



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 09.03.04 Программная инженерия;

**О Т Ч Е Т**

по лабораторной работе № 5

Название: Конвейер

Дисциплина: Анализ алгоритмов

Студент

ИУ7-526

(Группа)

\_\_\_\_\_  
(Подпись, дата)

Кузин А.А.

(И.О. Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

Волкова Л.Л.

(И.О. Фамилия)

Москва, 2020

## СОДЕРЖАНИЕ

Введение .....	3
1 Аналитический раздел .....	4
1.1 Конвейер и конвейерная обработка .....	4
1.2 Шифрование строк .....	5
1.3 Вывод .....	5
2 Конструкторский раздел .....	6
2.1 Схемы алгоритмов .....	6
2.2 Конвейеризация задачи .....	7
2.3 Вывод .....	7
3 Технологический раздел .....	8
3.1 Средства реализации .....	8
3.2 Листинг кода .....	8
3.3 Собираемые данные .....	11
4 Исследовательский раздел .....	12
4.1 Анализ результатов .....	12
Заключение .....	15
Список использованных источников .....	16

## **ВВЕДЕНИЕ**

Цель работы: изучение возможностей конвейерных вычислений и использование такого подхода на практике.

При выполнении лабораторной работы поставлены такие задачи:

- 1) спроектировать ПО, реализующее конвейерную обработку;
- 2) описать реализацию ПО;
- 3) провести тестирование;

## 1 Аналитический раздел

В данном разделе будет представлено понятие конвейерных вычислений, рассмотрены способы шифрования строк.

### 1.1 Конвейер и конвейерная обработка

Конвейер — машина непрерывного транспорта, предназначенная для перемещения сыпучих, кусковых или штучных грузов [1].

Конвейеризация в обобщённом смысле базируется на разделении выполняемой операции на более мелкие составляющие, которые называются подфункциями, и предоставлении для выполнения каждой подфункции своего аппаратного блока [2].

Вычислительный конвейер предполагает перемещение команд или данных по этапам цифрового вычислительного конвейера со скоростью, не зависящей от протяжённости конвейера (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 Шифрование строк

Шифрование — это преобразование информации, делающее ее нечитаемой для посторонних. При этом доверенные лица могут провести дешифрование и прочитать исходную информацию. Существует множество способов шифрования/дешифрования, но секретность данных основана не на тайном алгоритме, а на том, что ключ шифрования (пароль) известен только доверенным лицам [3].

Существуют разные алгоритмы шифрования [4].

- Шифр Атбаша – алфавит зеркально отражается, то есть "А" становится "Z" и так далее.

- Шифр Цезаря – имеется 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 разбиваем на отдельные символы и каждый символ представляем в бинарном виде. После чего по-символьно применим операцию XOR с ключом, в результате чего получается зашифрованное сообщение.

- Шифр кодового слова – выбирается ключевое слово, все неповторяющиеся буквы из него выписываются в начало алфавита, остальной алфавит сдвигается.

## 1.3 Вывод

В данном разделе было представлено понятие конвейерных вычислений, рассмотрены способы шифрования строк.

## 2 Конструкторский раздел

В данном разделе будет рассмотрена схема реализованного алгоритма шифрования и описан способ его конвейеризации.

### 2.1 Схемы алгоритмов

В программе реализовано шифрование строк используя шифр цезаря и XOR-шифр, на рисунках 2.1 и 2.2 представлены схемы выбранных алгоритмов шифрования.

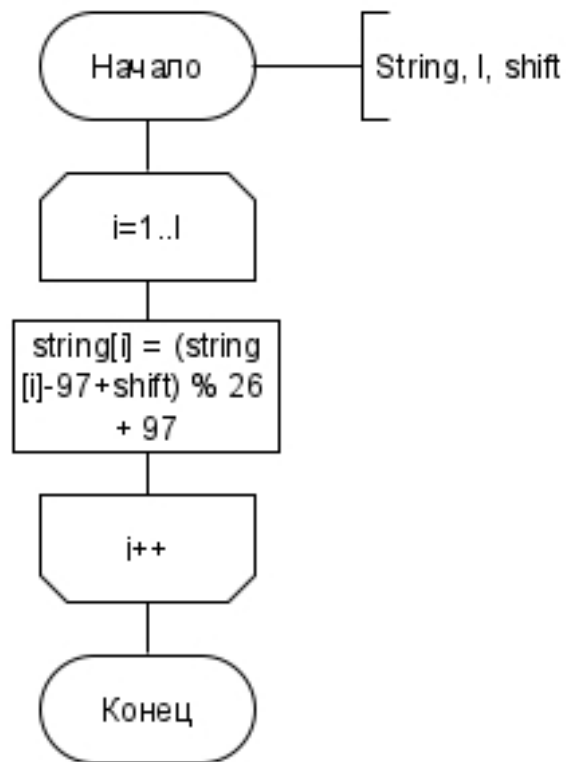


Рисунок 2.1 — Алгоритм шифрования Цезаря

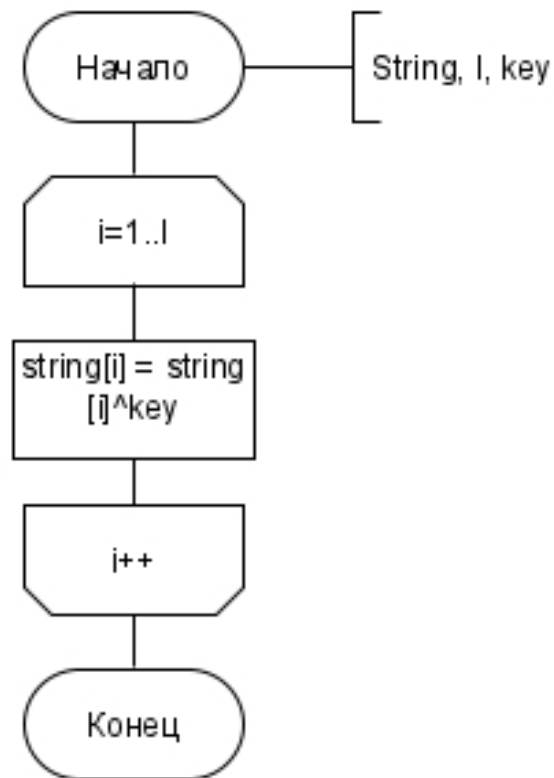


Рисунок 2.2 — Алгоритм XOR-шифрования

## 2.2 Конвейеризация задачи

Алгоритм шифрования с программы разбит на 3 этапа: сначала к строке применяется шифр Цезаря, затем XOR-шифр и снова шифр Цезаря. 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

В данном разделе была рассмотрены схемы алгоритмов шифра Цезаря и XOR-шифра, описана схема конвейеризации поставленной задачи.

### 3 Технологический раздел

В данном разделе будут представлены листинги кода реализованных алгоритмов и дано описание получаемых результатов работы программы.

#### 3.1 Средства реализации

В данной работе используется язык программирования C++. Среда разработки Visual Studio Code. Для замера процессорного времени используется функция QueryPerformanceCounter из библиотеки windows.h. Параллельные вычисления выполняются с использованием библиотеки `<thread>` [5], а также применяются mutex из библиотеки `<mutex>`.

Замеры времени были произведены на: Intel(R) Core(TM) i5-8250U CPU @1.60GHz 1.80 Ghz, 4 ядра, 8 логических процессоров.

#### 3.2 Листинг кода

В листинге 3.1 приведен код алгоритма XOR-шифрования.

Листинг 3.1 — Алгоритм XOR-шифра

```
1 int xorCrypt(char *string, int l, bitset<8> keyToEncrypt)
2 {
3     for (int i = 0; i < l; i++)
4         string[i] ^= keyToEncrypt.to_ulong();
5     return 0;
6 }
```

В листинге 3.2 представлен код алгоритма шифра Цезаря.

Листинг 3.2 — Алгоритм шифра Цезаря

```
1 int caesarCrypt(char *string, int l, int shift)
2 {
3     for (int i = 0; i < l; i++)
4         string[i] = (string[i] - 97 + shift) % 26 + 97;
5     return 0;
6 }
```

В листингах 3.3-3.5 представлен код 3 этапов выполнения конвейера.

Листинг 3.3 — Первый этап выполнения конвейера



```

1 void Conveyor::part1()
2 {
3     while (this->ft1 < this->taskNumber)
4     {
5         Request *req;
6         if (this->startingQ.size())
7         {
8             req = this->startingQ.front();
9             this->startingQ.pop();
10        }
11        else
12            continue;
13
14        req->s1 = GetTimestamp();
15
16        caesarCrypt(req->string, req->l, 12);
17
18        req->e1 = GetTimestamp();
19
20        this->ft1++;
21
22        this->m1.lock();
23        this->q2.push(req);
24        this->m1.unlock();
25    }
26 }

```

Листинг 3.4 — Второй этап выполнения конвейера

```

1 void Conveyor::part2()
2 {
3     while (this->q2.size() == 0)
4         continue;
5
6     while (this->ft2 < this->taskNumber)
7     {
8         while (this->q2.size() == 0)
9             continue;
10
11        Request *req;
12
13        this->m1.lock();
14        req = this->q2.front();
15        this->q2.pop();
16        this->m1.unlock();
17

```

```

18         req->s2 = GetTimestamp();
19         bitset<8> key('e');
20
21         xorCryt(req->string, req->l, key);
22         req->e2 = GetTimestamp();
23
24         this->ft2++;
25
26         this->m2.lock();
27         this->q3.push(req);
28         this->m2.unlock();
29     }
30 }

```

Листинг 3.5 — Третий этап выполнения конвейера

```

1 void Conveyor::part3()
2 {
3     while (this->q3.size() == 0)
4         continue;
5
6     while (this->ft3 < this->taskNumber)
7     {
8         while (this->q3.size() == 0)
9             continue;
10
11         Request *req;
12         this->m2.lock();
13         req = this->q3.front();
14         this->q3.pop();
15         this->m2.unlock();
16
17         req->s3 = GetTimestamp();
18
19         bitset<8> key('k');
20         xorCryt(req->string, req->l, key);
21         req->e3 = GetTimestamp();
22
23         this->ft3++;
24         this->result.push_back(req);
25     }
26 }

```

В листинге 3.6 представлен код основного потока и генератора заявок.

Листинг 3.6 — Третий этап выполнения конвейера

```

1 void Conveyor::run()
2 {
3     generateRequests();
4
5     thread t1 = thread(&Conveyor::part1, this);
6     thread t2 = thread(&Conveyor::part2, this);
7     thread t3 = thread(&Conveyor::part3, this);
8
9     t1.join();
10    t2.join();
11    t3.join();
12 }
13
14 void Conveyor::generateRequests()
15 {
16     for (int i = 0; i < taskNumber; i++)
17     {
18         Request *req = new Request(taskLength);
19         req->number = i;
20         stringGenerator(req->string, req->l);
21         startingQ.push(req);
22     }
23     cout << startingQ.size();
24 }

```

### 3.3 Собираемые данные

Результатом работы программы является массив зашифрованных строк, создаваемых случайно, все строки равной длины и данные о времени начала и конца обработки каждой из заявок в 1, 2 и 3 этапах. Зная эту информацию, мы можем получить информацию о максимальном, минимальном, среднем времени во 2 и 3 очередях и в системе вообще.

#### Вывод.

В данном разделе были рассмотрены листинги кода реализованных алгоритмов и конвейера, описаны результаты работы программы.

## 4 Исследовательский раздел

В данном разделе будут приведены результаты работы программы и сделаны выводы.

### 4.1 Анализ результатов

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

На графиках 4.1 и 4.2 представлено время проведённое каждой заявкой во 2-ой и 3-ей очередях.

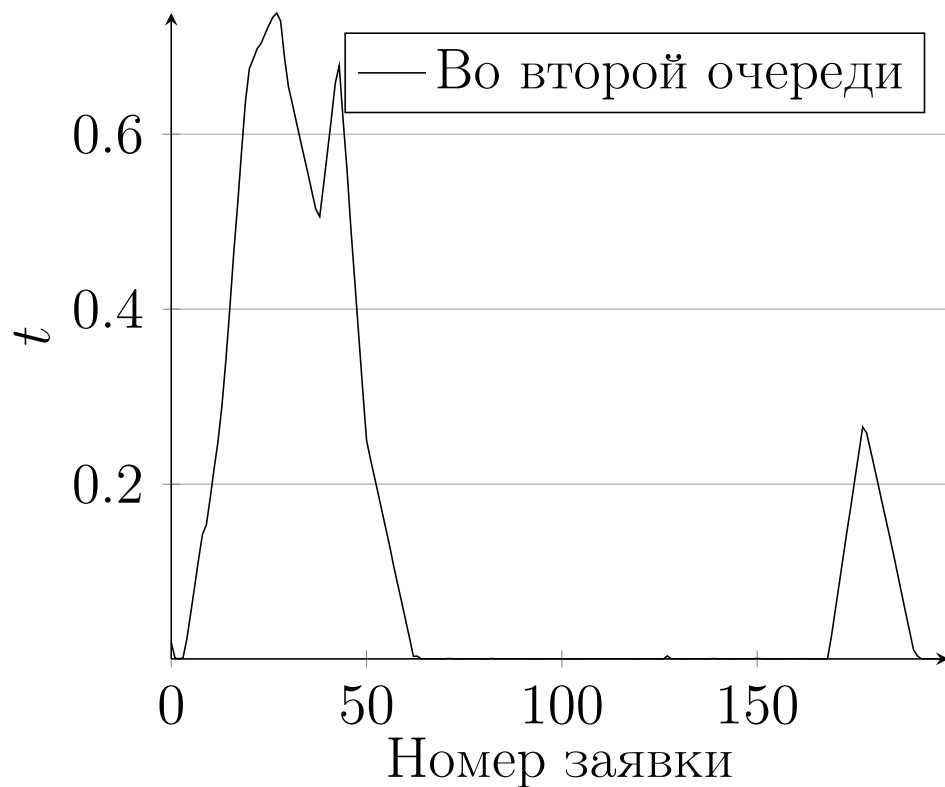


Рисунок 4.1 — Время проведённое заявкой во 2-ой и 3-ей очередях

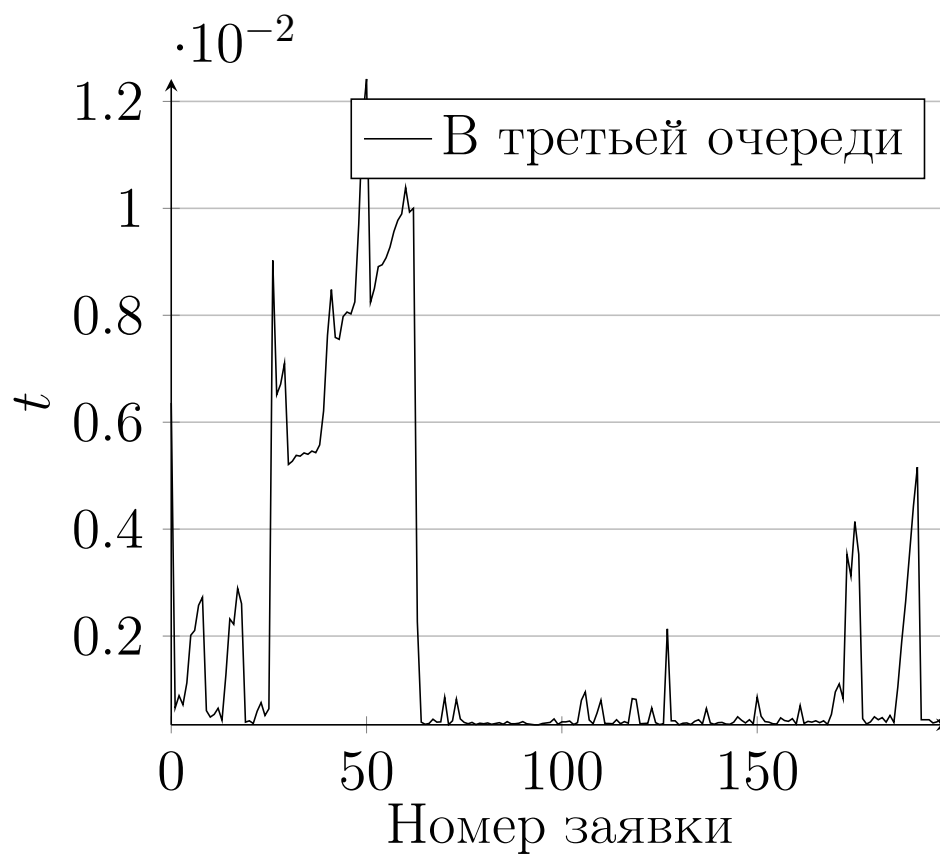


Рисунок 4.2 — Время проведённое заявкой в 3-ей очереди

На графике 4.3 представлено время, проведённое каждой заявкой во всей системе.

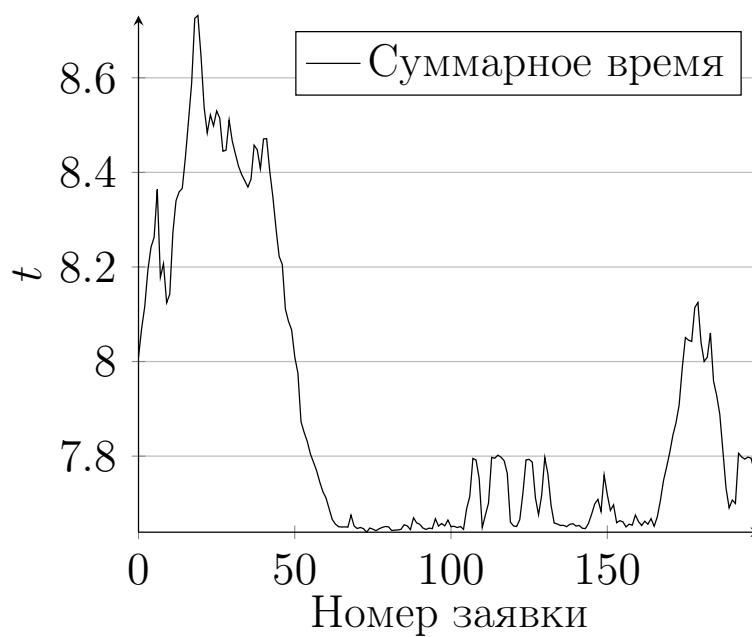


Рисунок 4.3 — Время проведённое в конвейере

Из графиков видно, что время нахождения во 2-ой очереди резко растёт только в начале обработки, затем спадает и держится на низком уровне. В третьей очереди и всей системе в общем ситуация схожа, однако пики выше. В таблице 4.1 представлен минимальные, максимальные и средние времени, проведенного заявками в очередях и системе вообще.

	min	max	avg
2-ая оч.	0.000295	0.73859	0.14
3-я оч.	0.00034	0.01242	0.0021
Система	7.7533	8.73173	7.89

Таблица 4.1 — Анализ данных

В результате, можно сказать, что наибольшее время потрачено в ожидании поступления на конвейер, а значит первый этап является наиболее затратным по времени, что замедляет всю работу.

### **Вывод**

В данном разделе были рассмотрены результаты работы программы, стало ясно, что первый этап - шифрование шифром Цезаря замедляет работу всей системы, а также, что разница во времени работы 2 и 3 этапов крайне низка, что следует из низкого времени, проведенного в 3-ей очереди.

## ЗАКЛЮЧЕНИЕ

В результате лабораторной работы цель была достигнута – изучены возможности конвейерных вычислений и данный подход был использован на практике. Все задачи были выполнены.

Стало ясно, что шифрование Цезаря занимает 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 ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Конвейер (значения)[Электронный ресурс]. — 2010. — Режим доступа: <https://dic.academic.ru/dic.nsf/ruwiki/141054> (дата обращения: 29.11.2020).
2. Конвейеризация как средство повышения производительности ЭВМ[Электронный ресурс]. — 2020. — Режим доступа: [https://spravochnick.ru/informatika/konveyerizaciya\\_kak\\_sredstvo\\_povysheniya\\_proizvoditelnosti\\_evm/](https://spravochnick.ru/informatika/konveyerizaciya_kak_sredstvo_povysheniya_proizvoditelnosti_evm/) (дата обращения: 29.11.2020).
3. Malenkovich Serge. Зачем нужно шифрование?[Электронный ресурс]. — 2013. — Режим доступа: <https://www.kaspersky.ru/blog/encryption-reasons/879/> (дата обращения: 29.11.2020).
4. malkoran. Элементарные шифры на понятном языке[Электронный ресурс]. — 2019. — Режим доступа: <https://habr.com/ru/post/444176/> (дата обращения: 29.11.2020).
5. Varun. C++11 Multithreading [Электронный ресурс]. — 2015. — Режим доступа: <https://thispointer.com/c-11-multithreading-part-1-three-different-ways-to-create-threads/> (дата обращения: 2.10.2020).