

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

Факультет компьютерных наук Департамент

программной инженерии

**Самостоятельная работа по дисциплине:**

**"Архитектура вычислительных систем"**

**Список целых чисел, содержащих от 4 до 9 значащих цифр, которые  
после умножения на  $n$ , будут содержать все те же самые цифры**

**Исполнитель**

Студент группы БПИ195

Кенесбек Ерасыл

Почта: [ekenesbek@edu.hse.ru](mailto:ekenesbek@edu.hse.ru)

### Задание:

Разработать программу с применением OpenMP и протестировать ее.

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.

### Решение:

Реализуется три метода:

1. `toListInt` – возвращает из задаваемого числа список его цифр (входной параметр – число `long`; возвращаемое значение – `vector<int>`);
2. `res` – проверяет содержат ли новое число все те же цифры, что первое число (входные параметры – список цифр начального числа и нового `vector<int>`; возвращаемое значение – `bool`);
3. `MainMethod(int start, int end, int n)` – возвращает строку из чисел, которые соответствуют условию (входные параметры – числа диапазон работы потока `int` и число  $n$  в разы которого изменится новое число `int`; возвращаемое значение – `string`).
4. `MainMethod` – реализует многопоточность, для этого в главном методе, т.е. в `main` задали количество потоков следующим образом: `omp_set_num_threads(numthreads)` – где `numthreads` является количеством потоков. Место, где нужно распараллелить выделяем под `#pragma omp parallel`, используем `#pragma omp for` для выполнения цикла `for` и `#pragma omp critical` для того, чтобы код выполняется только в одном потоке за раз.

## Код программы:

```
#include <iostream>
#include <math.h>
#include <string>
#include <vector>
#include <sstream>
#include <thread>
#include <chrono>
#include <omp.h>
#include <algorithm>

using namespace std;

/// <summary>
/// Метод для перевода числа в Вектор интов
/// </summary>
/// <param число="number"></param>
/// <returns></returns>
static vector<int> toListInt(long number)
{
    bool check = false;
    vector<int> el;
    el.resize(1);
    el.at(0) = number % 10;
    int count = 0;
    while (number > 0)
    {
        for (size_t i = 0; i < el.size(); i++)
        {
            if (number % 10 != el[i])
                check = true;
        }
        if (check == true)
        {
            el.resize(el.size() + 1);
            el.at(el.size()-1) = number % 10;
            check = false;
        }
        number /= 10;
    }
    return el;
}

/// <summary>
/// Метод для сравнения начального числа с числом * n
/// </summary>
/// <param вектор первого числа="as"></param>
/// <param вектор первого числа * на n="newas"></param>
/// <returns></returns>
static bool Res(vector<int> as, vector<int> newas)
{

```

```

bool check = false;
for (size_t i = 0; i < as.size(); i++)
{
    for (size_t j = 0; j < newas.size(); j++)
    {
        if (as[i] == newas[j])
        {
            check = true;
            break;
        }
    }
    if (check == false)
    {
        return false;
    }
    if (i != as.size() - 1) {
        check = false;
    }
}
return true;
}
/// <summary>
/// Метод, который возвращает подходящий ответ в виде вектора числа
/// </summary>
/// <param число вводимое пользователем="n"></param>
/// <param результат, который выводится по ссылке="res"></param>
static void MainMethod(int n, vector<int>* res)
{
#pragma omp parallel
{
#pragma omp for
    for (int i = 1000; i < 1000000000; i++)//вы можете изменить верхнюю границу
здесь для быстрой проверки
    {
#pragma omp critical
        {
            //cout << "Thread #" << omp_get_thread_num() << endl;
            if (Res(toListInt(i), toListInt(i * n)))
            {
                res->resize(res->size() + 1);
                res->at(res->size()-1) = i;
            }
        }
    }
}
}
int main()
{
    try
    {
        int n, numthreads;
        do
        {
            cout << "Enter the number of threads" << endl;
            cin >> numthreads;
        } while (numthreads < 1);
        do
        {
            cout << "Enter a number from 1 to 9" << endl;
            cin >> n;
        } while (n < 1 || n > 9);
        cout << "The program is running. Please wait..." << endl;
        vector<int> result;
        omp_set_num_threads(numthreads);
    }
}

```

```

        MainMethod(n, &result);

        system("Color A");
        sort(result.begin(), result.end());
        for (size_t i = 0; i < result.size(); i++)
        {

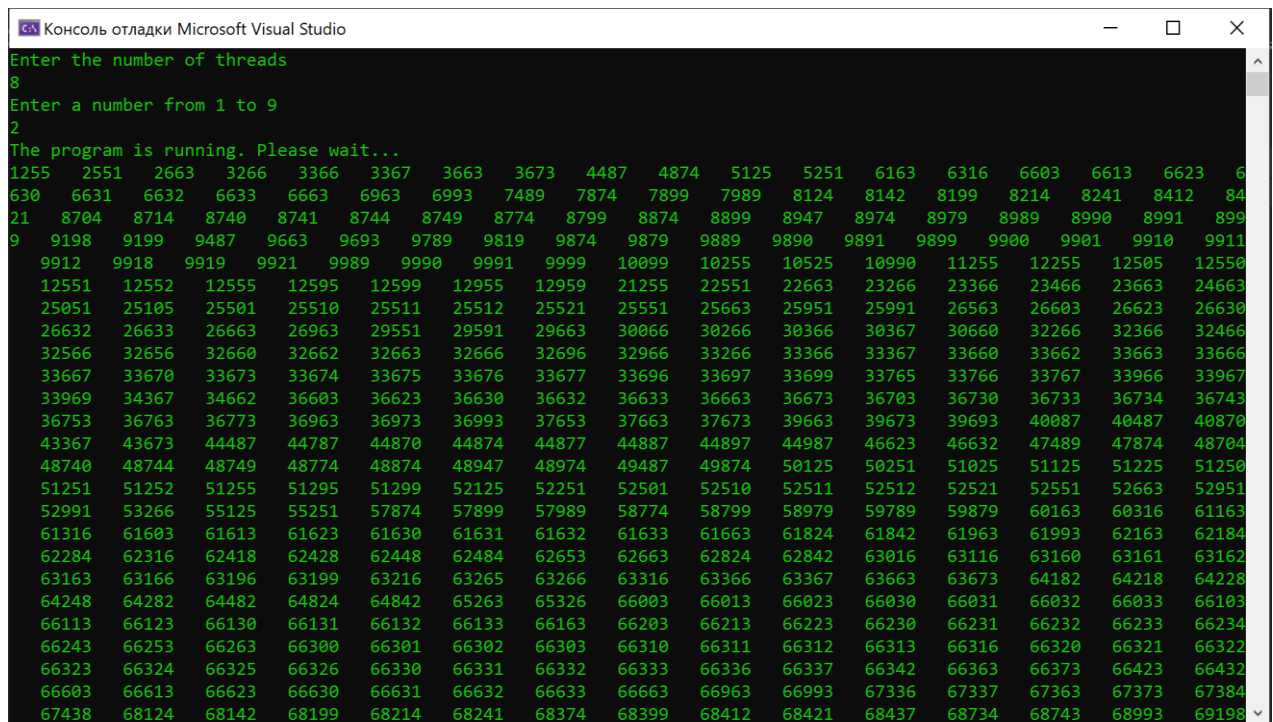
            cout << result[i] << " ";
        }
        cout << endl << "Program ended seccessufully!";
    }
    catch (exception e)
    {
        cout << e.what() << endl;
    }
}

```

## Выполнение задания:

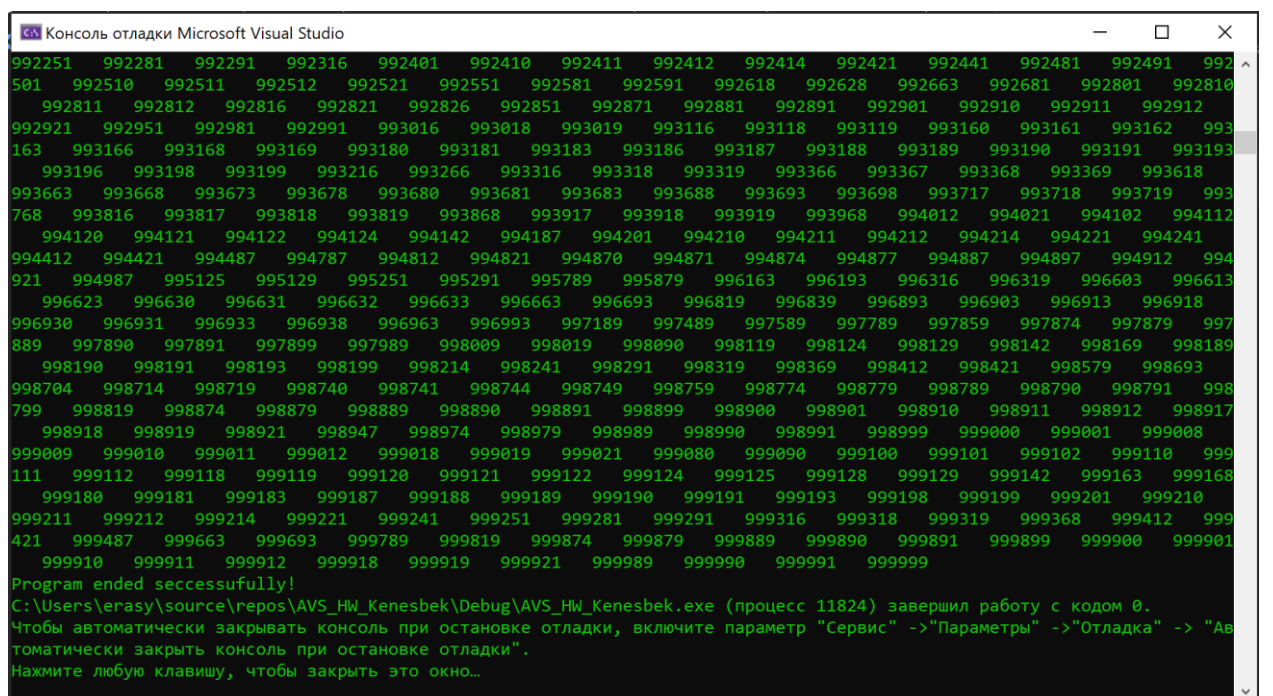
Верхняя граница была ограничена с 1 000 000.

(выполнение см. рис. 1 и рис. 2)



```
Консоль отладки Microsoft Visual Studio
Enter the number of threads
8
Enter a number from 1 to 9
2
The program is running. Please wait...
1255 2551 2663 3266 3366 3367 3663 3673 4487 4874 5125 5251 6163 6316 6603 6613 6623 6
630 6631 6632 6633 6663 6963 6993 7489 7874 7899 7989 8124 8142 8199 8214 8241 8412 84
21 8704 8714 8740 8741 8744 8749 8774 8799 8874 8899 8947 8974 8979 8989 8990 8991 899
9 9198 9199 9487 9663 9693 9789 9819 9874 9879 9889 9890 9891 9899 9900 9901 9910 9911
9912 9918 9919 9921 9989 9990 9991 9999 10099 10255 10525 10990 11255 12255 12505 12550
12551 12552 12555 12595 12599 12955 12959 21255 22551 22663 23266 23366 23466 23663 24663
25051 25105 25501 25510 25511 25512 25521 25551 25663 25951 25991 26563 26603 26623 26630
26632 26633 26663 26963 29551 29591 29663 30066 30266 30366 30367 30660 32266 32366 32466
32566 32656 32660 32662 32663 32666 32696 32966 33266 33366 33367 33660 33662 33663 33666
33667 33670 33673 33674 33675 33676 33677 33696 33697 33699 33765 33766 33767 33966 33967
33969 34367 34662 36603 36623 36630 36632 36633 36663 36673 36703 36730 36733 36734 36743
36753 36763 36773 36963 36973 36993 37653 37663 37673 39663 39673 39693 40087 40487 40870
43367 43673 44487 44787 44870 44874 44877 44887 44897 44987 46623 46632 47489 47874 48704
48740 48744 48749 48774 48874 48947 48974 49487 49874 50125 50251 51025 51125 51225 51250
51251 51252 51255 51295 51299 52125 52251 52501 52510 52511 52512 52521 52551 52663 52951
52991 53266 55125 55251 57874 57899 57989 58774 58799 58979 59789 60163 60316 61163 61163
61316 61603 61613 61623 61630 61631 61632 61633 61663 61824 61842 61963 61993 62163 62184
62284 62316 62418 62428 62448 62484 62653 62663 62824 62842 63016 63116 63160 63161 63162
63163 63166 63196 63199 63216 63265 63266 63316 63366 63367 63663 63673 64182 64218 64228
64248 64282 64482 64824 64842 65263 65326 66003 66013 66023 66030 66031 66032 66033 66103
66113 66123 66130 66131 66132 66133 66163 66203 66213 66223 66230 66231 66232 66233 66234
66243 66253 66263 66300 66301 66302 66303 66310 66311 66312 66313 66316 66320 66321 66322
66323 66324 66325 66326 66330 66331 66332 66333 66336 66337 66342 66363 66373 66423 66432
66603 66613 66623 66630 66631 66632 66633 66663 66963 66993 67336 67337 67363 67373 67384
67438 68124 68142 68199 68214 68241 68374 68399 68412 68421 68437 68734 68743 68993 69198
```

Рисунок 1. Начало программы в консоли



```
Консоль отладки Microsoft Visual Studio
992251 992281 992291 992316 992401 992410 992411 992412 992414 992421 992441 992481 992491 992
501 992510 992511 992512 992521 992551 992581 992591 992618 992628 992663 992681 992801 992810
992811 992812 992816 992821 992826 992851 992871 992881 992891 992901 992910 992911 992912
992921 992951 992981 992991 993016 993018 993019 993116 993118 993119 993160 993161 993162 993
163 993166 993168 993169 993180 993181 993183 993186 993187 993188 993189 993190 993191 993193
993196 993198 993199 993216 993266 993316 993318 993319 993366 993367 993368 993369 993618
993663 993668 993673 993678 993680 993681 993683 993688 993693 993698 993717 993718 993719 993
768 993816 993817 993818 993819 993868 993917 993918 993919 993968 994012 994021 994102 994112
994120 994121 994122 994124 994142 994187 994201 994210 994211 994212 994214 994221 994241
994412 994421 994487 994787 994812 994821 994870 994871 994874 994877 994887 994897 994912 994
921 994987 995125 995129 995251 995291 995789 995879 996163 996193 996316 996319 996603 996613
996623 996630 996631 996632 996633 996663 996693 996819 996839 996893 996903 996913 996918
996930 996931 996933 996938 996963 996993 997189 997489 997589 997789 997859 997874 997879 997
889 997890 997891 997899 997989 998009 998019 998090 998119 998124 998129 998142 998169 998189
998190 998191 998193 998199 998214 998241 998291 998319 998369 998412 998421 998579 998693
998704 998714 998719 998740 998741 998744 998749 998759 998774 998779 998789 998790 998791 998
799 998819 998874 998879 998889 998890 998891 998899 998900 998901 998910 998911 998912 998917
998918 998919 998921 998947 998974 998979 998989 998990 998991 998999 999000 999001 999008
999009 999010 999011 999012 999018 999019 999021 999080 999090 999100 999101 999102 999110 999
111 999112 999118 999119 999120 999121 999122 999124 999125 999128 999129 999142 999163 999168
999180 999181 999183 999187 999188 999189 999190 999191 999193 999198 999199 999201 999210
999211 999212 999214 999221 999241 999251 999281 999291 999316 999318 999319 999368 999412 999
421 999487 999663 999693 999789 999819 999874 999879 999889 999890 999891 999899 999900 999901
999910 999911 999912 999918 999919 999921 999989 999990 999991 999999
Program ended successfully!
C:\Users\erasy\source\repos\AVS_HW_Kenesbek\Debug\AVS_HW_Kenesbek.exe (процесс 11824) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" ->"Ав
томатически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

Рисунок 2. Конец программы в консоли