ОТЧЕТ

По лабораторной работе №5

По курсу «Анализ алгоритмов»

Тема: «Конвейерная обработка данных»

|  |  |
| --- | --- |
|  | **Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  «Московский государственный технический университет  имени Н.Э. Баумана  (национальный исследовательский университет)»  (МГТУ им. Н.Э. Баумана) |

Студент: Зейналов З. Г.

Группа: ИУ7-51

Преподаватель: Волкова Л.Л.

Москва, 2019г.

Оглавление

[Введение 4](#_Toc25189262)

[1. **Аналитическая часть** 5](#_Toc25189263)

[2. Конструкторская часть 6](#_Toc25189264)

[2.1 Описание архитектуры ПО 7](#_Toc25189265)

[2.2 Разработка алгоритмов 8](#_Toc25189266)

[3 Технологическая часть 10](#_Toc25189267)

[3.1 Средства реализации 10](#_Toc25189268)

[3.2 Листинг кода 11](#_Toc25189269)

[4 Экспериментальная часть 16](#_Toc25189270)

[4.1 Анализ лог-файла 16](#_Toc25189271)

[Вывод 18](#_Toc25189274)

[Заключение 19](#_Toc25189275)

# Введение

Выполнение каждой команды складывается из ряда последовательных этапов (шагов, стадий), суть которых не меняется от команды к команде. С целью увеличения быстродействия какого-либо процесса и максимального использования всех возможностей процессора в современных системах используется конвейерный принцип обработкиинформации. Этот принцип подразумевает, что в каждый момент времени процессор работает над различными стадиями выполнения нескольких команд, причем на выполнение каждой стадии выделяются отдельные аппаратные ресурсы.

Конвейеризация (или конвейерная обработка) в общем случае основана на разделении подлежащей исполнению функции на более мелкие части, называемые ступенями, и выделении для каждой из них отдельного блока аппаратуры. Производительность при этом возрастает благодаря тому, что одновременно на различных ступенях конвейера выполняются несколько команд. Конвейерная обработка такого рода широко применяется во всех современных быстродействующих процессорах.

Цель данной лабораторной работы: получить навык организации асинхронной передачи данных между потоками на примере конвейерной обработки информации.

Задачи лабораторной работы следующие.

1. Выбрать и описать методы обработки данных, которые будут сопоставлены методам конвейера.
2. Описать архитектуру программы, а именно какие функции имеет главный поток, принципы и алгоритмы обмена данными между потоками.
3. Реализовать конвейерную систему, а также сформировать лог событий с указанием времени их происхождения, описать реализацию.
4. Провести тестирование системы.
5. Интерпретировать сформированный лог.

1. **Аналитическая часть**

В данном разделе будут описан алгоритм работы конвейерного процесса

Конвейер будет состоять из трех уровней. Генератор подает на вход конвейера (первый уровень) некоторые числа. Далее на каждом уровне осуществляется обработка данных, занимающая определенное время. Обработанные данные передаются последовательно с одного уровня (одной ленты) конвейера на следующий (следующую ленту). Для организации работы каждой ленты будет использована очередь задач, которые должны обработаться на этой ленте. Таким образом, лента будет работать, пока в её очереди есть задачи, но задачи попадают в очередь только в том случае, если данные уже были обработаны на предыдущем уровне или если их только сформировал генератор (для первой ленты). При этом время обработки на какой-либо ленте не должно сильно отличаться от времени обработки на остальных лентах, так как в противном случае возможны ситуации простоя одной или нескольких лент конвейера. На последнем уровне конвейера обработанные объекты попадают в пул обработанных задач. После завершения работы конвейера, то есть после завершения работы третьей ленты, проверяется равенство количества поданных на вход конвейера объектов количеству объектов в пуле обработанных задач. Если равенство верное, то конвейер отработал корректно, иначе где-то произошла ошибка и какие-то данные были обработаны неправильно, либо вообще были потеряны.

На каждом уровне конвейера алгоритм обработки информации заменен задержкой выполняемой программы по времени.

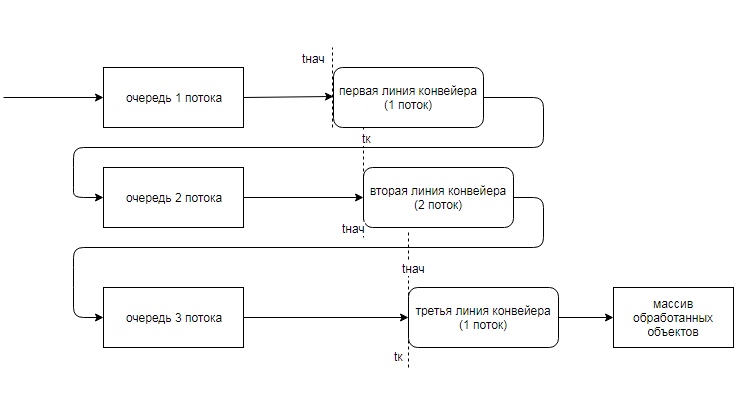
## Конструкторская часть

В данном разделе будет представлена схема работы конвейера, а так же описана архитектура ПО.

### Описание архитектуры ПО

В данном разделе будут размещены схемы алгоритмов и описана архитектура программы.

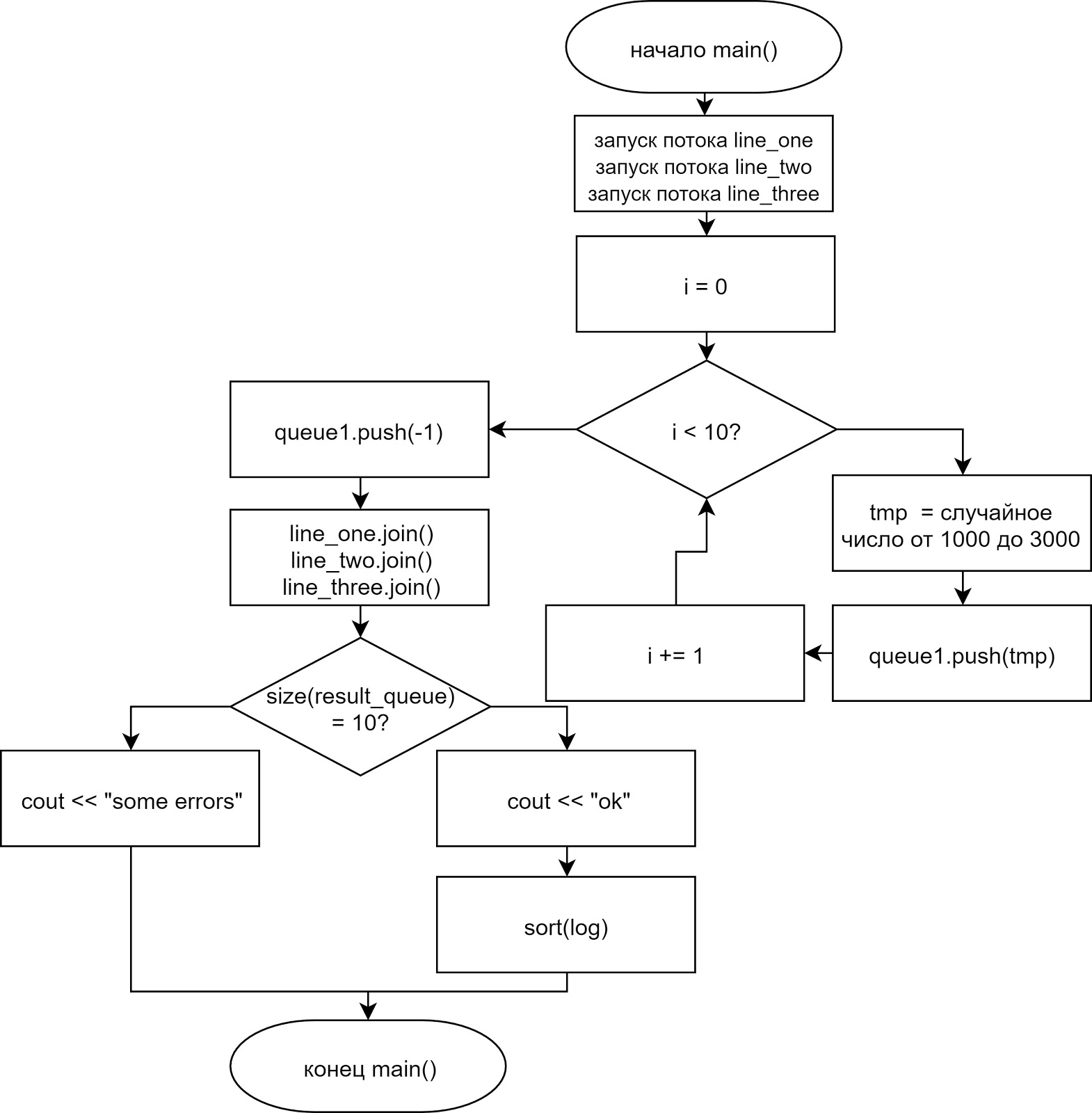
Каждой ленте конвейера выделен отдельный поток. В главном потоке main запускаются все три рабочих потока - ленты конвейера:  firstStep, secondStep, thirdStep. Для каждого потока есть своя очередь, в которой содержатся индекс выполняемой задачи и время в миллисекундах, необходимое на выполнение задачи (время задержки). Также в главном потоке генерируются входные данные, которые помещаются в очередь для первого потока. В рабочих потоках извлекается очередной элемент из соответствующей очереди, вносится запись в пулл обработанных объектов, выполняется задержка по времени. После этого новое значение помещается в очередь для следующего потока (или в массив обработанных значений если обработка происходит уже в третьем потоке) и далее в лог-файл заносится запись о завершении обработки очередного элемента на определенной ленте (в определенном потоке). Последним на вход в 1 очередь подается значение индекса равного -1, что говорит о завершении работы конвейера. Это значение передается из очереди в очередь по лентам. На рисунке 1 представлена схема работы конвейера.



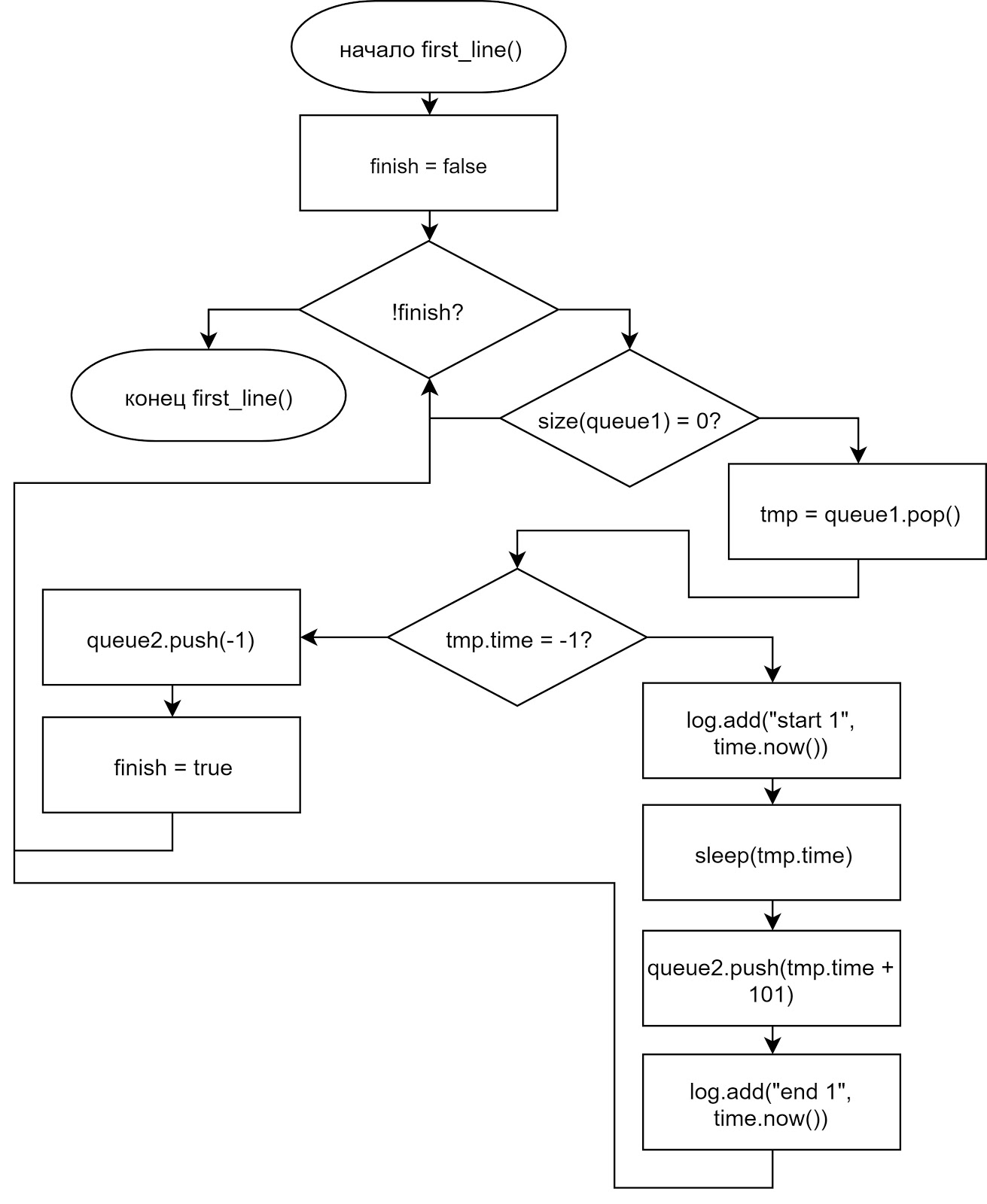
*Рисунок 1 - схема работы конвейера*

### Разработка алгоритмов

На рисунках 2 – 3 приведены схемы алгоритма, демонстрирующего работу.



*Рисунок 2 - схема работы главного потока*



*Рисунок 3 - схема работы алгоритма рабочего потока*

## Технологическая часть

В данном разделе будут приведены Требования к программному обеспечению, средства реализации, листинг кода и примеры тестирования.

### Средства реализации

В качестве языка программирования был выбран С++ (компилятор g++) в связи с его широким функционалом и быстротой работы, а так же благодаря привычному для меня синтаксису и семантики языка. Среда разработки - Qt. Для работы с потоками использовалась библиотека thread. Так как в ходе работы программы возможно обращение разных потоков к одному участку памяти, используются объекты класса mutex. Для реализации очереди выбрана библиотека queue.

### Листинг кода

В листинге 1 представлена функция главного потока. В листингах 2 – 4 представлены ленты конвейера. Очереди, мьютексы описаны как глобальные переменные.

int **main**()

{

thread tr1 (*firstStep*);

thread tr2 (*secondStep*);

thread tr3 (*thirdStep*);

fiilObjectPool(10);

*for* (int i = 0; i < objectPool.size(); i++)

{

mtx1.lock();

qFirst.push(objectPool[i]);

mtx1.unlock();

Sleep(50);

}

std::cout << objectPool.size() << " is isize" << endl;

tr1.join();

tr2.join();

tr3.join();

sortOutput();

ofstream file;

file.open("results.txt");

*for* (int i = 0; i < resultPool.size(); i++)

{

file << i << " " << resultPool[i].index << resultPool[i].str << ctime(&(resultPool[i].time)) << endl;

}

file.close();

std::cout << "all products were pushed into ResultObjectPool";

*return* 0;

}

Листинг 1 – функция главного потока - main

void **firstStep**(void)

{

bool finish = *false*;

*while* (!finish)

{

mtx1.lock();

*if*(!qFirst.empty())

{

inputObject obj = qFirst.front();

*if* (obj.itemindex != -1)

{

time(&obj.time);

obj.index = 1;

qFirst.pop();

Sleep(1000);

mtx2.lock();

qSecond.push(obj);

mtx2.unlock();

resmtx.lock();

resultPool.push\_back(obj);

resmtx.unlock();

}

*else*

{

mtx2.lock();

qSecond.push(obj);

mtx2.unlock();

finish = *true*;

}

}

mtx1.unlock();

}

}

Листинг 2 – Первая лента конвейера

Листинг 3 – Вторая лента конвейера

void **secondStep**(void)

{

bool finish = *false*;

*while* (!finish)

{

mtx2.lock();

*if* (!qSecond.empty())

{

inputObject obj = qSecond.front();

*if* (obj.itemindex != -1)

{

time(&obj.time);

obj.index = 2;

qSecond.pop();

Sleep(1000);

mtx3.lock();

qThird.push(obj);

mtx3.unlock();

resmtx.lock();

resultPool.push\_back(obj);

resmtx.unlock();

}

*else*

{

mtx3.lock();

qThird.push(obj);

mtx3.unlock();

finish = *true*;

}

}

mtx2.unlock();

}

}

void **thirdStep**(void)

{

bool finish = *false*;

*while* (!finish)

{

mtx3.lock();

*if* (!qThird.empty())

{

inputObject obj = qThird.front();

*if* (obj.itemindex != -1)

{

time(&obj.time);

obj.index = 3;

qThird.pop();

Sleep(1000);

resmtx.lock();

resultPool.push\_back(obj);

resmtx.unlock();

*//std::cout* *<<* *c.resultPool.size()* *<<* *"* *size"* *<<* *endl;*

}

*else*

{

finish = *true*;

}

}

mtx3.unlock();

}

}

Листинг 4 – Последняя лента конвейера

## Экспериментальная часть

В данном разделе будут приведены примеры работы программы, а также будет приведена интерпретация сформированного программой лог-файла.

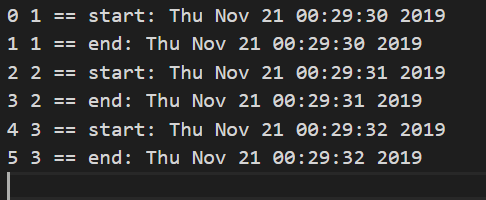
### Анализ лог-файла

Ниже на рисунке 4 приведен лог файл – результат работы программы.

### 

*Рисунок 4 - лог файл*

Лог файл состоит из упорядоченных по временному возрастанию событий, происшедших в программе. В первом столбце указан индекс записи, во втором – индекс конвейерной ленты и время начала или конца. В данном примере на обработку объекта на каждом конвейере затрачивается одинаковое кол-во времени, а именно ~1 секунда. Как мы видим, время простоя конвейерных линий на каждом элементе равно 1 секунде.



*Рисунок 5 - 1 элемент на входе.*

### В случае, если на вход поступает 1 элемент (Рисунок 5), то можно отследить поэтапную обработку объекта на конвейерных линиях.

### В случае, если каждая линия конвейера обрабатывает объект с одинаковым количеством времени, то использование многопоточной реализации конвейерной обработки теряет свой смысл. В ином случае, если каждая из конвейерных линий обрабатывает объект с разным количеством времени, то использование такой конвейерной обработки становится разумным.

# Заключение

В ходе лабораторной работы был изучен и реализован вычислительный конвейер с использованием методов распараллеливания процессов. Были выявлены оптимальные для конвейерной обработки параметры входных задач: сбалансированное время обработки данных на всех лентах (уровнях) конвейера.