



Министерство науки и высшего образования 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)

---

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

## Отчет по лабораторной работе №5 по курсу "Анализ алгоритмов"

Тема Конвейерная обработка данных

Студент Прохорова Л. А.

Группа ИУ7-53Б

Оценка (баллы) \_\_\_\_\_

Преподаватели Волкова Л.Л., Строганов Ю.В.

Москва — 2020 г.

## Содержание

<b>Введение</b>	<b>2</b>
<b>1 Аналитическая часть</b>	<b>3</b>
<b>Аналитическая часть</b>	<b>3</b>
1.1 Описание задачи . . . . .	3
<b>2 Конструкторская часть</b>	<b>7</b>
<b>Конструкторская часть</b>	<b>7</b>
2.1 Описание архитектуры ПО . . . . .	7
2.2 Схемы алгоритмов работы конвейерной обработки. . . . .	9
<b>3 Технологический раздел</b>	<b>12</b>
3.1 Выбор языка программирования . . . . .	12
3.2 Технические характеристики . . . . .	12
3.3 Листинги кода . . . . .	12
3.4 Тестирование программы . . . . .	16
<b>4 Исследовательская часть</b>	<b>17</b>
<b>Исследовательская часть</b>	<b>17</b>
4.1 Демонстрация работы программы . . . . .	17
<b>Литература</b>	<b>22</b>

## Введение

Имеется большое количество важнейших задач, решение которых требует использования огромных вычислительных мощностей, зачастую недоступных для современных вычислительных систем.

Постоянно появляются новые задачи подобного рода и возрастают требования к точности и к скорости решения прежних задач; поэтому вопросы разработки и использования сверхмощных компьютеров (называемых суперкомпьютерами) актуальны сейчас и в будущем [1]. Но пока эти трудности пока что не удается преодолеть. Из-за этого приходится и эти по пути создания параллельных вычислительных систем, т.е. систем, в которых предусмотрена одновременная реализация ряда вычислительных процессов, связанных с решением одной задачи [2]. На современном этапе развития вычислительной техники такой способ, по-видимому, является одним из основных способов ускорения вычислений.

## **1 Аналитическая часть**

В данном разделе будет поставлена цель и описаны задачи, описана идея конвейеризации.

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

Для достижения поставленной цели требуется выполнить следующие задачи.

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.

Одной из важнейших идей при создании многопроцессорных систем и при эффективной реализации алгоритмов на этих системах является идея конвейерных вычислений.

### **1.1 Описание задачи**

Конвейеризация – это техника, в результате которой задача или команда разбивается на некоторое число подзадач, которые выполняются последовательно. Каждая подкоманда выполняется на своем логическом

устройстве. Все логические устройства (ступени) соединяются последовательно таким образом, что выход  $i$ -ой ступени связан с входом  $(i+1)$ -ой ступени, все ступени работают одновременно. Множество ступеней называется конвейером. Выигрыш во времени достигается при выполнении нескольких задач за счет параллельной работы ступеней, вовлекая на каждом такте новую задачу или команду [3].

В конвейере различают  $r$  последовательных этапов, так что когда  $i$ -я операция проходит  $s$ -й этап, то  $(i + k)$ -я операция проходит  $(s - k)$ -й этап. На рисунке 1 изображена работа конвейера.

## Работа конвейера

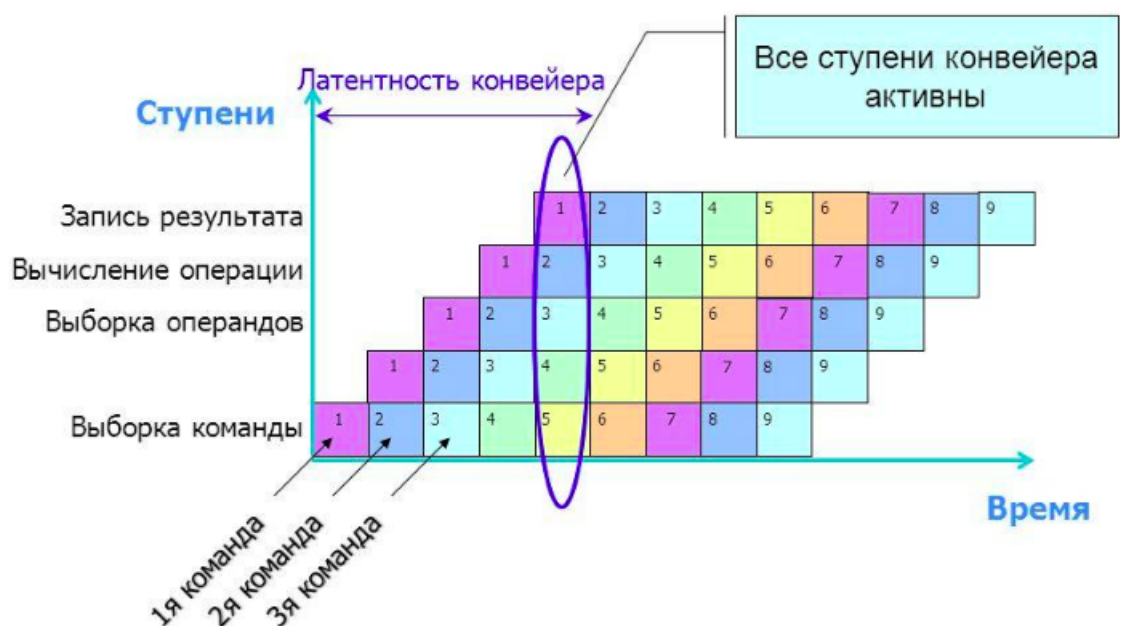


Рисунок 1 – Работа конвейера.

В данной лабораторной работе реализована некая функцию кодирования строки, которая состоит из трех последовательных действий: применения первой функции шифра Цезаря, применения функции которая меняет регистр буквы и применения функции которая меняет элемент местами с элементом, стоящим на  $n / 2$  от него, где  $n$  - размер строки. Если

необходимо закодировать какой-то массив строк, то можно использовать конвейерную обработку данных. Таким образом задача будет решена эффективнее, чем при последовательном применении алгоритмов к массиву значений.

Конвейер будет состоять из четырех уровней. Обработанные данные передаются последовательно с одного уровня (одной ленты) конвейера на следующий (следующую ленту). Далее на каждом уровне осуществляется обработка данных, занимающая определенное время. Для каждой ленты создается своя очередь задач, в которой хранятся все необработанные строки. На последнем уровне конвейера обработанные объекты попадают в пул обработанных задач.

Уровни конвейера:

- 0 уровень — генерация входных данных в первую очередь;
- 1 уровень(лента) — применение шифра Цезаря к строкам из первой очереди, запись результата во 2 очередь;
- 2 уровень(лента) — применение функции, меняющей регистры символов к строкам 2 очереди, запись результата в 3 очередь;
- 3 уровень(лента) — применение функции, меняющей местами символы в строке по определённому выше закону к строкам 3 очереди, запись результата в пул обработанных задач.

Поскольку запись в очередь и извлечение из очереди это не атомарные операции, необходимо создать их таковыми путем использования мьютексов (по одному на одну очередь) и критических секций, чтобы избежать ошибок в ситуации гонок.

## **Вывод**

В данном разделе были поставлена цель и заданы задачи, описана идея конвейеризации.

## 2 Конструкторская часть

В данном разделе будет представлено описание архитектуры ПО и схемы рабочего и главного процессов.

### 2.1 Описание архитектуры ПО

В конвейере 3 основные ленты, содержание их работы описано выше, в аналитической части отчета. Каждой ленте выделен свой поток, в котором она выполняется. В главном потоке (функция `main`) создаются три рабочих потока: по одному на каждую ленту. Для каждой ленты есть своя очередь, однако с ней могут работать все потоки, поэтому при доступе к элементам очереди необходимо блокировать доступ для других потоков. Для реализации доступа из разных потоков используются мьютексы, по одному для каждой очереди и для результирующего массива создан. Также в главном потоке генерируется массив входных данных и заполняется первая очередь (уровень 0). В рабочих процессах считывается по одному элементу из соответствующей очереди, выполняется вызов обрабатывающих функций, замеры времени и задержка по времени. После обработки текущей задачи, рабочий процесс записывает результат в следующую очередь или в результирующий массив. Также выполняется запись в лог-файл.

На рисунке 2 изображена схема работы конвейерной обработки.



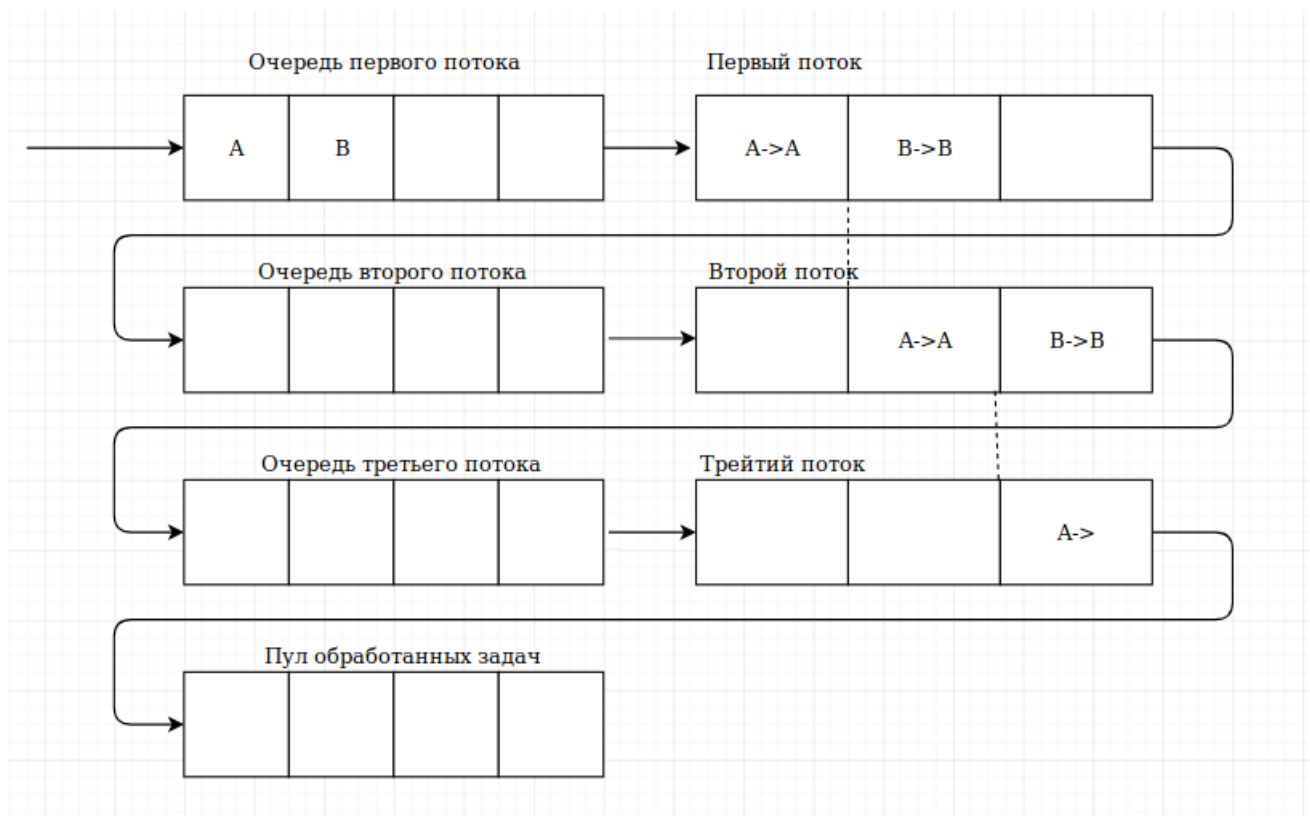


Рисунок 2 – Схема работы конвейерной обработки.

## 2.2 Схемы алгоритмов работы конвейерной обработки.

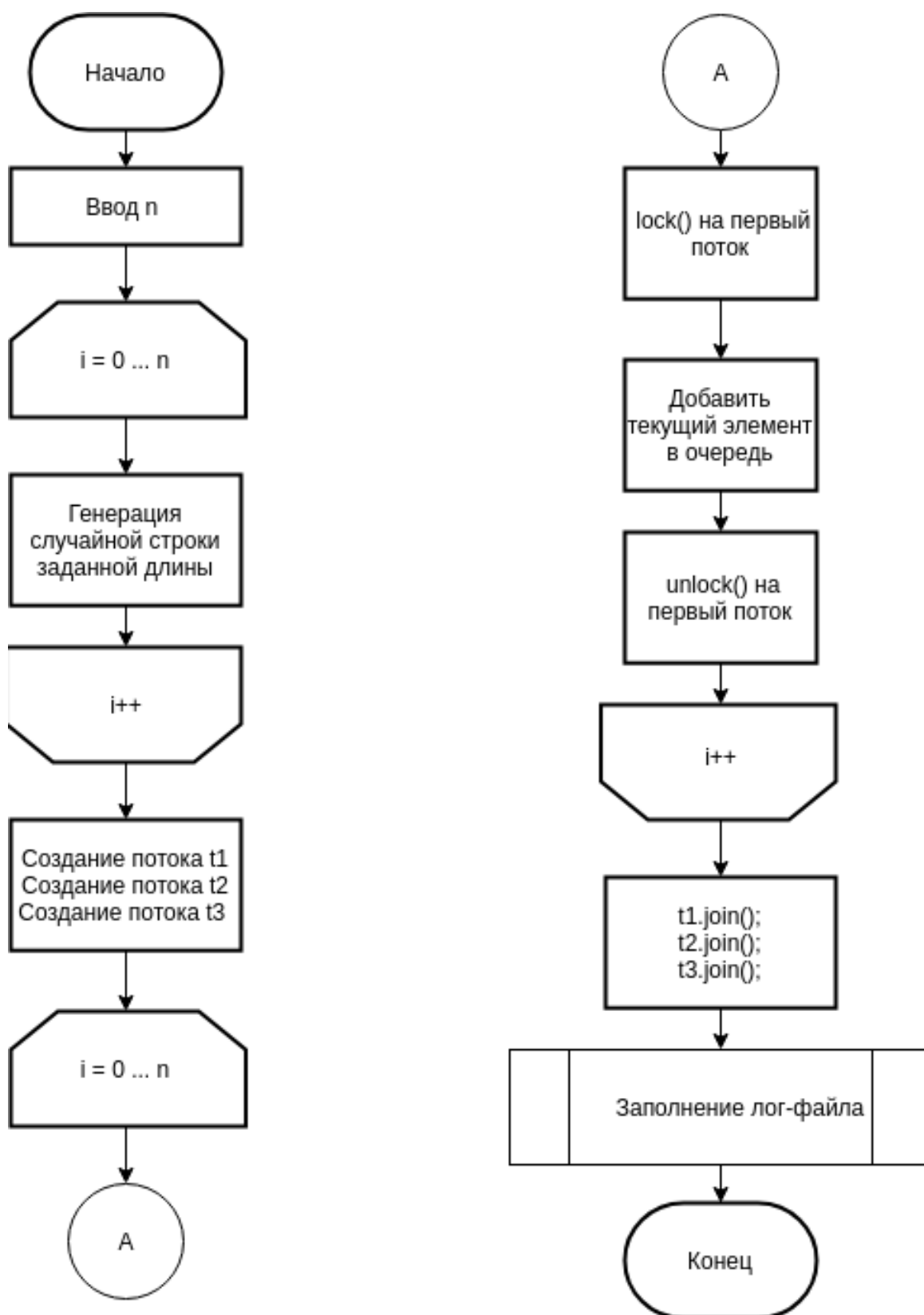


Рисунок 3 – Схема работы главного процесса.

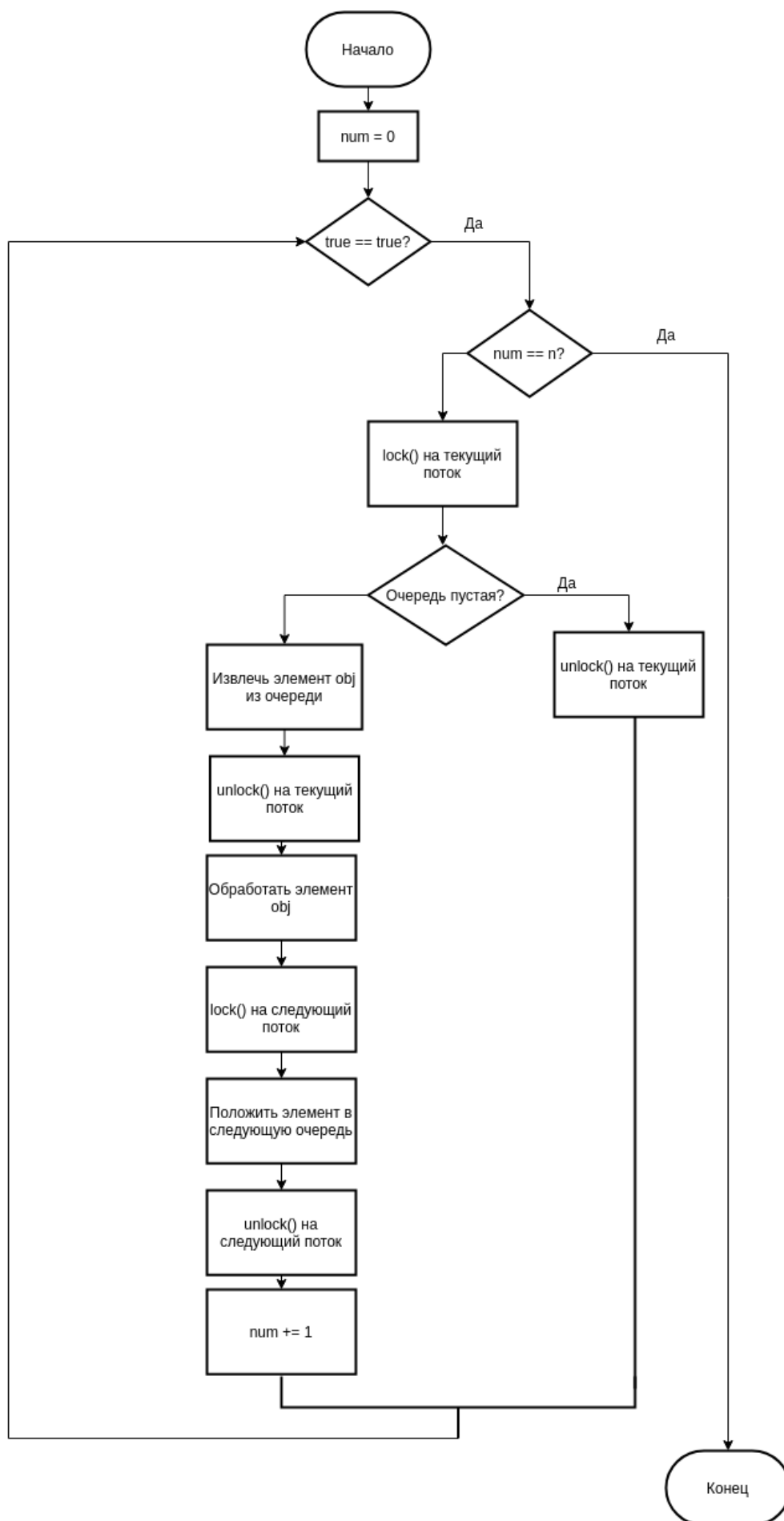


Рисунок 4 – Схема работы рабочего процесса.

На рисунках 3 и 4 показаны схем главного и рабочего процессов.

## **Вывод**

В данном разделе представлено описание архитектуры ПО и схемы рабочего и главного процессов.

### 3 Технологический раздел

В этом разделе будет обоснован выбор языка программирования, описаны технические характеристики, приведены листинги кода реализованных алгоритмов.

#### 3.1 Выбор языка программирования

В качестве языка программирования мной был выбран C++ так как этот язык удобен для работы с потоками. Для замера времени выполнения использовалась функция *clock()* из библиотеки *ctime*. Эта функция возвращает количество временных тактов, прошедших с начала запуска программы [4].

#### 3.2 Технические характеристики

Технические характеристики устройства, на котором выполнялось тестирование, следующие.

- Операционная система Ubuntu 18.04 64-bit.
- Память 8 GiB.
- Процессор Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz.

#### 3.3 Листинги кода

В листингах 1, 2, 3 представлен код рабочих потоков, в листинге 4 представлен код главного потока.

**Листинг 1** – Реализация первого уровня конвейера.

```
0 void first_conv() {  
1     int num = 0;  
2     while (true) {  
3         if (num == n)  
4             break;
```

```

5      m1.lock();
6      if (queue1.empty()) {
7          m1.unlock();
8          continue;
9      }
10     string cur_str = queue1.front().str;
11     int cur_task_num = queue1.front().task_num;
12     timer.add_time(1, clock() - queue1.front().time, queue1.front().
task_num);
13     queue1.pop();
14
15     clock_t cur_time = clock();
16     m1.unlock();
17     string new_str = caesar(cur_str);
18     m2.lock();
19     timer.add_time(0, clock() - cur_time, cur_task_num);
20
21     queue2.push(Object(new_str, cur_task_num, clock()));
22     m2.unlock();
23     num++;
24 }
25 }

```

## Листинг 2 – Реализация второго уровня конвейера.

```

0 void second_conv() {
1     int num = 0;
2     while (true) {
3         if (num == n)
4             break;
5         m2.lock(); // wait in queue
6         if (queue2.empty()) {
7             m2.unlock();
8             continue;
9         }
10        string cur_str = queue2.front().str;
11        int cur_task_num = queue2.front().task_num;
12        timer.add_time(1, clock() - queue2.front().time, queue2.front().
task_num);
13        queue2.pop();
14

```

```

15     clock_t cur_time = clock();
16     m2.unlock();
17     string new_str = upper_lower(cur_str);
18     m3.lock();
19     timer.add_time(0, clock() - cur_time, cur_task_num);
20
21     queue3.push(Object(new_str, cur_task_num, clock()));
22     m3.unlock();
23     num++;
24 }
25 }

```

### Листинг 3 – Реализация третьего уровня конвейера.

```

0 void third_conv() {
1     int num = 0;
2     while (true) {
3         if (num == n)
4             break;
5         m3.lock(); // wait in queue
6         if (queue3.empty()) {
7             m3.unlock();
8             continue;
9         }
10        string cur_str = queue3.front().str;
11        int cur_task_num = queue3.front().task_num;
12        timer.add_time(1, clock() - queue3.front().time, queue3.front().
task_num);
13        queue3.pop();
14
15        clock_t cur_time = clock();
16        m3.unlock();
17        string new_str = reverse(cur_str);
18        resm.lock();
19        timer.add_time(0, clock() - cur_time, cur_task_num);
20
21        res.push_back(new_str);
22        resm.unlock();
23        num++;
24    }
25 }

```

#### Листинг 4 – Основная функция программы.

```
0 int main(){
1     srand(time(nullptr));
2     cout << "Введите количество строк: ";
3     cin >> n;
4     if (n <= 0){
5         cout << "Некорректное количество строк";
6         return -1;
7     }
8
9     objvec.resize(n);
10    timer.set_size(n, 3);
11
12    for (int i = 0; i < n; i++){
13        string s = generate();
14        objvec[i] = (s);
15    }
16
17    clock_t start_t = clock();
18
19    thread t1(first_conv);
20    thread t2(second_conv);
21    thread t3(third_conv);
22
23    for (int i = 0; i < n; i++) {
24        m1.lock();
25        queue1.push(Object(objvec[i], i, clock()));
26        m1.unlock();
27    }
28
29    t1.join();
30    t2.join();
31    t3.join();
32
33    create_log(clock() - start_t);
34    return 0;
35 }
```



### **3.4 Тестирование программы**

На вход подаем 6 строк длиной 50000. Программа выдала файл с логом.

Все тесты пройдены успешно.

#### **Вывод**

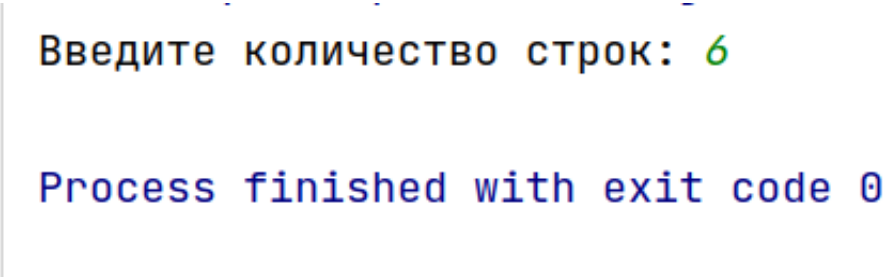
В этом разделе обоснован выбор языка программирования, описаны технические характеристики, приведены листинги кода реализованных алгоритмов. Программа прошла тестирование и работает правильно.

## 4 Исследовательская часть

В этом разделе будет приведена демонстрация работы программы и исследование полученных результатов.

### 4.1 Демонстрация работы программы

Демонстрация работы программы приведена на рисунке 5. На вход подаётся количество строк для обработки.

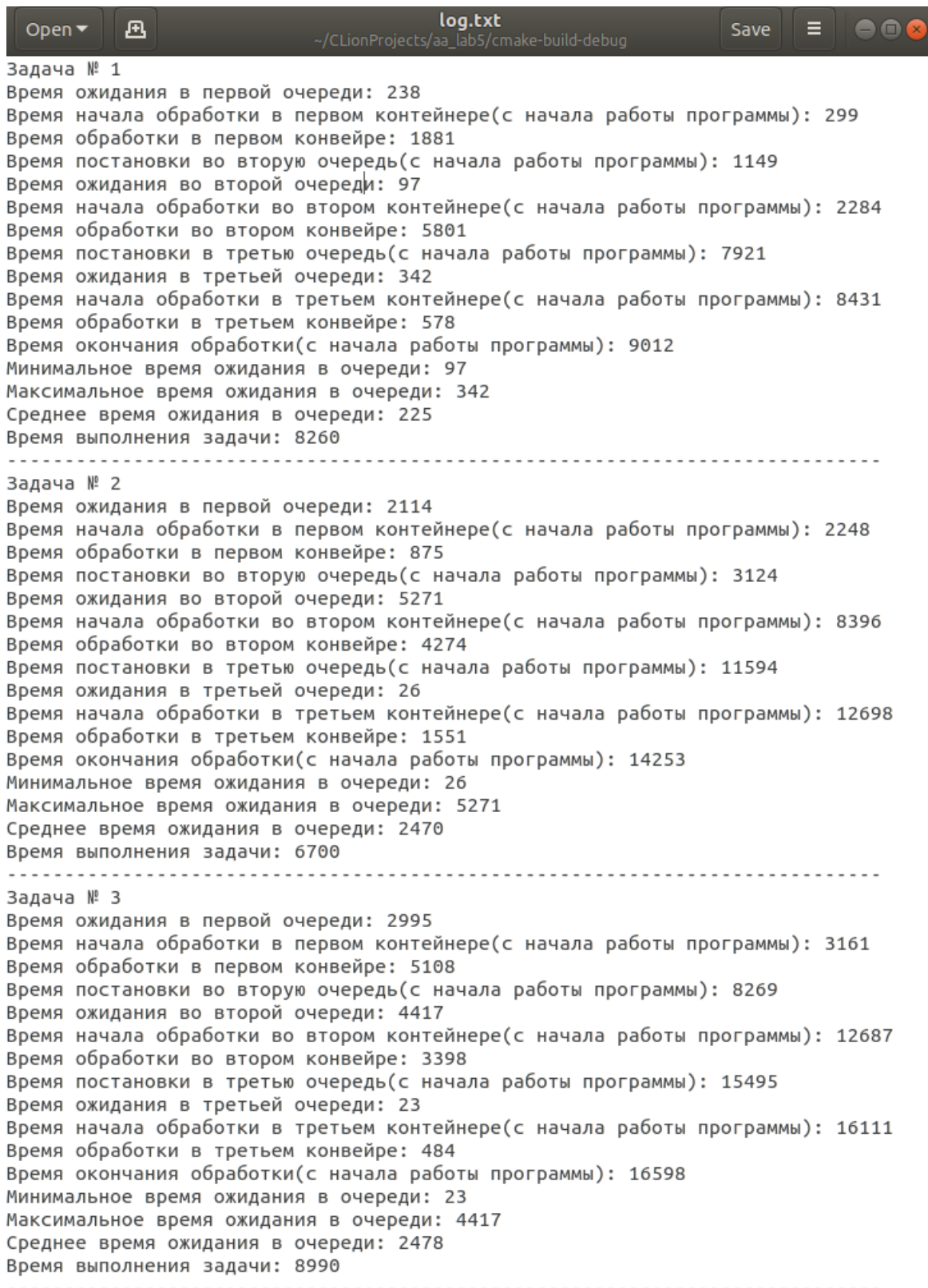


```
Введите количество строк: 6
```

```
Process finished with exit code 0
```

Рисунок 5 – Демонстрация работы программы.

В результате работы программы заполняется лог-файл. Содержимое лог-файла приведено на рисунках 6, 7. Время замерено с тактах процессора.

A screenshot of a code editor window titled 'log.txt' with a path '~/.CLionProjects/aa\_lab5/cmake-build-debug'. The window contains three sections of log data, each separated by a dashed line. Each section starts with 'Задача №' followed by a task number. The data lists various time metrics for three different stages (queues) of a program, including waiting times, processing times in containers and queues, and completion times. The metrics are listed in Russian.

```
Open [icon] log.txt Save [icon] [icon] [icon] [icon]
~/CLionProjects/aa_lab5/cmake-build-debug

Задача № 1
Время ожидания в первой очереди: 238
Время начала обработки в первом контейнере(с начала работы программы): 299
Время обработки в первом конвейере: 1881
Время постановки во вторую очередь(с начала работы программы): 1149
Время ожидания во второй очереди: 97
Время начала обработки во втором контейнере(с начала работы программы): 2284
Время обработки во втором конвейере: 5801
Время постановки в третью очередь(с начала работы программы): 7921
Время ожидания в третьей очереди: 342
Время начала обработки в третьем контейнере(с начала работы программы): 8431
Время обработки в третьем конвейере: 578
Время окончания обработки(с начала работы программы): 9012
Минимальное время ожидания в очереди: 97
Максимальное время ожидания в очереди: 342
Среднее время ожидания в очереди: 225
Время выполнения задачи: 8260
-----

Задача № 2
Время ожидания в первой очереди: 2114
Время начала обработки в первом контейнере(с начала работы программы): 2248
Время обработки в первом конвейере: 875
Время постановки во вторую очередь(с начала работы программы): 3124
Время ожидания во второй очереди: 5271
Время начала обработки во втором контейнере(с начала работы программы): 8396
Время обработки во втором конвейере: 4274
Время постановки в третью очередь(с начала работы программы): 11594
Время ожидания в третьей очереди: 26
Время начала обработки в третьем контейнере(с начала работы программы): 12698
Время обработки в третьем конвейере: 1551
Время окончания обработки(с начала работы программы): 14253
Минимальное время ожидания в очереди: 26
Максимальное время ожидания в очереди: 5271
Среднее время ожидания в очереди: 2470
Время выполнения задачи: 6700
-----

Задача № 3
Время ожидания в первой очереди: 2995
Время начала обработки в первом контейнере(с начала работы программы): 3161
Время обработки в первом конвейере: 5108
Время постановки во вторую очередь(с начала работы программы): 8269
Время ожидания во второй очереди: 4417
Время начала обработки во втором контейнере(с начала работы программы): 12687
Время обработки во втором конвейере: 3398
Время постановки в третью очередь(с начала работы программы): 15495
Время ожидания в третьей очереди: 23
Время начала обработки в третьем контейнере(с начала работы программы): 16111
Время обработки в третьем конвейере: 484
Время окончания обработки(с начала работы программы): 16598
Минимальное время ожидания в очереди: 23
Максимальное время ожидания в очереди: 4417
Среднее время ожидания в очереди: 2478
Время выполнения задачи: 8990
-----
```

Рисунок 6 – Содержимое лог-файла.

```

-----
Задача № 4
Время ожидания в первой очереди: 8119
Время начала обработки в первом контейнере(с начала работы программы): 8315
Время обработки в первом конвейере: 1569
Время постановки во вторую очередь(с начала работы программы): 9885
Время ожидания во второй очереди: 6215
Время начала обработки во втором контейнере(с начала работы программы): 16101
Время обработки во втором конвейере: 3301
Время постановки в третью очередь(с начала работы программы): 18548
Время ожидания в третьей очереди: 22
Время начала обработки в третьем контейнере(с начала работы программы): 19412
Время обработки в третьем конвейере: 518
Время окончания обработки(с начала работы программы): 19997
Минимальное время ожидания в очереди: 22
Максимальное время ожидания в очереди: 8119
Среднее время ожидания в очереди: 4785
Время выполнения задачи: 5388
-----
Задача № 5
Время ожидания в первой очереди: 9671
Время начала обработки в первом контейнере(с начала работы программы): 9896
Время обработки в первом конвейере: 3669
Время постановки во вторую очередь(с начала работы программы): 13580
Время ожидания во второй очереди: 5847
Время начала обработки во втором контейнере(с начала работы программы): 19427
Время обработки во втором конвейере: 2979
Время постановки в третью очередь(с начала работы программы): 22426
Время ожидания в третьей очереди: 8
Время начала обработки в третьем контейнере(с начала работы программы): 22436
Время обработки в третьем конвейере: 579
Время окончания обработки(с начала работы программы): 23049
Минимальное время ожидания в очереди: 8
Максимальное время ожидания в очереди: 9671
Среднее время ожидания в очереди: 5175
Время выполнения задачи: 7227
-----
Время работы системы 23095|

```

**Рисунок 7** – Содержимое лог-файла.

По лог-файлу можно проследить выполнение каждого этапа задач. Также можно проследить, что обработка выполняется параллельно. Например, второй конвейер не ждёт полного завершения работы первого конвейера прежде, чем начать работу, а начинает выполнение как только в очередь поступает первый элемент.

Если бы использовалась линейная реализация, то проделанная работа заняла бы 36656 тактов процессора, что в 1,5 раза больше, чем время, потраченное при конвейерной реализации.

## Вывод

Конвейерная обработка данных - полезный инструмент, который уменьшает время выполнения программы за счёт параллельной обработки данных. Самым эффективным временем считается время, когда все линии конвейера работают параллельно, обрабатывая свои задачи. Этот метод даёт выигрыш по времени в том случае, когда выполняемые задачи намного больше по времени, чем время, затрачиваемое на реализацию конвейера (работу с потоками, переключивание из очереди в очередь и тд).

## Заключение

В ходе лабораторной работы была достигнута цель. Был получен навык организации асинхронной передачи данных между потоками на примере конвейерной обработки информации.

Для достижения поставленной цели были выполнены следующие задачи.

1. Выбраны и описаны методы обработки данных, которые будут сопоставлены методам конвейера.
2. Описана архитектура программы, а именно какие функции имеет главный поток, принципы и алгоритмы обмена данными между потоками.
3. Реализована конвейерная система, а также сформирован лог событий с указанием времени их происхождения, описана реализация.
4. Проведено тестирование системы.
5. Интерпретирован сформированный лог.

Экспериментальным путём выявлено, что при большой нагрузке конвейер работает примерно в 1,5 раза быстрее чем линейная реализация, что является показателем эффективности работы конвейера по времени.

## Список литературы

- [1] Воеводин В.В. Математические модели и методы в параллельных процессах. М., 1986. 296 с.
- [2] Корнеев В.В. Параллельные вычислительные системы. М., 1999. 320 с.
- [3] Конвейерные вычисления [Электронный ресурс].Режим доступа:<http://www.myshared.ru/slide/674082> Дата обращения: 31.10.2020
- [4] Функция `clock()`. [Электронный ресурс].Режим доступа <http://cppstudio.com/post/561/> Дата обращения 01.11.2020