ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ 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

Москва 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.

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`).

Сохраняем ответ в строку с помощью MainMethod и выводим ответ.

Потоки делились относительно количества потоков. Программа выполнялась относительно значении чисел. Например трехзначное и четырехзначное число обрабатывается в разных потоках одновременно.

Код программы:

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


using namespace std;

static vector<int> toListInt(long number)
{
	bool check = false;
	vector<int> el;
	el.push_back(number % 10);

	while (number > 0)
	{
		for (size_t i = 0; i < el.size(); i++)
		{
			if (number % 10 != el[i])
				check = true;
		}
		if (check == true) {
			el.push_back(number % 10);
			check = false;
		}
		number /= 10;
	}
	return el;
}
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;
}
```

```
static string MainMethod(int start, int end,int n)
{
	string res = "";
	for (size_t i = start; i < end; i++)
	{
		if (Res(toListInt(i), toListInt(i * n)))
		{
			res+=to_string(i);
			if (i != end - 1)
			{
				res += "\n";
			}
		}
	}
	return res;
}
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);
		int step = (pow(10, 9) - pow(10, 3)) / numthreads;
		int start = 1000;
		int remainder = (int)(pow(10, 9) - pow(10, 3)) % numthreads;
		vector<thread> threads(numthreads);
		vector<string> result;
		for (size_t i = 0; i < numthreads; i++)
		{
			if (i != numthreads - 1)
			{
				threads.at(i) = thread([&result, i, n, start, step]()
{result.at(i) = MainMethod(start, start + step, n); });
			}
			else
			{
				threads.at(i) = thread([&result, i, n, start, step,
remainder]() {result.at(i) = MainMethod(start, start + step+remainder, n); });
			}
			start += step;
		}
		for (size_t i = 0; i < numthreads; i++)
		{
			threads[i].join();
			cout << result[i] << endl;
		}
	}
	catch (exception e)
	{
		cout << e.what() << endl;
	}
}
```