

Пояснительная записка к задаче №3 по АВС, вариант 15

Михайлова Ксения Дмитриевна, БПИ192

1 модуль, 2020

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.

Интерпретация:

Пройти числа с 1000 по 999999999, если после умножения на $1 < n < 9$ число $q = i \cdot n$ содержит все цифры, из которых состоит $i \in [1000, 999999999]$, вывести число q в консоль.

Решение:

Прочитать аргументы, запустить фабрику потоков, собрать потоки во избежание утечки памяти. Весь массив чисел распределить равномерно между потоками. К этой задаче применима модель **итеративного параллелизма**.

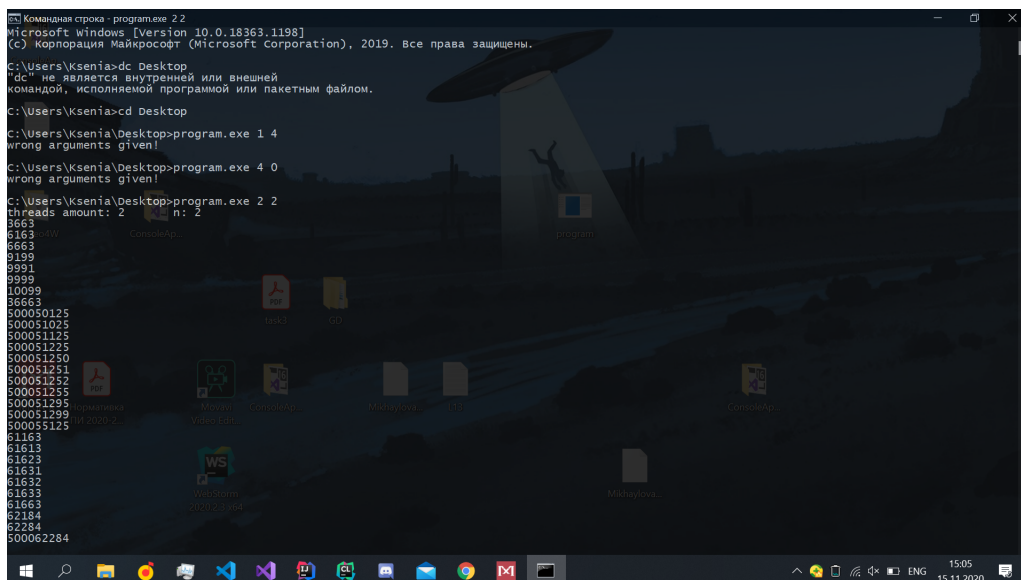
Эксплуатация:

Чтобы запустить программу необходимо ввести ее название в консоли и дать два параметра: число n и количество потоков. Например: «program.exe 2 10».

Примеры работы:

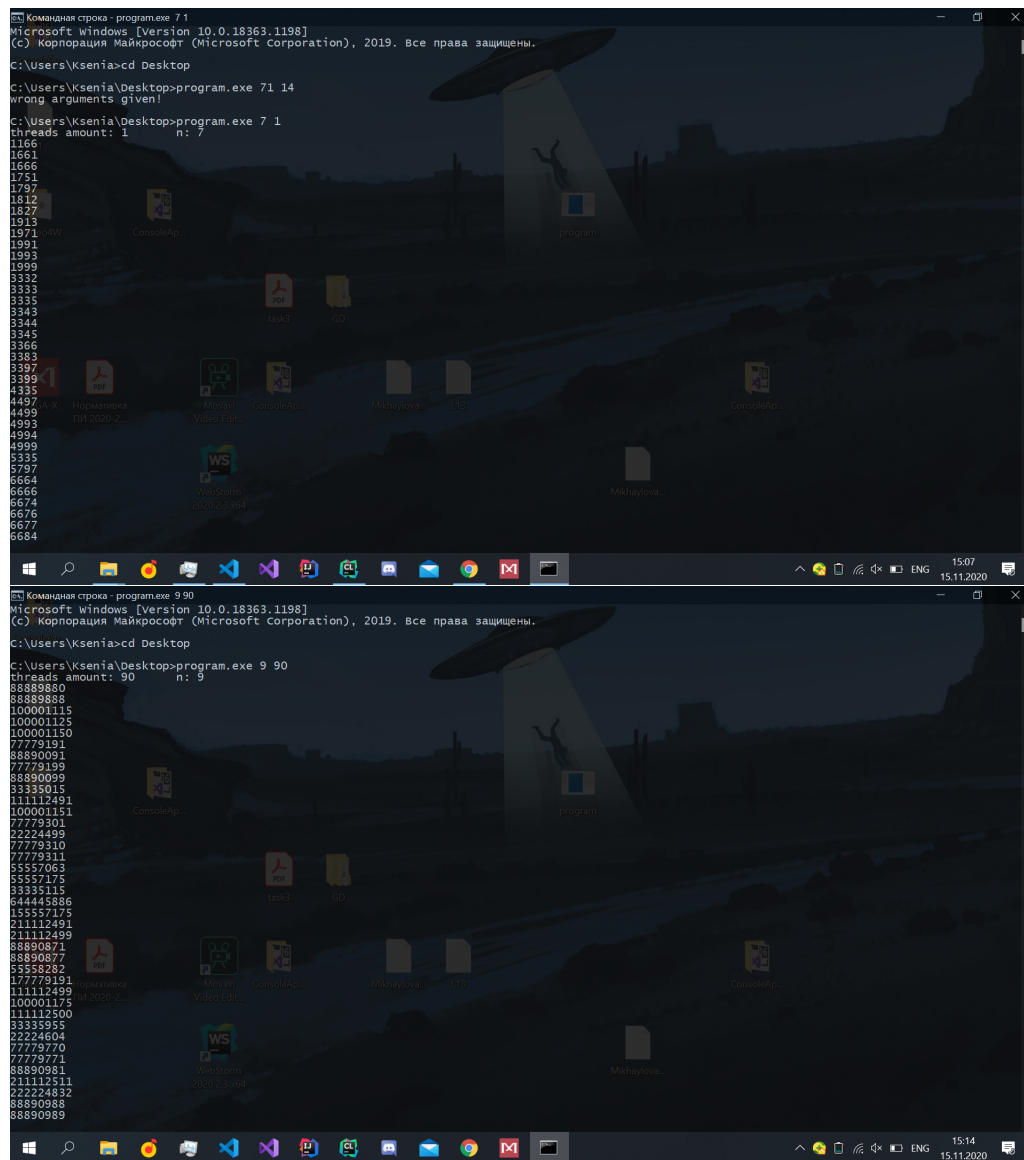
Проведены тесты на:

Некорректный ввод числа n , количества потоков (по отдельности и совместно). Работу программы при корректных входных данных.



```
Командная строка - program.exe 2.2
Microsoft Windows [Version 10.0.18363.1198]
(c) Корпорация Майкрософт (Microsoft Corporation), 2019. Все права защищены.

C:\Users\Ksenia>cd Desktop
C:\Users\Ksenia\Desktop>program.exe 1 4
wrong arguments given!
C:\Users\Ksenia\Desktop>program.exe 4 0
wrong arguments given!
C:\Users\Ksenia\Desktop>program.exe 2 2
threads amount: 2 n: 2
3663
6163
6663
9199
9991
9999
10099
3663
500050125
500051025
500051125
500051225
500051250
500051251
500051252
500051255
500051295
500051299
50005125
61163
61613
61623
61631
61632
61633
61663
62184
62284
500062284
```



Текст программы:

См. TextProgrammy.cpp

```

1  #include <iostream>
2  #include <thread>
3  #include <string>
4  #include <unordered_map>
5  #include <vector>
6
7  using namespace std;
8
9  const int values_amount = 999999000;
10 int n, threads_amount;
11 vector<thread> threads;
12
13 void work(int thread_index) {
14     int amount = values_amount / threads_amount, lower_bound = 1000 + amount * (thread_index - 1),
15     upper_bound = (thread_index == 10 ? 0 : 999) + amount * thread_index;
16
17     for (int i = lower_bound; i <= upper_bound; ++i) {
18         bool flag = true;
19         string modified = to_string(_Val: i * n), original = to_string(i);
20
21         for (char c : original) {
22             unsigned int l = 0, r = modified.length() - 1;
23             while (l < r - 1) {
24                 unsigned int m = l + (r - l) / 2;
25                 if (modified[m] < c)
26                     l = m;
27                 else
28                     r = m;
29             }
30             if (!(modified[l] == c || modified[r] == c)) {
31                 flag = false;
32                 break;
33             }
34         }
35
36         if (flag)
37             cout << original.append(_Ptr: "\n");
38     }
39 }
40
41 void make_threads() {
42     for (int i = 1; i <= threads_amount; ++i)
43         threads.emplace_back(thread(work, i));
44
45     for (int i = 0; i < threads_amount; ++i)
46         threads[i].join();
47 }

```

```

48
49 ► int main(int argc, char *argv[]) { /* N, then thread numb */
50     if (argc != 2) {
51         cout << "two params required\n";
52         return -1;
53     }
54
55     n = (int) stol(_Str: argv[0]); /* number to work with */
56     threads_amount = (int) stol(_Str: argv[1]); /* how many threads program's allowed to create */
57
58     if (n < 2 || n > 9 || threads_amount < 1) {
59         cout << "wrong arguments given!\n";
60         return -1;
61     }
62
63     cout << "threads amount: " << threads_amount << "\tn: " << n << '\n';
64
65     make_threads();
66     return 0;
67 }

```