

## Пике Кирилл БПИ197 Вариант 15.

**Текст 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.

Примечание: для сокращения количества получаемых чисел, было принято, что «в произвольном количестве» трактуется как от 1 и более, то есть при факторе умножения  $n=3$  и числе 1037, у нас получается  $1037*3=3111$ , число не подходит, так как нету 0 и 7. Однако даже с таким ограничением, количество получаемых в результате чисел слишком много, чтобы выводить на экран. Потому было решено выводить на экран лишь первые 100 чисел. Целое положительное  $n$  больше 1 и меньше 10 интерпретируется как строгое неравенство

Так как в этом задании я мог пользоваться библиотекой OpenMP, мне не нужно было распараллеливать вычисления вручную, библиотечные функции сами за меня всё сделали. Соответственно задача оставалась за малым, воспользоваться алгоритмом из прошлой итерации задачи для вычисления подхождимости чисел под критерии задания и реализовать выполнение программы, используя OpenMP.

Алгоритм проверки чисел:

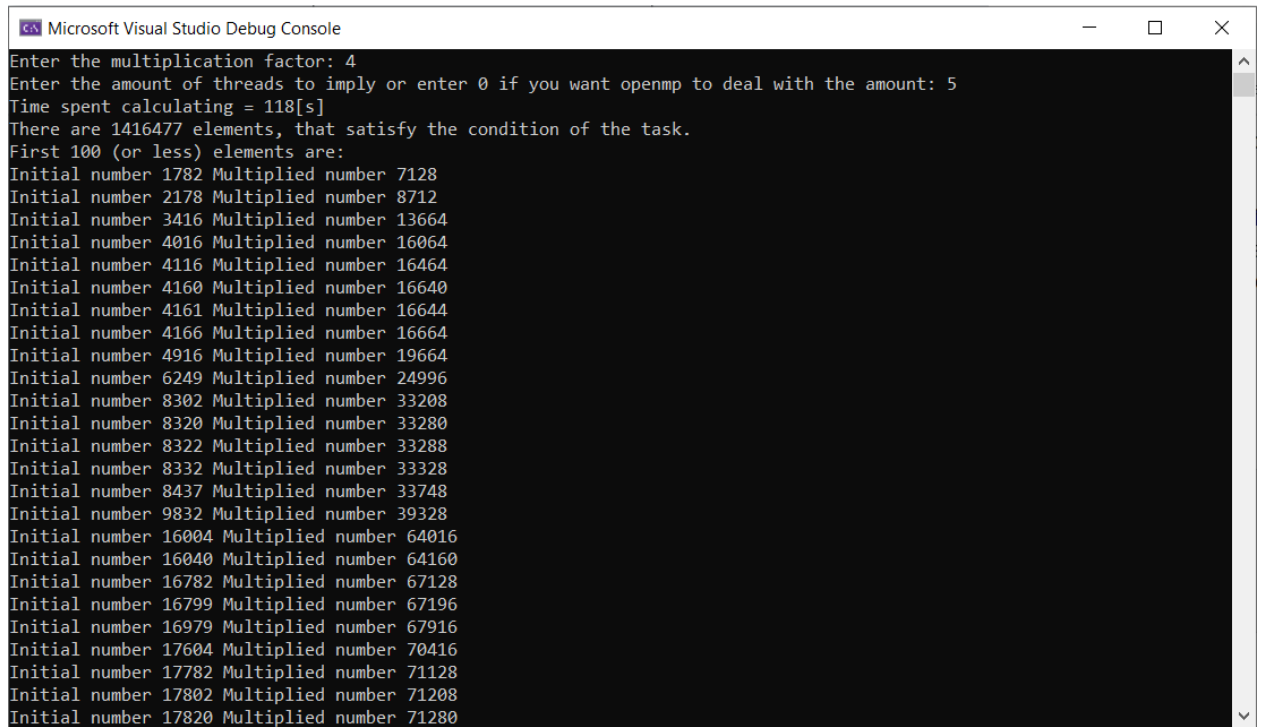
- Создавал 2 массива из 10 нулей (10 возможных цифр).
- Заполнял первый массив единицами по индексу равному цифрам числа и проверял, чтобы по тем же индексам второго числа были строго единицы.
- Попутно с проверкой заполнял второй массив тем же образом для второго числа и проверял, чтобы элементы по индексу цифр первого числа отличались от нуля.
- Если при проверке наткнулся на нулевой элемент, то заканчивал проверку отрицательным ответом.
- Если проверка доходила до конца, то числа выполняли условие задачи.

```
bool check_same_digits(uint64_t a, uint64_t b)
{
    int c[10] = { 0 };
    int d[10] = { 0 };
    uint64_t temp_b = b;
    while (b > 0)
    {
        c[b % 10] = 1;
        b /= 10;
    }
    while (a > 0)
    {
        if (c[a % 10] == 0)
        {
            return false;
        }
        d[a % 10] = 1;
        a /= 10;
    }
    while (temp_b > 0)
    {
        if (d[temp_b % 10] == 0)
        {
            return false;
        }
        temp_b /= 10;
    }
    return true;
}
```

Мы используем OpenMP для того, чтобы разбить выполняемый цикл for на более или менее равные куски. Затем выполняем подсчеты на каждом промежутке и затем в перекладываем их в основной вектор в порядке их появления в изучаемом радиусе чисел.

```
std::vector<uint64_t> vect;
#pragma omp parallel
{
    std::vector<uint64_t> private_vec;
    #pragma omp for nowait schedule(static)
    for (int i = 1000; static_cast<uint64_t>(i) <= 999999999; i++)
    {
        if (check_same_digits(a:static_cast<uint64_t>(i), b:static_cast<uint64_t>(i) * n))
        {
            private_vec.push_back(static_cast<uint64_t>(i));
        }
    }
    #pragma omp for schedule(static) ordered
    for (int i = 0; i < omp_get_num_threads(); i++) {
        #pragma omp ordered
        vect.insert(_Where: vect.end(), _First: private_vec.begin(), _Last: private_vec.end());
    }
}
```

Проверим программу на работоспособность:



```
Microsoft Visual Studio Debug Console
Enter the multiplication factor: 4
Enter the amount of threads to imply or enter 0 if you want openmp to deal with the amount: 5
Time spent calculating = 118[s]
There are 1416477 elements, that satisfy the condition of the task.
First 100 (or less) elements are:
Initial number 1782 Multiplied number 7128
Initial number 2178 Multiplied number 8712
Initial number 3416 Multiplied number 13664
Initial number 4016 Multiplied number 16064
Initial number 4116 Multiplied number 16464
Initial number 4160 Multiplied number 16640
Initial number 4161 Multiplied number 16644
Initial number 4166 Multiplied number 16664
Initial number 4916 Multiplied number 19664
Initial number 6249 Multiplied number 24996
Initial number 8302 Multiplied number 33208
Initial number 8320 Multiplied number 33280
Initial number 8322 Multiplied number 33288
Initial number 8332 Multiplied number 33328
Initial number 8437 Multiplied number 33748
Initial number 9832 Multiplied number 39328
Initial number 16004 Multiplied number 64016
Initial number 16040 Multiplied number 64160
Initial number 16782 Multiplied number 67128
Initial number 16799 Multiplied number 67196
Initial number 16979 Multiplied number 67916
Initial number 17604 Multiplied number 70416
Initial number 17782 Multiplied number 71128
Initial number 17802 Multiplied number 71208
Initial number 17820 Multiplied number 71280
```

```
Microsoft Visual Studio Debug Console
Enter the multiplication factor: 9
Enter the amount of threads to imply or enter 0 if you want openmp to deal with the amount: 0
Time spent calculating = 119[s]
There are 2314507 elements, that satisfy the condition of the task.
First 100 (or less) elements are:
Initial number 1089 Multiplied number 9801
Initial number 1250 Multiplied number 11250
Initial number 1750 Multiplied number 15750
Initial number 2250 Multiplied number 20250
Initial number 2491 Multiplied number 22419
Initial number 2500 Multiplied number 22500
Initial number 2875 Multiplied number 25875
Initial number 3491 Multiplied number 31419
Initial number 3625 Multiplied number 32625
Initial number 3750 Multiplied number 33750
Initial number 3955 Multiplied number 35595
Initial number 3995 Multiplied number 35955
Initial number 4450 Multiplied number 40050
Initial number 4500 Multiplied number 40500
Initial number 4505 Multiplied number 40545
Initial number 4901 Multiplied number 44109
Initial number 4910 Multiplied number 44190
Initial number 4911 Multiplied number 44199
Initial number 4955 Multiplied number 44595
Initial number 4966 Multiplied number 44694
Initial number 4991 Multiplied number 44919
Initial number 4995 Multiplied number 44955
Initial number 4996 Multiplied number 44964
Initial number 5045 Multiplied number 45405
Initial number 5428 Multiplied number 48852
```

Так же есть проверка на верные значения

```
C:\Users\kiril\OneDrive\Рабочий стол\Homework 4\hw3\Debug\hw4.exe
Enter the multiplication factor: 0
Enter the amount of threads to imply or enter 0 if you want openmp to deal with the amount: 6
Wrong values. 1<n<10 and amount of threads involved must be positive.
Enter the multiplication factor: 4
Enter the amount of threads to imply: -1
Wrong values. 1<n<10 and amount of threads involved must be positive.
Enter the multiplication factor:
```