



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» (ИУ7)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника

О Т Ч Е Т

по лабораторной работе № 5

Название: Конвейер

Дисциплина: Анализ алгоритмов

Студент

ИУ7-52Б

(Группа)

Сучков А.Д.

(Подпись, дата)

(И.О. Фамилия)

Преподаватель

Волкова Л.Л.

(Подпись, дата)

(И.О. Фамилия)

Москва, 2020

Оглавление

| | |
|--|-----------|
| Введение | 3 |
| 1 Аналитическая часть | 4 |
| 1.1 Конвейер и конвейерная обработка | 4 |
| 1.2 Алгоритмы шифрования строк | 5 |
| 1.3 Выводы | 5 |
| 2 Конструкторская часть | 6 |
| 2.1 Схемы алгоритмов | 6 |
| 2.2 Конвейеризация алгоритмов | 7 |
| 2.3 Вывод | 8 |
| 3 Технологическая часть | 9 |
| 3.1 Выбор языка программирования | 9 |
| 3.2 Листинг кода | 9 |
| 3.3 Результаты выполнения программы | 13 |
| 3.4 Оценка времени | 13 |
| 3.5 Вывод | 14 |
| 4 Исследовательская часть | 15 |
| 4.1 Результаты экспериментов | 15 |
| 4.2 Вывод | 17 |
| Заключение | 18 |
| Список литературы | 19 |

Введение

В данной лабораторной работе реализуются и оцениваются конвейерные вычисления на примере шифрования строк.

Конвейер – машина непрерывного транспорта, предназначенная для перемещения разного рода грузов.

Вычислительный конвейер – способ организации вычислений, используемый в современных процессорах и контроллерах с целью повышения их производительности (увеличения числа инструкций, выполняемых в единицу времени — эксплуатация параллелизма на уровне инструкций), технология, используемая при разработке компьютеров и других цифровых электронных устройств.

1. Аналитическая часть

Целью лабораторной работы является разработка и исследование конвейерных вычислений.

Можно выделить следующие задачи лабораторной работы:

- описание понятия конвейерных вычислений и их применение на практике;
- реализация конвейерных вычислений на примере шифрования строк;
- проведение замеров процессорного времени работы алгоритмов;
- анализ полученных результатов.

1.1. Конвейер и конвейерная обработка

Идея конвейера в обобщённом смысле базируется на разделении выполняемой операции на более мелкие составляющие, которые называются подфункциями, и предоставлении для выполнения каждой подфункции своего аппаратного блока [1].

Если рассматривать вычислительный конвейер, то он предполагает перемещение команд или данных по этапам цифрового вычислительного конвейера со скоростью, не зависящей от протяжённости конвейера (или количества этапов), а зависит только от скорости подачи информации на конвейерные этапы. Скорость задаётся временем, в течение которого один компонент вычислительной операции способен пройти каждый этап, то есть самой большой задержкой на этапе, который выполняет отдельный участок функции. Это также значит, что скорость вычислений задаётся и скоростью поступления информации на вход конвейера.

В случае, когда какая-либо функция при её обычном выполнении реализуется за временной интервал T , но имеется возможность её деления на поочерёдное исполнение N подфункций, то в идеальном конвейере, если вычисление этой функции повторяется многократно, возможно её исполнение за временной период T/N , то есть в N раз увеличить производительность.

Различие реального и идеального конвейера заключается в наличии в реальной вычислительной системе различных помех. Общий смысл помехи заключается в присутствии фактора, который связан с самой функцией, конструктивными особенностями конвейера или его применения, препятствующих постоянному приходу новой информации на конвейерные этапы с самой большой скоростью.

1.2. Алгоритмы шифрования строк

Идея шифрования подразумевает под собой преобразование информации, которое скрывает её суть для посторонних. В то же время, те, кому предназначалась информация, способны дешифровать и обработать исходную информацию. Существует множество алгоритмов шифрования и дешифрования, но секретность данных заключается в том, что ключ шифрования известен только доверенным лицам.

В лабораторной работе будут реализованы шифры Вернама (XOR-шифр) и Цезаря.

Шифр Цезаря – имеет ключ в виде числа от 1 до 25 (для латиницы) и каждая буква алфавита смещается вправо или влево на ключевое число значений.

Шифр Вернама (XOR-шифр) – сообщение разбивается на отдельные символы и каждый символ представляется в бинарном виде. После чего по-символьно применяется операция XOR с ключом, в результате чего получается зашифрованное сообщение.

1.3. Выводы

Результатом аналитического раздела стало определение цели и задач работы, описание понятия вычислительного конвейера и алгоритмов шифрования.

2. Конструкторская часть

В данном разделе рассмотрим схемы описанных выше алгоритмов шифрования и описание способа их конвейеризации.

2.1. Схемы алгоритмов

На рисунках 2.1 - 2.2 представлены схемы выбранных алгоритмов шифрования строк

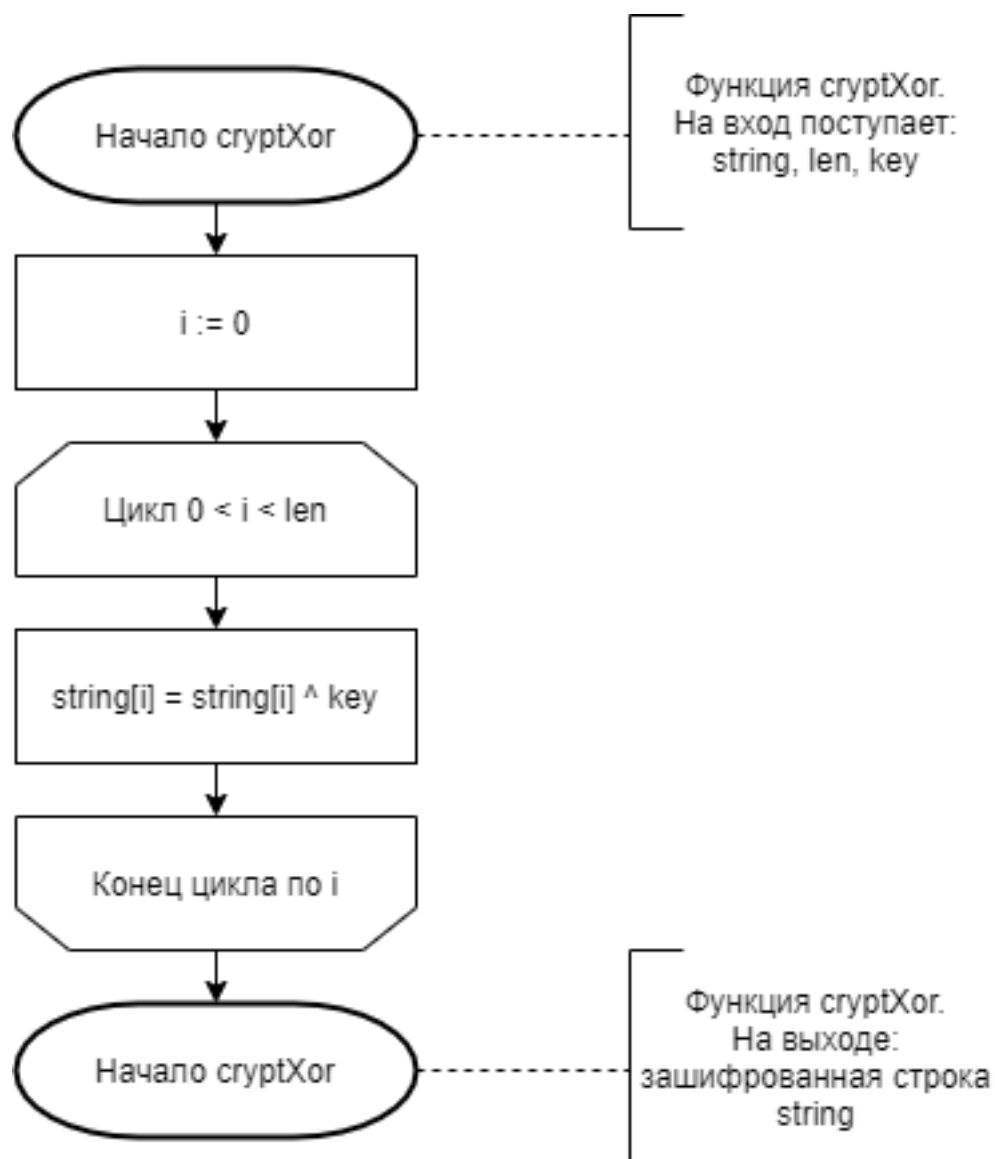


Рис. 2.1: Схема алгоритма шифрования Вернама (XOR-шифр)

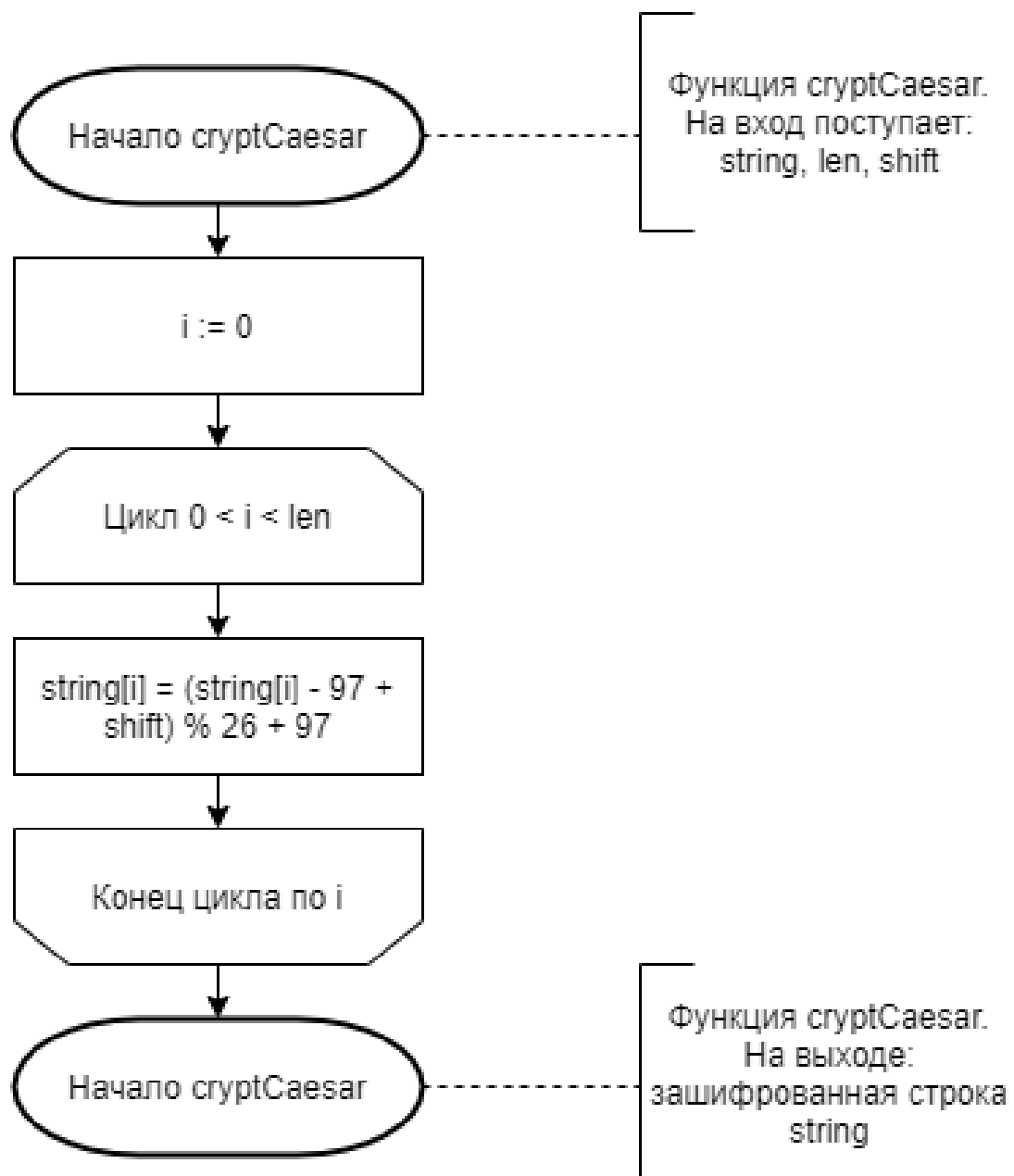


Рис. 2.2: Схема алгоритма шифрования Цезаря

2.2. Конвейеризация алгоритмов

Шифрование строки в программе разбивается на 3 этапа: первым применяется шифрование Цезаря, затем два раза XOR-шифр. Каждый из этих этапов выделен в отличную стадию выполнения конвейера.

Таким образом, главный поток при запуске вызывает генератор заявок, после чего создаёт 3 потока, каждому из которых выделяет определённую задачу.

2.3. Вывод

Результатом конструкторской части стало схематическое описание алгоритмов умножения матриц, сформулированы тесты и требования к программному обеспечению.

3. Технологическая часть

В данном разделе будет проведён выбор подходящих языка программирования и среды разработки, а также будут представлены реализации выбранных алгоритмов.

3.1. Выбор языка программирования

В качестве языка программирования был выбран C++, так как имеется опыт работы с ним и с библиотеками, позволяющими провести исследование и тестирование программы [2]. Также в языке имеются средства для использования многопоточности, что позволит реализовать конвейерные вычисления. Разработка проводилась в среде Visual Studio Code.

3.2. Листинг кода

В листингах 3.1 - 3.2 приведены реализации алгоритмов шифрования. В листингах 3.3 - 3.5 приведены 3 этапа выполнения конвейера. В листинге 3.6 представлен реализации основного потока и генератора заявок.

Листинг 3.1: функция XOR-шифра

```
1 void cryptXor(char k)
2 {
3     for (int i = 0; i < len; i++)
4         dataStr[i] ^= k;
5 }
```

Листинг 3.2: функция шифра Цезаря

```
1 void cryptCaesar(int shift)
2 {
3     for (int i = 0; i < len; i++)
4     {
5         dataStr[i] = (dataStr[i] - 97 + shift) % 26 + 97;
6     }
7 }
```

Листинг 3.3: первая часть выполнения конвейера

```
1 void Conveyor::part1()
2 {
3     for (; ft1 < ntask; ft1++)
4     {
5         Request *req;
6
7         if (startQ.size())
8         {
9             req = startQ.front();
10            startQ.pop();
11        }
12        else
13            continue;
14
15        req->timeS[0] = GetTime();
16        req->cryptCaesar(12);
17        req->timeE[0] = GetTime();
18
19        m1.lock();
20        q2.push(req);
21        m1.unlock();
22    }
23 }
```

Листинг 3.4: вторая часть выполнения конвейера

```
1 void Conveyor::part2()
2 {
3     while(q2.size() == 0)
4         continue;
5
6     for (; ft2 < ntask; ft2++)
7     {
8         while(q2.size() == 0)
9             continue;
```

```

10
11     Request *req;
12
13     m1.lock ();
14
15     req = q2.front ();
16     q2.pop ();
17
18     m1.unlock ();
19
20     req->timeS [1] = GetTime ();
21     req->cryptXor ( 'p' );
22     req->timeE [1] = GetTime ();
23
24     m2.lock ();
25     q3.push (req );
26     m2.unlock ();
27 }
28 }

```

Листинг 3.5: третья часть выполнения конвейера

```

1 void Conveyor::part3 ()
2 {
3     while (q3.size () == 0)
4         continue;
5
6     for (; ft3 < ntask; ft3++)
7     {
8         while (q3.size () == 0)
9             continue;
10
11         Request *req;
12
13         m2.lock ();
14         req = q3.front ();

```

```

15         q3.pop();
16         m2.unlock();
17
18         req->timeS[2] = GetTime();
19         req->cryptXor('a');
20         req->timeE[2] = GetTime();
21
22         result.push_back(req);
23     }
24 }

```

Листинг 3.5: основной поток и генератор заявок

```

1 void Conveyor::run()
2 {
3     generateRequest();
4
5     std::thread t1 = std::thread(&Conveyor::part1, this);
6     std::thread t2 = std::thread(&Conveyor::part2, this);
7     std::thread t3 = std::thread(&Conveyor::part3, this);
8
9     t1.join();
10    t2.join();
11    t3.join();
12 }
13
14 void Conveyor::generateRequest()
15 {
16     for (int i = 0; i < ntask; i++)
17     {
18         Request *req = new Request(taskLen, i);
19         req->generateString();
20         startQ.push(req);
21     }
22 }

```

3.3. Результаты выполнения программы

Результатом работы программы является массив зашифрованных строк, которые перед входом в конвейер генерируются случайным образом (листинг 3.6), все строки равной длины. Также, в качестве результата программа выводит данные о времени начала и конца обработки каждой из заявок в 1, 2, и 3 этапах. Зная это, мы можем получить информацию о максимальном, минимальном и среднем времени в 2 и 3 очередях и в во всей системе.

Листинг 3.6: функция генерации случайных строк

```
1 void generateString()
2 {
3     srand(time(0));
4
5     for (int i = 0; i < len; i++)
6     {
7         dataStr.push_back(rand() % 26 + 97);
8     }
9 }
```

3.4. Оценка времени

В листинге 3.7 приведена функция, с помощью которой проводились замеры процессорного времени.

Листинг 3.7: основной поток и генератор заявок

```
1 double GetTime()
2 {
3     LARGE_INTEGER li;
4     !QueryPerformanceFrequency(&li);
5     double PCFreq = double(li.QuadPart);
6
7     QueryPerformanceCounter(&li);
8     return double(li.QuadPart) / PCFreq * 1000;
9 }
```

3.5. Вывод

Результатом технологической части стал выбор используемых технических средств реализации и последующая реализация алгоритмов и замера времени работы на языке C++.

4. Исследовательская часть

В данном разделе будут приведены результаты работы программы и последующий их анализ.

Эксперименты проводились на компьютере со следующими характеристиками:

- ОС - Windows 10, 64bit;
- Процессор - Intel Core i6 7300HQ 2.5GHz, 4 Core 8 Logical Processor;
- ОЗУ - 8Gb.

4.1. Результаты экспериментов

Замеры времени проводились на конвейере, обрабатывающем 200 заявок, каждая из которых содержит строку длиной 1000000 символов. Было замерено время 20 конвейеров, приведены усреднённые результаты.

На графиках 4.1 - 4.2 представлены результаты замеров времени во 2 и 3 очередях, и общее время, проведённое заявкой во всей системе, где t в миллисекундах.

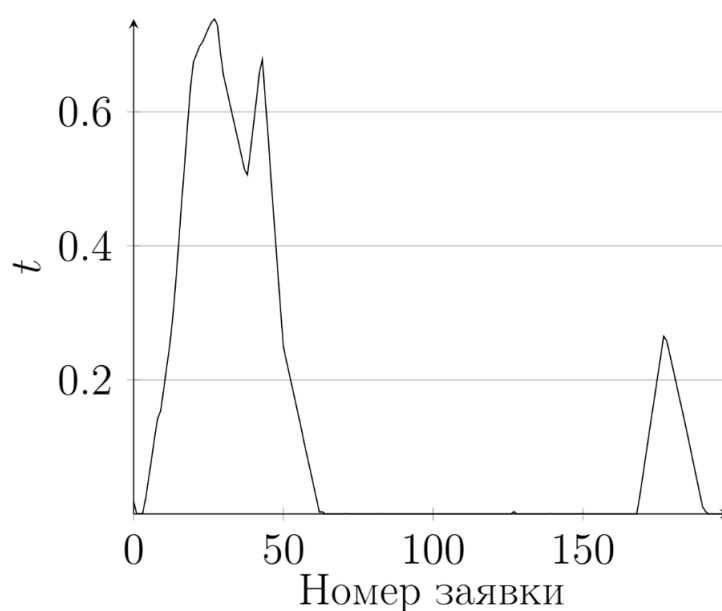


Рис. 4.1: время проведённое заявкой во второй очереди

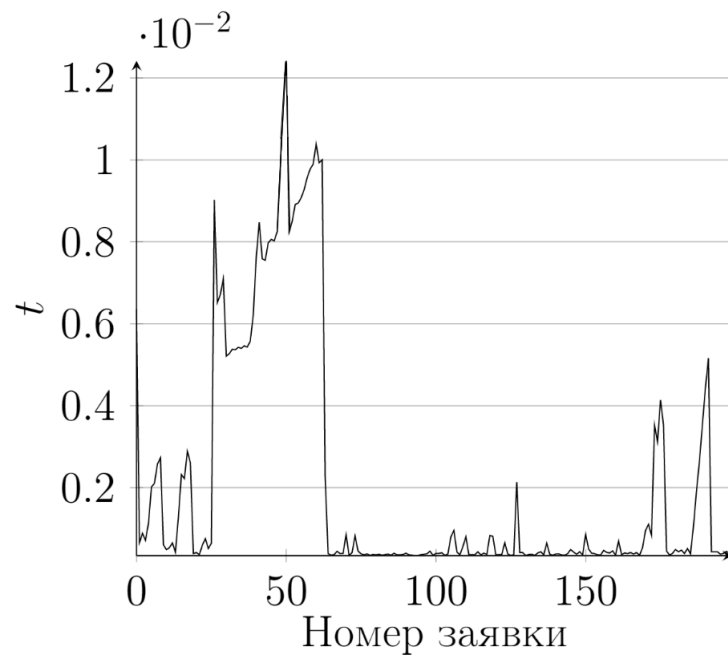


Рис. 4.2: время проведенное заявкой в третьей очереди

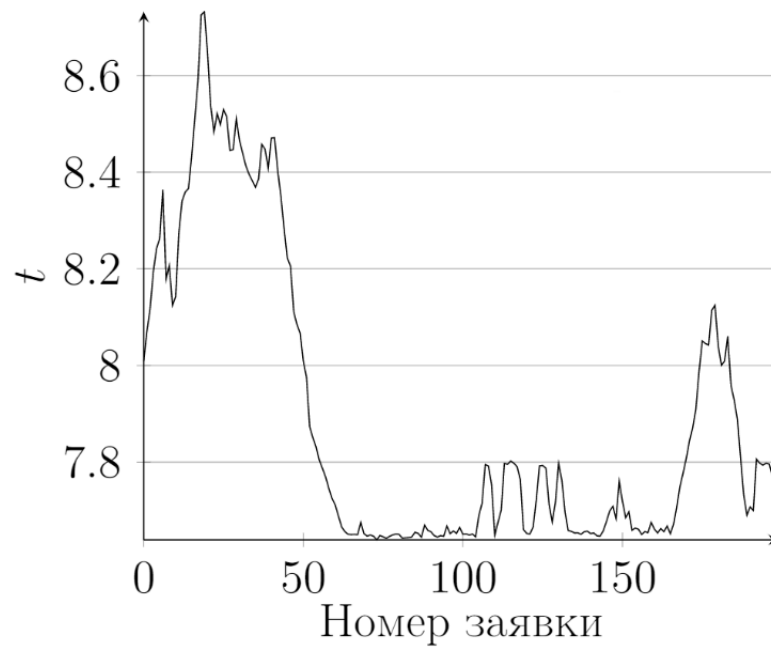


Рис. 4.3: время проведенное заявкой в конвейере

Из графиков видно, что время нахождения во второй очереди резко растёт только в начале обработки, но затем спадает и держится на низком уровне. В третьей очереди и всей системе наблюдается такая же картина, однако пики выше.

В следующей таблице 4.1 приведены минимальные, максимальные и средние результаты замеренного времени в миллисекундах, проведённого заявками в очередях и во всём конвейере.

Таблица 4.1: результаты проведённого заявками времени в очередях и системе

| | min | max | average |
|----------------|-------------------|-------|---------|
| Вторая очередь | $3 \cdot 10^{-4}$ | 0.739 | 0.140 |
| Третья очередь | $3 \cdot 10^{-3}$ | 0.013 | 0.002 |
| Система | 7.753 | 8.732 | 7.890 |

Можно сказать, что наибольшее время потрачено в ожидании поступления на конвейер, а значит первый этап является наиболее затратным по времени.

4.2. Вывод

В данном разделе были рассмотрены результаты работы программы. Из анализа стало ясно, что первый этап - шифрование с помощью шифра Цезаря замедляет работу всей системы, а также, что разница во времени работы 2 и 3 этапов крайне мала, что следует из малого времени, проведённого в третьей очереди.

Заключение

В ходе лабораторной работы достигнута поставленная цель: разработка и исследование конвейерных вычислений и использование их на практике. Также решены все поставленные задачи.

Стало ясно, что шифрование Цезаря занимает достаточно большую часть времени обработки заявки, в то время как два XOR шифра выполняются с равной скоростью. Также стало ясно, что разделение основной задачи на этапы даёт положительный результат на общем времени работы программы.

Список литературы

1. Дж. Макконнел. Анализ алгоритмов. Активный обучающий подход. – М.: Техносфера, 2017. – 267с.
2. Документация языка C++ 98 [Электронный ресурс], режим доступа: <http://www.open-std.org/JTC1/SC22/WG21/> (дата обращения 14.12.2020)