Государственное образовательное учреждение высшего профессионального образования
«Московский Государственный Технический Университет им Н. Э. Баумана.»

**Отчет**
По лабораторной работе №6
По курсу «Анализ Алгоритмов»

Тема: «Конвейер»

Студент: Губайдуллин Р.Т.
Группа: ИУ7-51

Преподаватели: Волкова Л.Л.
Строганов Ю.В.

## Постановка задачи

Необходимо реализовать конвейер на произвольной вычислительной системе.
Провести тесты и сравнить последовательное и конвейерные вычисления.

### Теория

Конвейер - совокупность ступеней и средств передачи данных между ними, организованных таким образом, что на вход системы поступают исходные данные, затем они последовательно, в соответствии с разбиением базовой функции на подфункции, перемещаются между ступенями, подвергаясь на каждом этапе промежуточной обработке, в результате чего на выходе получается требуемый результат. Одновременно в конвейере может находиться более одного элемента входных данных.

### Реализация

Конвейер реализован на языке C# с использованием System.Threading.

Исходная функция F(x) = F3(F2(F1(x))) разбита на 3 подзадачи:

- F1(x) = X*X
- F2(x) = X + 1
- F3(x) = X * 2

Для каждой подфункции выделяется свой поток.

Код программы.

```
static void f1()
        {
            int temp = 0;
            for (int i = 0; i < len; i++)
            {
                temp = input[i] * input[i];
                lock (locker)
                {
                    queue1.Enqueue(temp);
                }
            }
        }

        static void f2()
        {
            int temp;
            while (work)
            {
                if (queue1.Count != 0)
                {
                    lock (locker)
                    {
                        temp = queue1.Dequeue();
                    }
                    lock (locker2) {
                        queue2.Enqueue(temp + 1);
                    }
                }
            }
        }

        static void f3()
        {
            int temp2;
            while (work)
            {
                if (queue2.Count != 0)
                {
                    lock (locker2)
                    {
                         temp2 = queue2.Dequeue();
                    }

                    output[schetchik] = temp2 * 2;

                    lock (locker3)
                    {
                        queue3.Enqueue(temp2);
                    }
                    schetchik++;
                    if (schetchik == len)
                        work = false;
                }
            }
        }
```

Где:
queue1, queue2, queue3 - Глобальные очереди
Input - входной массив
Output - выходной массив

Основная проблема при распараллеливании задачи - контроль за обращением программы к памяти. Для решении этой проблемы было использована команда lock, которая не дает другим потокам войти в выбранный фрагмент кода. Это решение сказывается на скорости. вычисления.

**Трудоемкость**

Если предположить, что t - число операций, выполняемых на ступени (в среднем), то трудоемкость конвейера:

$$F_k = N * t$$

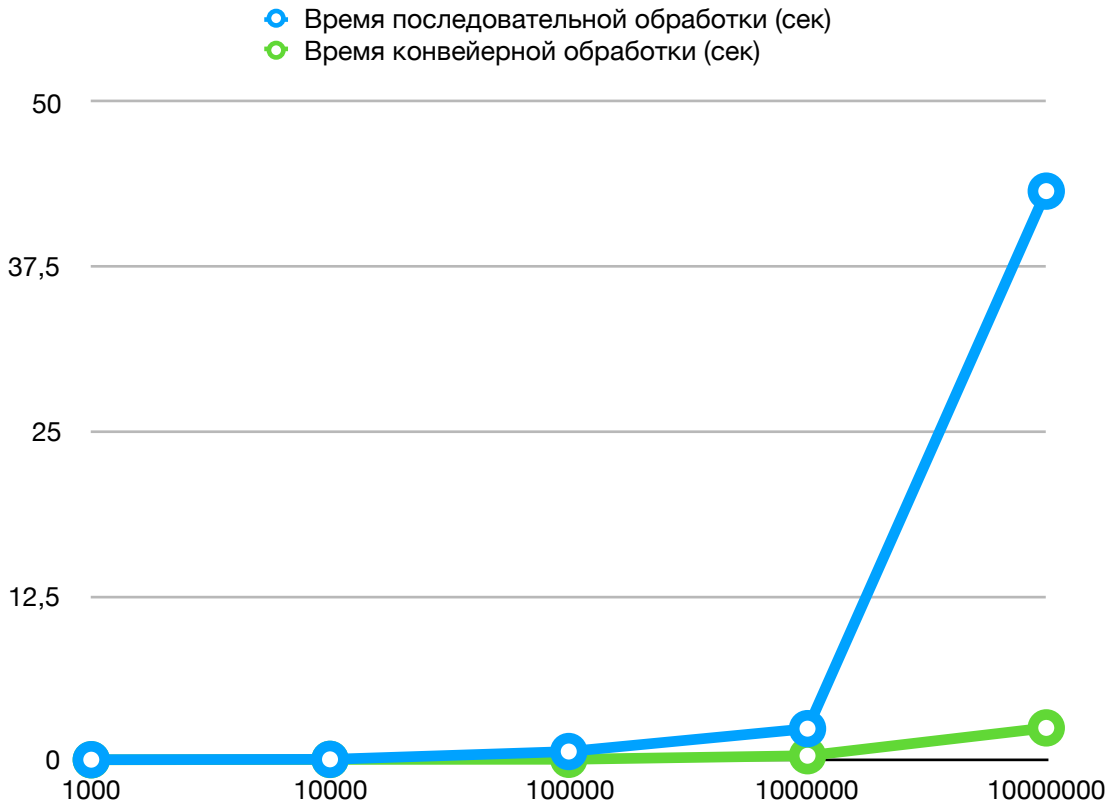
То трудоемкость последовательной обработки данных будет выглядеть:
$F_{\text{посл}} = M * N * t$
где M - количество ступеней конвейера

**Временные эксперименты**

До определенного размера массива последовательная обработка эффективнее конвейерного.

| Размер массива | Время последовательной обработки (сек) | Время конвейерной обработки (сек) |
|---|---|---|
| 1000 | 00.0608074 | 00.0560760 |
| 10000 | 00.0875775 | 00.0797420 |
| 100000 | 00.6803432 | 00.0903568 |

| | | |
|---|---|---|
| 1000000 | 02.4174608 | 00.3613353 |
| 10000000 | 43.2289462 | 2.4750189 |




**Вывод:**
Конвейерные вычисления - один из видов параллелизации программ.

Плюсы конвейера:
- Быстрота вычислений больших объемов данных
- Разбиение сложной задачи на простые подзадачи
- Легкое добавление/удаление подзадач в конвейере