Правительство Российской Федерации

Национальный исследовательский университет

«Высшая школа экономики»

Факультет компьютерных наук

Департамент программной инженерии

Домашнее задание 3

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 потоки и собрать все потоки во избежание лишней потери памяти. Весь массив равномерно распределяется между всеми потоками. Применяется модель итеративного параллелизма.

Итеративный параллелизм - процессы выполняют циклические вычисления, решая одну задачу (итерации одного цикла).

2.3 Формат ввода данных

Ввод входных данных осуществлен через консоль (не через командную строку). Для запуска необходимо ввести в консоль два аргумента: число потоков и число n, больше единицы и меньше десяти. Пример входных данных продемонстрирован на рис. 1.

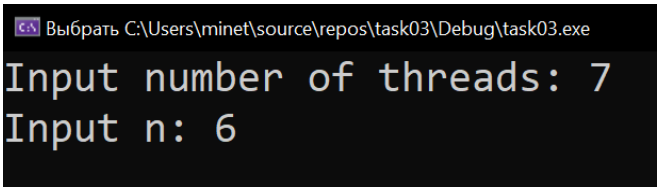


рис. 1

2.4 Формат выводы данных

Результат программы – числа, которые при умножении на число n (лежащее в промежутке от 1 до 10), содержит все числа, лежащие в промежутке от 1000 до 999999999. Каждое число выводится с новой строки. Пример выходных данных продемонстрирован на рис. 2 и рис. 3.

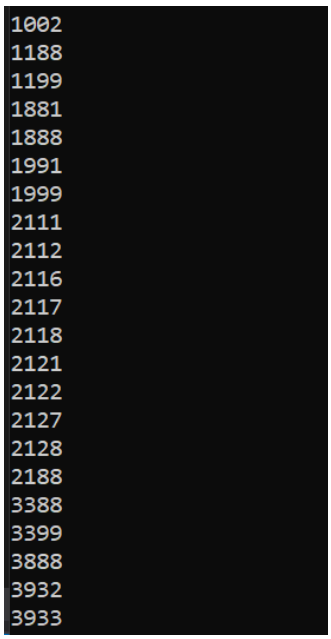


рис. 2

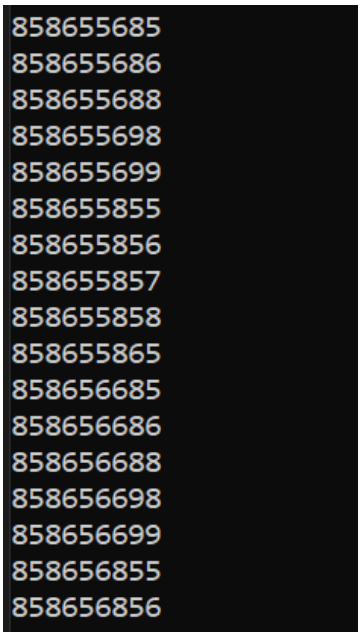


рис. 3

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

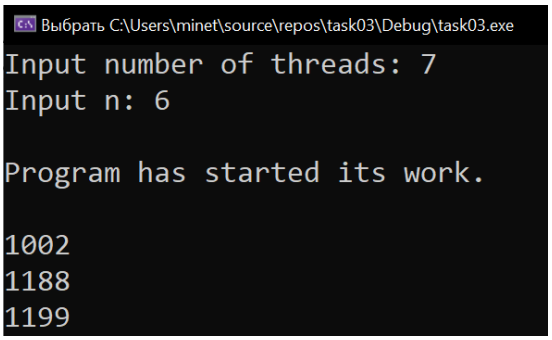
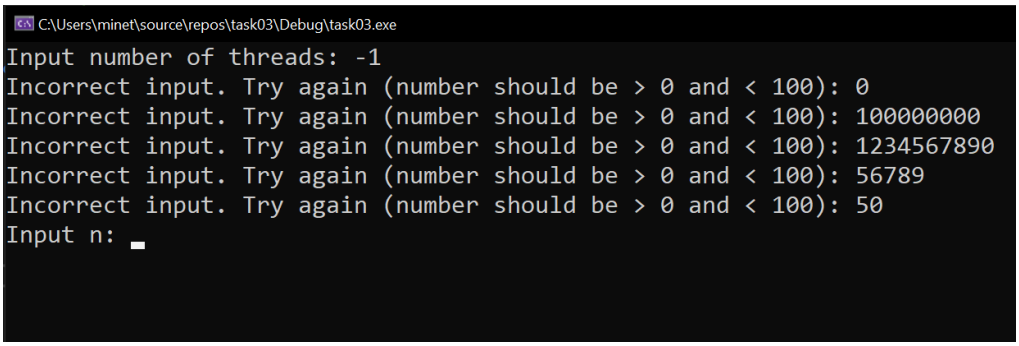


рис. 4

3. Тестирование программы

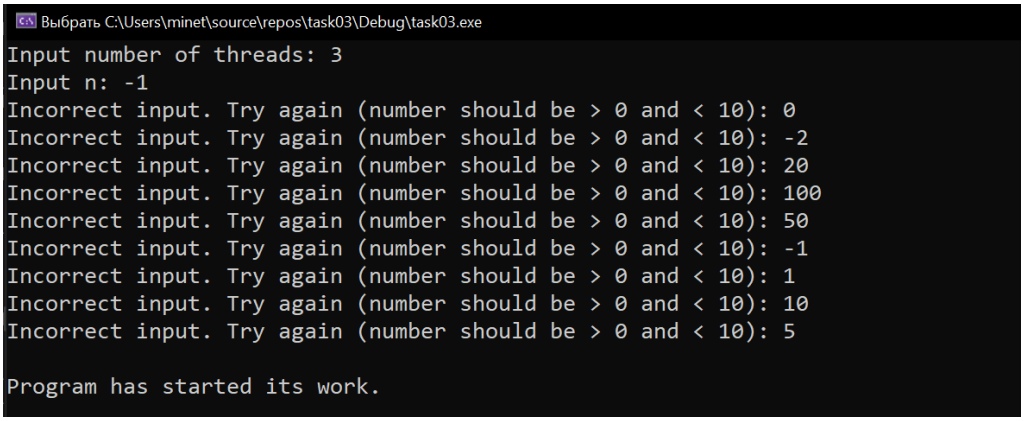
3.1 Тест № 1

При некорректном вводе количества потоков, программа предупреждает о некорректном вводе и просит пользователя ввести число еще раз. Максимальным числом потоков взято число 100 (на усмотрение разработчика).



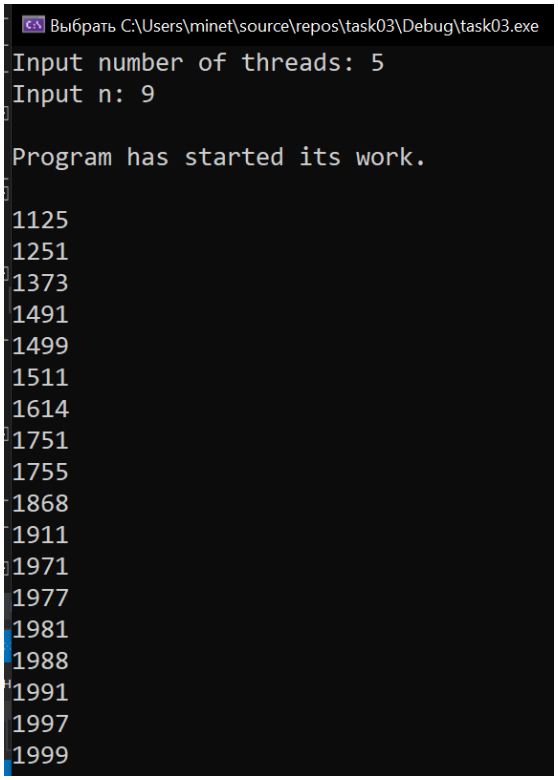
3.2 Тест № 2

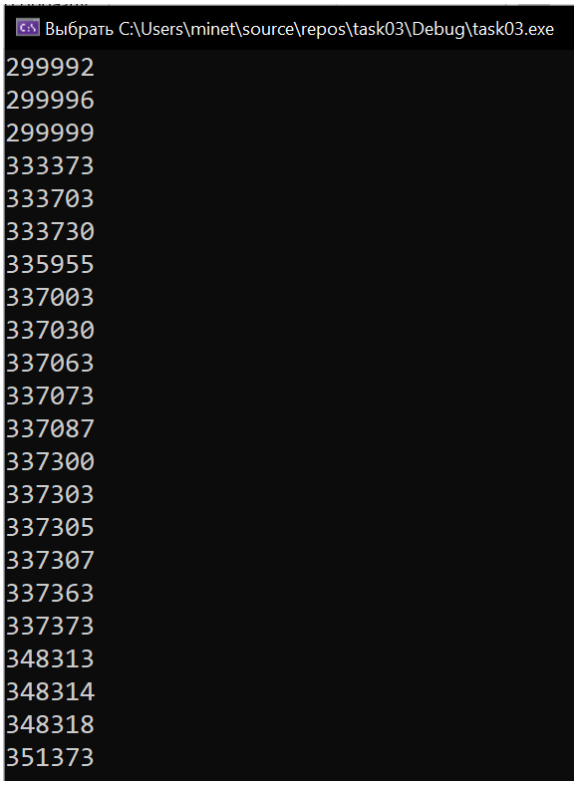
При некорректном вводе числа n, программа предупреждает о некорректном вводе и просит пользователя ввести число еще раз. Пограничными значениями являются числа 1 и 10 (по условию задачи).



3.3 Тест № 3

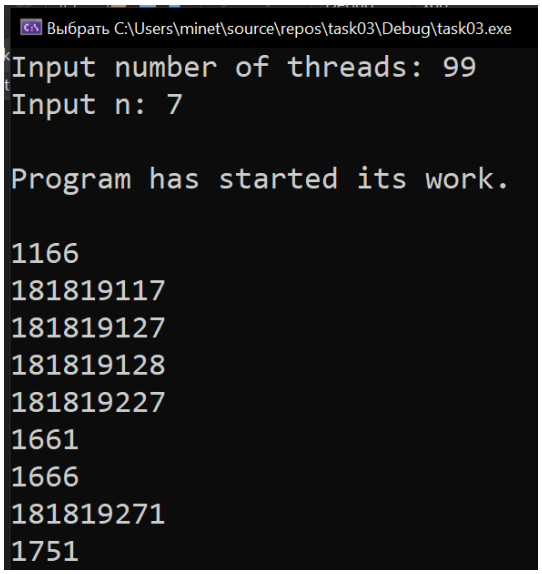
Работа программы при корректных данных. Число потоков равно 5, число n = 9.

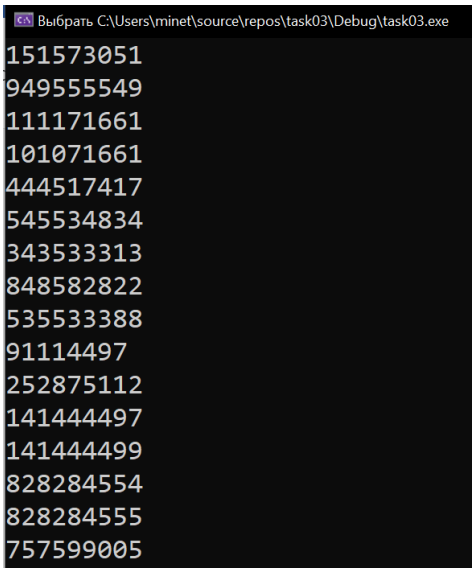




3.4 Тест № 4

Работа программы при корректных данных. Число потоков равно 99, число n = 7.





4. Текст программы

```
#include <iostream>
#include <cstdlib>
#include <thread>
#include <string>
#include <stdio.h>
#include <vector>

const int values_amount = 999999000; // Количество чисел от первого с 4 значащами цифрами
по максимальное из 9 цифр.
int n, threads_amount; // Число n (из условия) и количество тредов.
std::vector<int> vec;

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

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

	return num;
}

/// <summary>
/// Пользователь вводит число потоков.
/// </summary>
/// <returns> Возвращает число потоков </returns>
int inputnumerOfThreads()
{
	int num;
	std::cin >> num;

	while (num <= 0 || num >= 100) // Поставлена верхняя граница для количества
потоков (по усмотрению): 100.
	{
		std::cout << "Incorrect input. Try again (number should be > 0 and < 100):
";
		std::cin >> num;
	}

	return num;
}

/// <summary>
/// Метод, осуществляющий бинарный поиск.
/// </summary>
/// <param name="i"> Индекс </param>
/// <param name="num"> Число </param>
/// <returns> Возвращает переменную типа bool. </returns>
bool binSearch(int i, std::string num)
{
	if (num.length() == 0)
		return false;
```

```
	int lhs = 0;
	int rhs = num.length() - 1;
	while (lhs < rhs - 1)
	{
		int m = lhs + (rhs - lhs) / 2;

		if (num[m] < i)
			lhs = m;
		else
			rhs = m;
	}

	return num[lhs] == i || num[rhs] == i;
}

bool compare(std::string modified, std::string original) {

	for (int i = 0; i < original.length(); ++i) {
		if (!binSearch(original[i], modified)) {
			return false;
		}
	}

	return true;
}

void stream(int index) {
	int amount = values_amount / threads_amount;  // Длина каждого подотрезка из
отрезка длиной в values_amount.

	int lower = 1000 + amount * (index - 1); // Нижняя граница для текущего треда.
	int upper; // Верхняя граница для текущего треда.

	if (index == 10)
		upper = amount* index;
	else
		upper = 999 + amount * index;

	for (int i = lower; i <= upper; ++i) {
		if (compare(std::to_string(i * n), std::to_string(i))) {
			std::cout << "\n" << i;
		}
	}
}

/// <summary>
/// Метод, реализующий создание необходимого количество потоков.
/// </summary>
void makeThreads() {
	std::vector<std::thread> ts;

	std::cout << "\nProgram has started its work.\n"; // Показываем в консоли, что
программа начала свою работу.

	for (int i = 1; i <= threads_amount; ++i) {
		ts.push_back(std::thread(stream, i));
	}

	for (int i = 0; i < threads_amount; ++i) {
		ts[i].join();
	}
}
```

```
int main(int argc, char* argv[]) {
	std::cout << "Input number of threads: ";
	threads_amount = inputnumerOfThreads(); // Количество потоков.
	std::cout << "Input n: ";
	n = input_n(); // Число n.

	makeThreads();

	return 0;
}
```

5. Список использованной литературы

1. Уильямс Энтони. C++. Практика многопоточного программирования. — СПб.: Питер, 2020. — 640 с.

2. Богачёв К. Ю. Основы параллельного программирования: учебное пособие / К. Ю. Богачёв. — 4-е изд., электрон. — М.: Лаборатория знаний, 2020. — 345 с.

3. Парадигмы параллельного программирования [Электронный ресурс] - http://staff.mmcs.sfedu.ru/~dubrov/files/sl_parallel_05_paradigm.pdf

4. Википедия. Многопоточность [Электронный ресурс] - https://ru.wikipedia.org/wiki/Многопоточность

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