ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

Домашнее задание №4
по дисциплине «Архитектура вычислительных систем»

Разработка многопоточных приложений с использованием OpenMP

Исполнитель

студент группы БПИ196-1

Махнач Ф. О.

29.11.2020 г.

Оглавление

1. Текст задания

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 $1 < n < 10$. Количество потоков является входным параметром

2. Подготовка

Алгоритм решения задачи представлен и описан в домашнем задании №3, которое можно найти в репозитории GitHub: https://github.com/FMakhnach/ARCT_task3.

Для решения этого задания будет использоваться тот же алгоритм решения и, по большей части, тот же программный код. В частности, используется **итеративный параллелизм** в качестве модели построения многопоточного приложения. Основной задачей является использование механизмов OpenMP вместо механизмов стандартной библиотеки `<thread>` языка C++. В частности, понадобятся директивы OMP:

```
#pragma omp parallel for num_threads(nthreads)
```

для распараллеливания цикла for, в котором проверяется на заданное условие каждое целое число из диапазона [1e4; 1e9]. Эта конструкция заменяет использование массива `thread`'ов.

```
#pragma omp critical
```

для выделения критической секции: добавление новых элементов в множество ответов. Эта конструкция заменяет использование мьютексов.

3. Изменения в программе

Программа создавалась на основе программы, реализованной для домашнего задания №3. Ниже перечислены блоки кода, которые подверглись изменениям:

1) Директивы `#include <mutex>`, `#include <thread>` заменены на `#include <omp.h>`

2) Из функции `CalculateMultiThread` удалён код, создающий и запускающий потоки (объекты типа `std::thread`). Функция `ProcessNums` удалена, цикл перенесён в `CalculateMultiThread`. с изменением диапазона цикла и добавлением директивы `#pragma omp parallel for`.

Было:

```
// Checking every k * i + thread_num value (this way every thread will calculate its
part).
void ProcessNums(int thread_num) {
	for (int i = glob::LOWER_BOUND + thread_num; i < glob::UPPER_BOUND; i +=
glob::num_of_threads) {
		CheckNum(i);
	}
}

// Launches the threads.
void CalculateMultiThread() {
	std::thread* threads = new std::thread[glob::num_of_threads];

	// Launching the threads.
	for (size_t i = 0; i < glob::num_of_threads; ++i) {
		threads[i] = std::thread(ProcessNums, i);
	}
	// Waiting until each of them will finish.
	for (size_t i = 0; i < glob::num_of_threads; ++i) {
		threads[i].join();
	}

	delete[] threads;
}
```

Стало:

```
void CalculateMultiThread() {
#pragma omp parallel for num_threads(glob::num_of_threads)
	for (int i = glob::LOWER_BOUND; i < glob::UPPER_BOUND; ++i) {
		CheckNum(i);
	}
}
```

3) Использование `std::mutex` в функции `CheckNum` заменено на директиву `#pragma omp critical`.

Было:

```
if (num_map == prod_map) {
	glob::result_set_mutex.lock();
	// If num satisfies the condition, then -num does too
	glob::result_set.insert(-num);
	glob::result_set.insert(num);
	glob::result_set_mutex.unlock();
}
```

Стало:

```
	if (num_map == prod_map) {
#pragma omp critical
		{
			// If num satisfies the condition, then -num does too
			glob::result_set.insert(-num);
			glob::result_set.insert(num);
		}
	}
```

4. Тестирование программы

Результаты вывода программы при различных входных параметрах приведены в папке tests репозитория с работой. Название файла output_N.txt обозначает, что это вывод программы при входных параметрах n = N и кол-во потоков = 4 (максимальное на моём компьютере), например, output_3.txt.

Также протестированы следующие случаи некорректного ввода:

1) n = 0

```
Please, enter N: 0
Wrong input! n must be a integer in range [2; 9].

C:\Users\fedya\source\repos\CPP_Fun\Debug\CPP_Fun.exe (process 7712) exited with code 1.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

2) n = 10

```
Please, enter N: 10
Wrong input! n must be a integer in range [2; 9].

C:\Users\fedya\source\repos\CPP_Fun\Debug\CPP_Fun.exe (process 19328) exited with code 1.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

3) n = 1

```
Please, enter N: 1
Wrong input! n must be a integer in range [2; 9].

C:\Users\fedya\source\repos\CPP_Fun\Debug\CPP_Fun.exe (process 19352) exited with code 1.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

4) n = -1

```
Please, enter N: -1
Wrong input! n must be a integer in range [2; 9].

C:\Users\fedya\source\repos\CPP_Fun\Debug\CPP_Fun.exe (process 7816) exited with code 1.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

5) n = -100

```
Please, enter N: -100
Wrong input! n must be a integer in range [2; 9].

C:\Users\fedya\source\repos\CPP_Fun\Debug\CPP_Fun.exe (process 20384) exited with code 1.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

6) n = 5 num_of_threads = 1000

```
Please, enter N: 5
Please, enter number of threads: 1000
Wrong input! Num of threads cannot be less than 1 or more than 4!

C:\Users\fedya\source\repos\CPP_Fun\Debug\CPP_Fun.exe (process 19000) exited with code 1.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

7) n = 7 num_of_threads = -15

```
Please, enter N: 7
Please, enter number of threads: -15
Wrong input! Num of threads cannot be less than 1 or more than 4!

C:\Users\fedya\source\repos\CPP_Fun\Debug\CPP_Fun.exe (process 9708) exited with code 1.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```