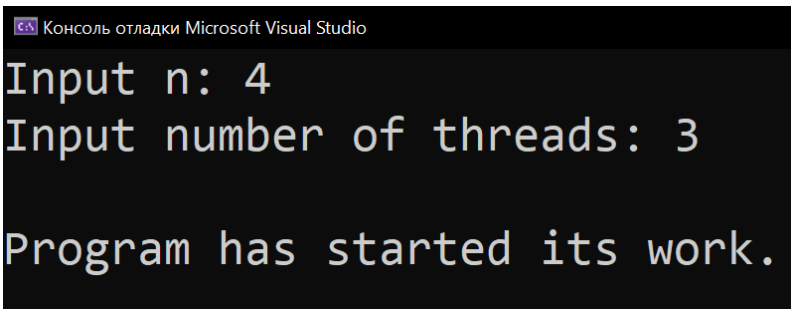

Консоль отладки Microsoft Visual Studio
Input n: 4
Input number of threads: 3
Program has started its work.

рис. 4

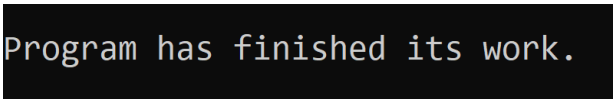
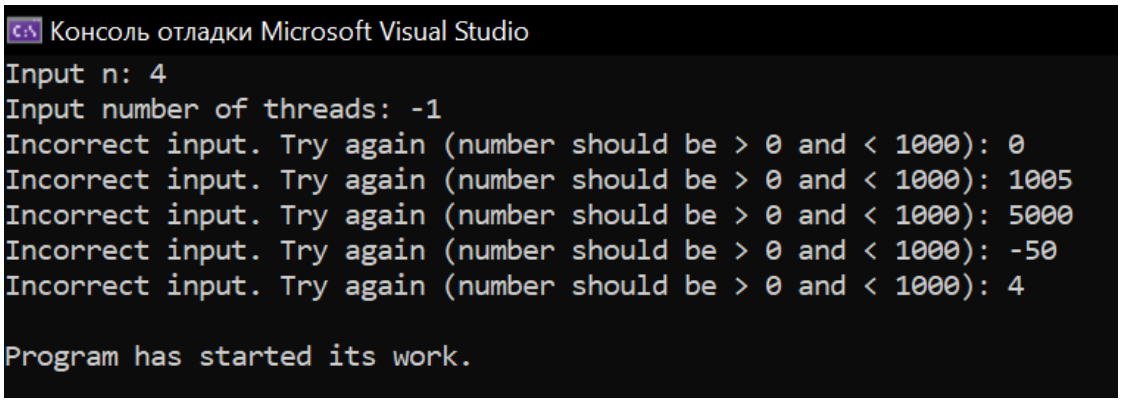

Program has finished its work.

рис. 5

3. Тестирование программы

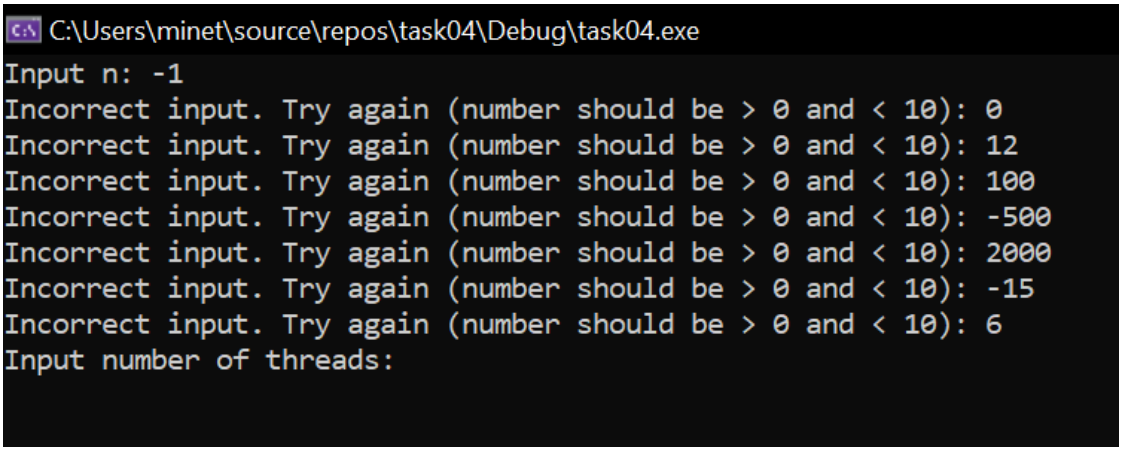
3.1 Тест № 1

При некорректном вводе количества потоков, программа предупреждает о некорректном вводе и просит пользователя ввести число еще раз. Максимальным числом потоков взято число 1000 (на усмотрение разработчика).



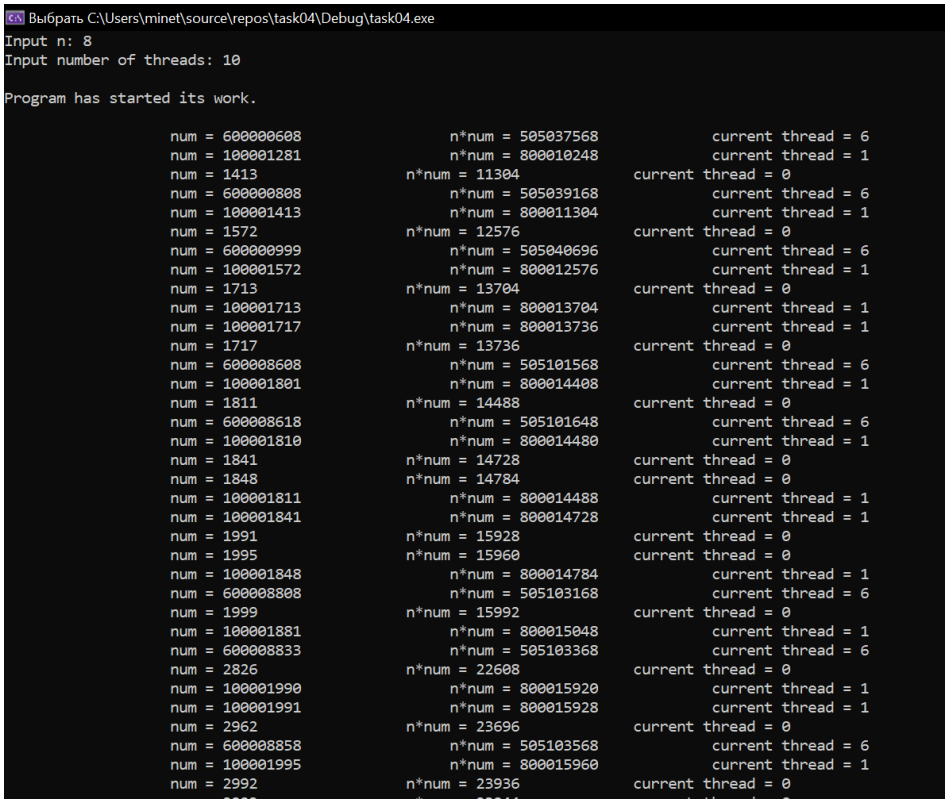
3.2 Тест № 2

При некорректном вводе числа n, программа предупреждает о некорректном вводе и просит пользователя ввести число еще раз. Пограничными значениями являются числа 1 и 10 (по условию задачи).



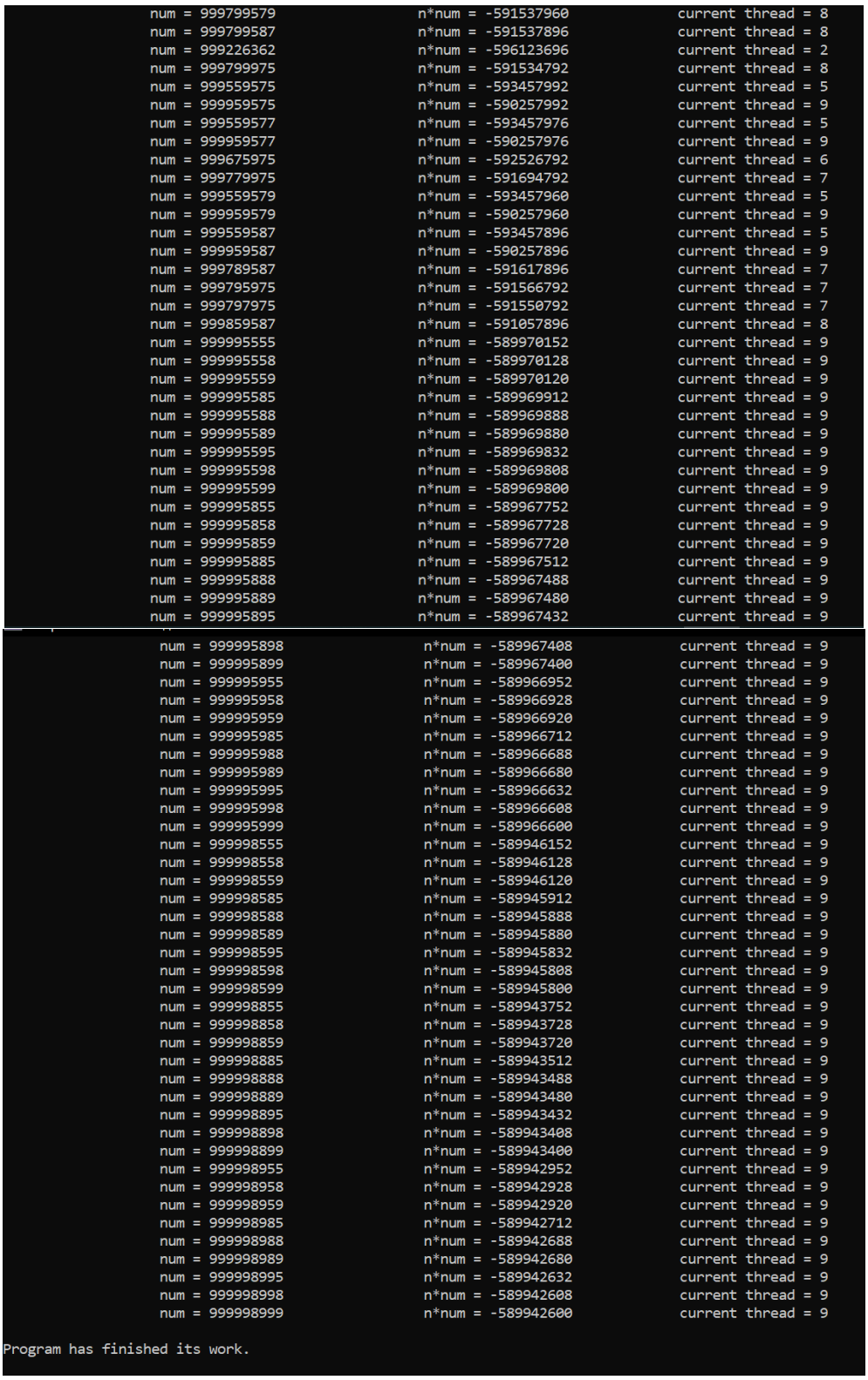
3.3 Тест № 3

Работа программы при корректных данных. Число n = 8, число потоков = 10.



(тут еще много циферок)

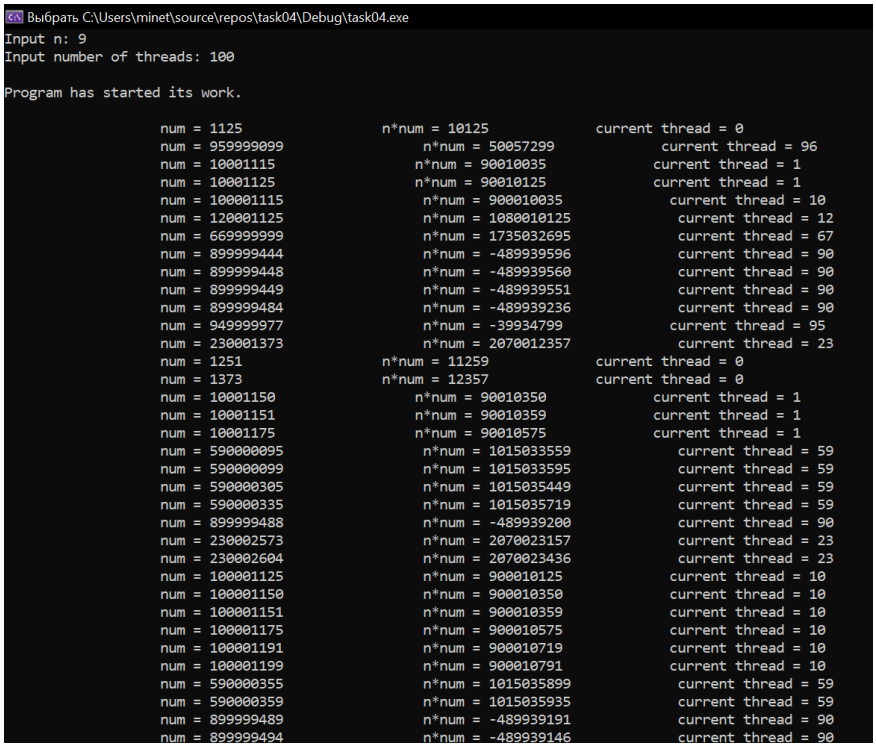
(продолжение теста см. на следующий странице)



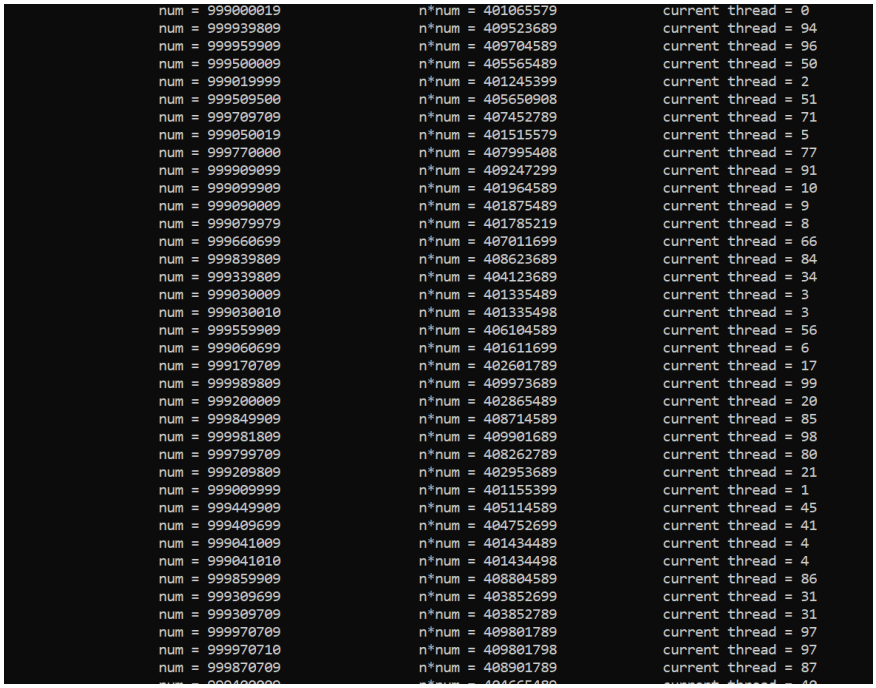
Примечание к тесту № 3: (n*num) является отрицательным из-за переполнения типа данных int.

3.4 Тест № 4

Работа программы при корректных данных. Число n = 9, число потоков = 100.



…



…

(продолжение теста см. на следующий странице)

…

```
num = 999022620          n*num = 401268988          current thread = 2
num = 999033919          n*num = 401370679          current thread = 3
num = 999019499          n*num = 401240899          current thread = 1
num = 999022621          n*num = 401268997          current thread = 2
num = 999033939          n*num = 401370859          current thread = 3
num = 999019629          n*num = 401242069          current thread = 1
num = 999022629          n*num = 401269069          current thread = 2
num = 999033979          n*num = 401371219          current thread = 3
num = 999019699          n*num = 401242699          current thread = 1
num = 999022669          n*num = 401269429          current thread = 2
num = 999033999          n*num = 401371399          current thread = 3
num = 999019709          n*num = 401242789          current thread = 1
num = 999019710          n*num = 401242798          current thread = 1
num = 999022699          n*num = 401269699          current thread = 2
num = 999019809          n*num = 401243689          current thread = 1
num = 999022779          n*num = 401270419          current thread = 2
num = 999019829          n*num = 401243869          current thread = 1
num = 999022799          n*num = 401270599          current thread = 2
num = 999019909          n*num = 401244589          current thread = 1
num = 999038709          n*num = 401413789          current thread = 3
num = 999022809          n*num = 401270689          current thread = 2
num = 999019910          n*num = 401244598          current thread = 1
num = 999022829          n*num = 401270869          current thread = 2
num = 999039809          n*num = 401423689          current thread = 3
num = 999019919          n*num = 401244679          current thread = 1
num = 999022909          n*num = 401271589          current thread = 2
num = 999019921          n*num = 401244697          current thread = 1
num = 999022910          n*num = 401271598          current thread = 2
num = 999019929          n*num = 401244769          current thread = 1
num = 999022919          n*num = 401271679          current thread = 2
num = 999019944          n*num = 401244904          current thread = 1
num = 999022921          n*num = 401271697          current thread = 2
num = 999019949          n*num = 401244949          current thread = 1
num = 999022929          n*num = 401271769          current thread = 2
num = 999022979          n*num = 401272219          current thread = 2
num = 999022999          n*num = 401272399          current thread = 2
num = 999026329          n*num = 401302369          current thread = 2

Program has finished its work.
```

3.5 Тест № 5

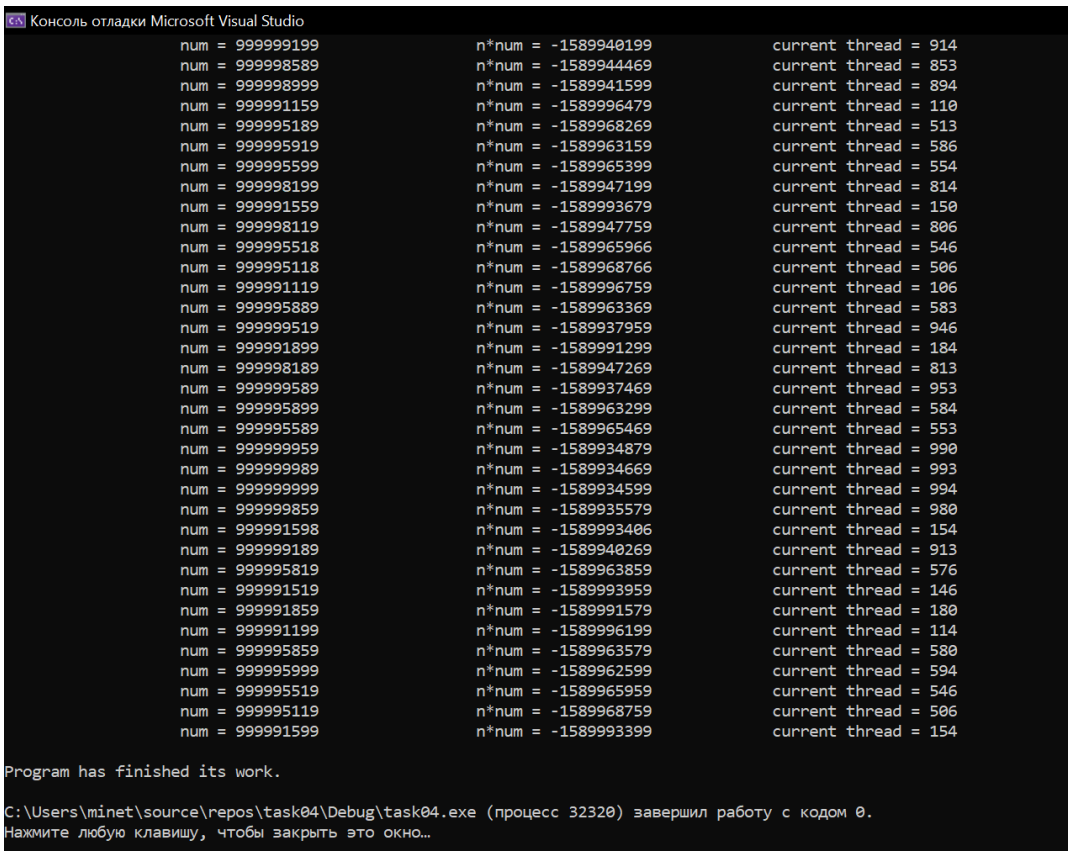
Работа программы при корректных данных, где число потоков будет приближено к 1000. Число n = 7, число потоков = 995.

```
Выбрать C:\Users\minet\source\repos\task04\Debug\task04.exe
Input n: 7
Input number of threads: 995

Program has started its work.

                num = 1166              n*num = 8162            current thread = 0
                num = 901506855              n*num = 2015580689            current thread = 897
                num = 888441722              n*num = 1924124758            current thread = 884
                num = 901506995              n*num = 2015581669            current thread = 897
                num = 901506999              n*num = 2015581697            current thread = 897
                num = 901507005              n*num = 2015581739            current thread = 897
                num = 555778874              n*num = -404515178            current thread = 553
                num = 649246299              n*num = 249756797           current thread = 646
                num = 977888819              n*num = -1744712859             current thread = 973
                num = 919597575              n*num = 2142215729            current thread = 915
                num = 455277577              n*num = -1108024257             current thread = 453
                num = 748744868              n*num = 946246780           current thread = 745
                num = 391961611              n*num = -1551236019             current thread = 390
                num = 391961661              n*num = -1551235669             current thread = 390
                num = 917589155              n*num = 2128156789            current thread = 913
                num = 88443344              n*num = 619103408           current thread = 88
                num = 666333333              n*num = 369366035           current thread = 663
                num = 666333336              n*num = 369366056           current thread = 663
                num = 666333363              n*num = 369366245           current thread = 663
                num = 666333366              n*num = 369366266           current thread = 663
                num = 666333367              n*num = 369366273           current thread = 663
                num = 831155822              n*num = 1523123458            current thread = 827
                num = 67337673              n*num = 471363711           current thread = 67
                num = 67337676              n*num = 471363732           current thread = 67
                num = 67337677              n*num = 471363739           current thread = 67
                num = 67337683              n*num = 471363781           current thread = 67
                num = 67337797              n*num = 471364579           current thread = 67
                num = 67337967              n*num = 471365769           current thread = 67
                num = 161811116              n*num = 1132677812             current thread = 161
                num = 981911819              n*num = -1716551859             current thread = 977
                num = 177891117              n*num = 1245237819            current thread = 177
                num = 177891127              n*num = 1245237889            current thread = 177
                num = 177891128              n*num = 1245237896            current thread = 177
                num = 511559591              n*num = -714050159            current thread = 509
                num = 437186174              n*num = -1234664078             current thread = 435
```

...

(продолжение теста см. на следующий странице)

...



Примечание к тесту № 5: (n*num) является отрицательным из-за переполнения типа данных int.

4. Текст программы

```
#include <iostream>
#include "omp.h"
#include <iomanip>
#include <string>
#include <chrono>
#include <mutex>

/*
	Минец Максим
	БПИ-193

	Вариант 15
	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.

*/

// Количество чисел от первого с 4 значащами цифрами по максимальное из 9 цифр.
constexpr int VALUES_AMOUNT = 999999000;
// lowerBound - минимальное четырехзначное число
// upperBound - максимальное девятизначное число
constexpr int lowerBound = 1000, upperBound = 999999999;
std::mutex mut;

/// <summary>
/// Пользователь вводит число n, на которое умножаются числа.
/// </summary>
/// <returns> Возвращает число n, на которое умножаются числа </returns>
int input_n()
{
	int num;

	do
	{
		std::cin >> num;
		if (num <= 1 || num >= 10)
			std::cout << "Incorrect input. Try again (number should be > 0 and <
10): ";
	} while (num <= 1 || num >= 10);

	return num;
}

/// <summary>
/// Пользователь вводит число потоков.
/// </summary>
/// <returns> Возвращает число потоков </returns>
int inputnumerOfThreads()
{
	int num;
	// Поставлена верхняя граница для количества потоков (по усмотрению разработчика):
100.
	do
	{
		std::cin >> num;
		if (num < 1 || num > 1000)
			std::cout << "Incorrect input. Try again (number should be > 0 and <
1000): ";
	} while (num < 1 || num > 1000);
```

```
	return num;
}

void process(int n, int t) {
	// Устанавливаем количество потоков.
	omp_set_num_threads(t);
#pragma omp parallel for num_threads(t)
	for (int i = lowerBound; i <= upperBound; ++i) {
		// Текущая цифра.
		std::string number = std::to_string(i);
		// Текущая цифра, умноженная на n.
		std::string result = std::to_string(i * n);
		// Флаг, показывающий, соблюдает ли новое число необходимым условиям.
		bool flag = true;

		for (char num : number) {
			unsigned int lhs = 0;
			unsigned int rhs = result.length() - 1;
			while (lhs < rhs - 1) {
				unsigned int mid = lhs + (rhs - lhs) / 2;
				result[mid] < num ? lhs = mid : rhs = mid;
			}
			if (!(result[lhs] == num || result[rhs] == num)) {
				flag = false;
				break;
			}
		}

		// Вывод подходящих нам чисел.
		if (flag) {
			mut.lock();
			std::cout << std::setfill(' ') << std::setw(25) << "num = " << number
				<< std::setfill(' ') << std::setw(25) << "n*num = " << result
				<< std::setfill(' ') << std::setw(30) << "current thread = "
				<< std::to_string(omp_get_thread_num()) << "\n";
			mut.unlock();
		}
	}
}

int main(int argc, char* argv[]) {
	std::cout << "Input n: ";
	int n = input_n();

	std::cout << "Input number of threads: ";
	int threads = inputnumerOfThreads();

	std::cout << "\nProgram has started its work.\n\n";

	process(n, threads);

	std::cout << "\nProgram has finished its work.\n";

	return 0;
}
```

5. Список использованной литературы

1. Википедия. OpenMP [Электронный ресурс] -https://ru.wikipedia.org/wiki/OpenMP

2. Грегори Р. Эндрюс. Основы многопоточного, параллельного и распределенного программирования. - М.: Издательский дом "Вильямс", 2003.

3. Легалов А. И. Учебно-методические материалы [Электронный ресурс] - http://softcraft.ru/edu/comparch/