



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 Пискунов П.

Группа ИУ7-56Б

Преподаватель Волкова Л.Л., Строганов Д.В.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Аналитическая часть	4
1.1 Конвейерная обработка данных	4
1.2 Описание задачи	4
2 Конструкторская часть	6
2.1 Требования к программному обеспечению	6
2.2 Разработка алгоритмов	6
3 Технологическая часть	12
3.1 Средства реализации	12
3.2 Реализация алгоритмов	12
4 Исследовательская часть	17
4.1 Интерфейс приложения	17
4.2 Технические характеристики	17
4.3 Временные характеристики	17
4.4 Вывод	19
ЗАКЛЮЧЕНИЕ	20
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	21

ВВЕДЕНИЕ

При обработке данных могут возникать ситуации, когда один набор данных необходимо обработать последовательно несколькими алгоритмами. В таком случае удобно использовать конвейерную обработку данных, что позволяет на каждой следующей «линии» конвейера использовать данные, полученные с предыдущего этапа [1].

Помимо линейной конвейерной обработки данных, существуют параллельная конвейерная обработка данных. При таком подходе все линии работают с меньшим времени простоя по сравнению с линейным вариантом, так как могут обрабатывать задачи независимо от других линий.

Целью данной лабораторной работы является исследование параллельной и последовательной реализации конвейерной обработки данных.

В рамках выполнения работы необходимо решить следующие задачи:

- описать конвейерную обработку данных;
- реализовать параллельную и линейную версию конвейерных вычислений;
- исследовать зависимость времени выполнения от количества задач для параллельной и линейной обработки конвейера;
- сделать выводы на основе проделанной работы.

1 Аналитическая часть

В данном разделе представлены теоретические сведения о рассматриваемых алгоритмах.

1.1 Конвейерная обработка данных

Конвейер — система поточного производства. В терминах программирования ленты конвейера представлены функциями, выполняющими над неким набором данных операции и предающие их на следующую ленту конвейера. Моделирование конвейерной обработки хорошо сочетается с технологией многопоточного программирования — под каждую ленту конвейера выделяется отдельный поток, все потоки работают в асинхронном режиме [2].

1.2 Описание задачи

В качестве примера для конвейерной обработки будет обрабатываться текстовый файл. Всего будет использовано три ленты, выполняющих следующие функции:

- извлечение строки(текста) из файла;
- поиск всех вхождений подстроки в строку по алгоритму Бойера—Мура;
- сохранение результатов поиска в отдельный файл.

Ленты конвейера (обработчики) будут передавать друг другу заявки, каждая из которых будет содержать:

- имя файла, содержащего строку;
- строку из файла;
- искомую подстроку;
- результаты поиска подстроки в строке.

Вывод

В данном разделе были рассмотрены особенности построения конвейерных вычислений.

2 Конструкторская часть

В данном разделе будут представлены требования к программному обеспечению и схемы рассматриваемых алгоритмов.

2.1 Требования к программному обеспечению

К программе предъявлены ряд требований:

- наличие меню для выбора запускаемого режима работы конвейера — последовательного или параллельного;
- предоставление интерфейса для ввода числа заявок;
- формирование файла с результатами журналирования работы конвейера.

2.2 Разработка алгоритмов

На рисунке 2.1 представлен последовательный алгоритм работы стадий конвейера.

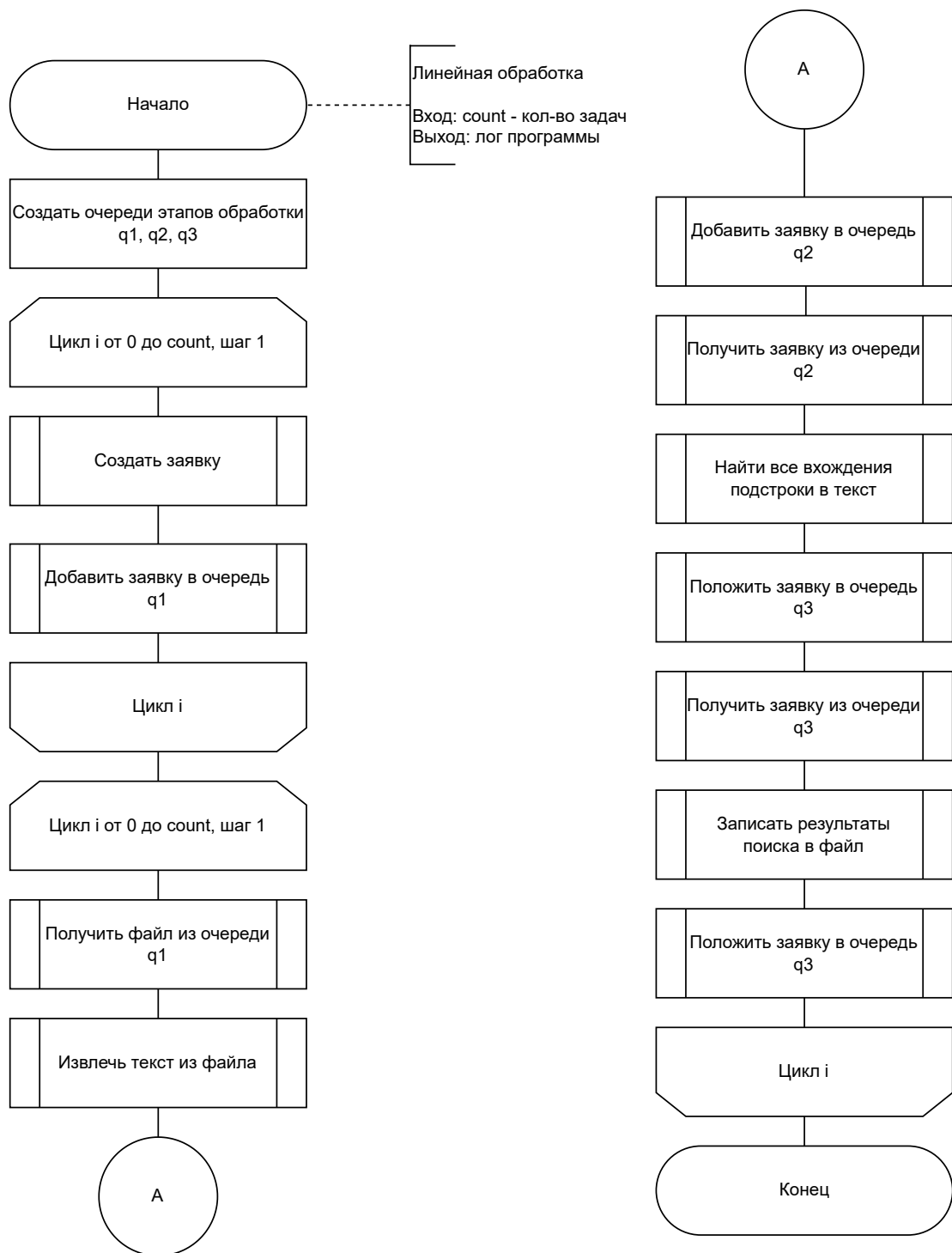


Рисунок 2.1 – Схема алгоритма последовательной конвейерной обработки

Параллельная работа будет реализована с помощью добавления 3-х вспомогательных потоков, где каждый поток отвечает за свою стадию обработки.

На рисунке 2.2 представлена схема главного потока при параллельной работе стадий конвейера.

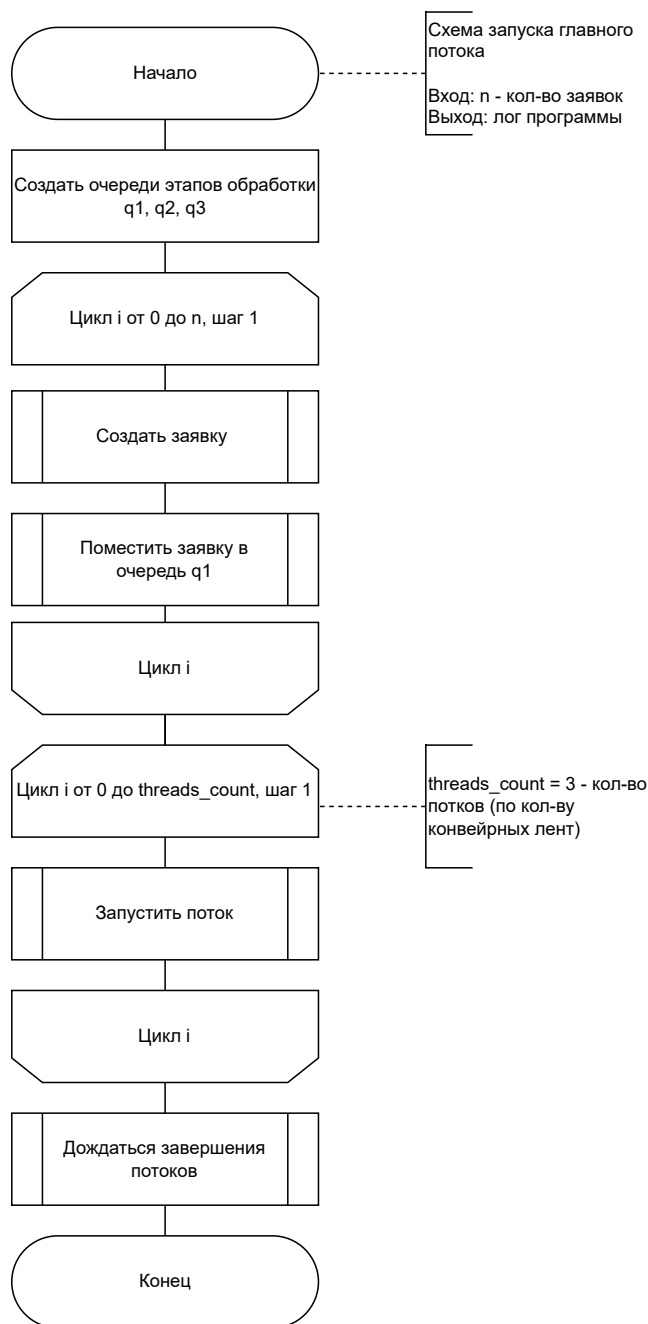


Рисунок 2.2 – Схема главного потока при параллельной работе конвейера

На рисунках 2.3, 2.4 и 2.5 представлены схемы алгоритмов каждого из обработчиков (потоков) при параллельной работе.

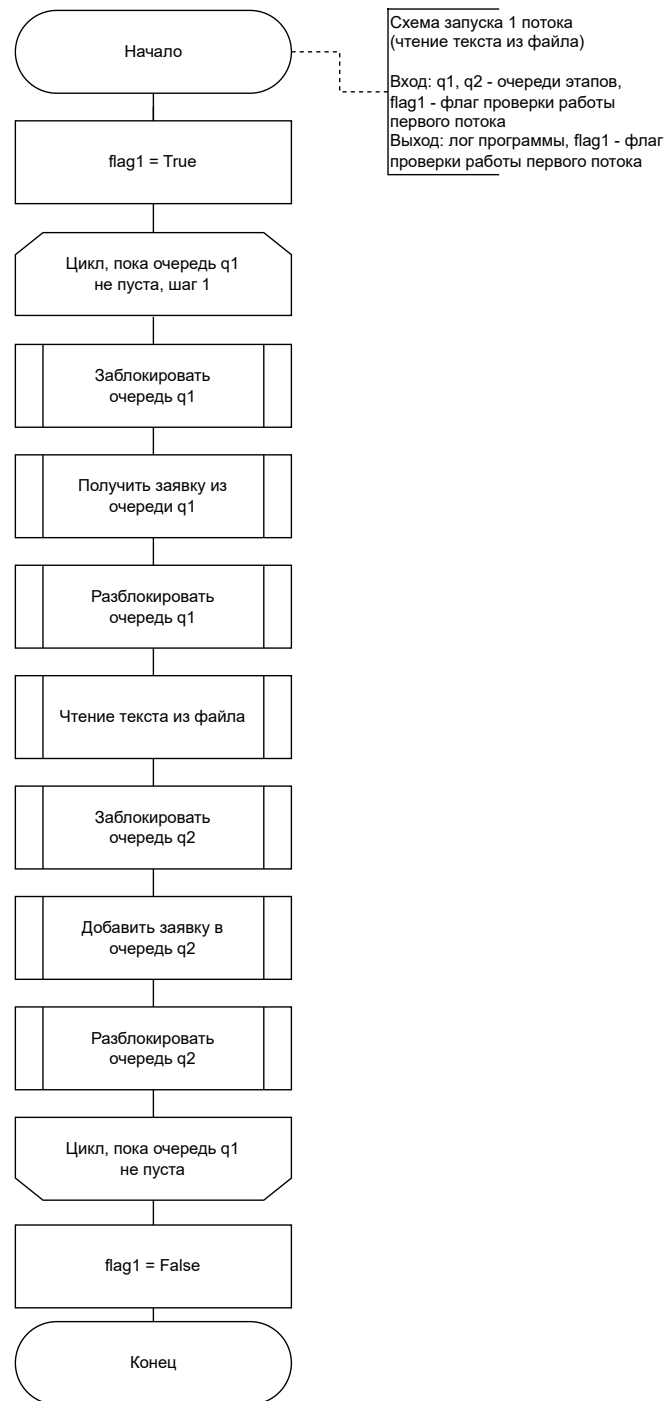


Рисунок 2.3 – Схема алгоритма потока 1

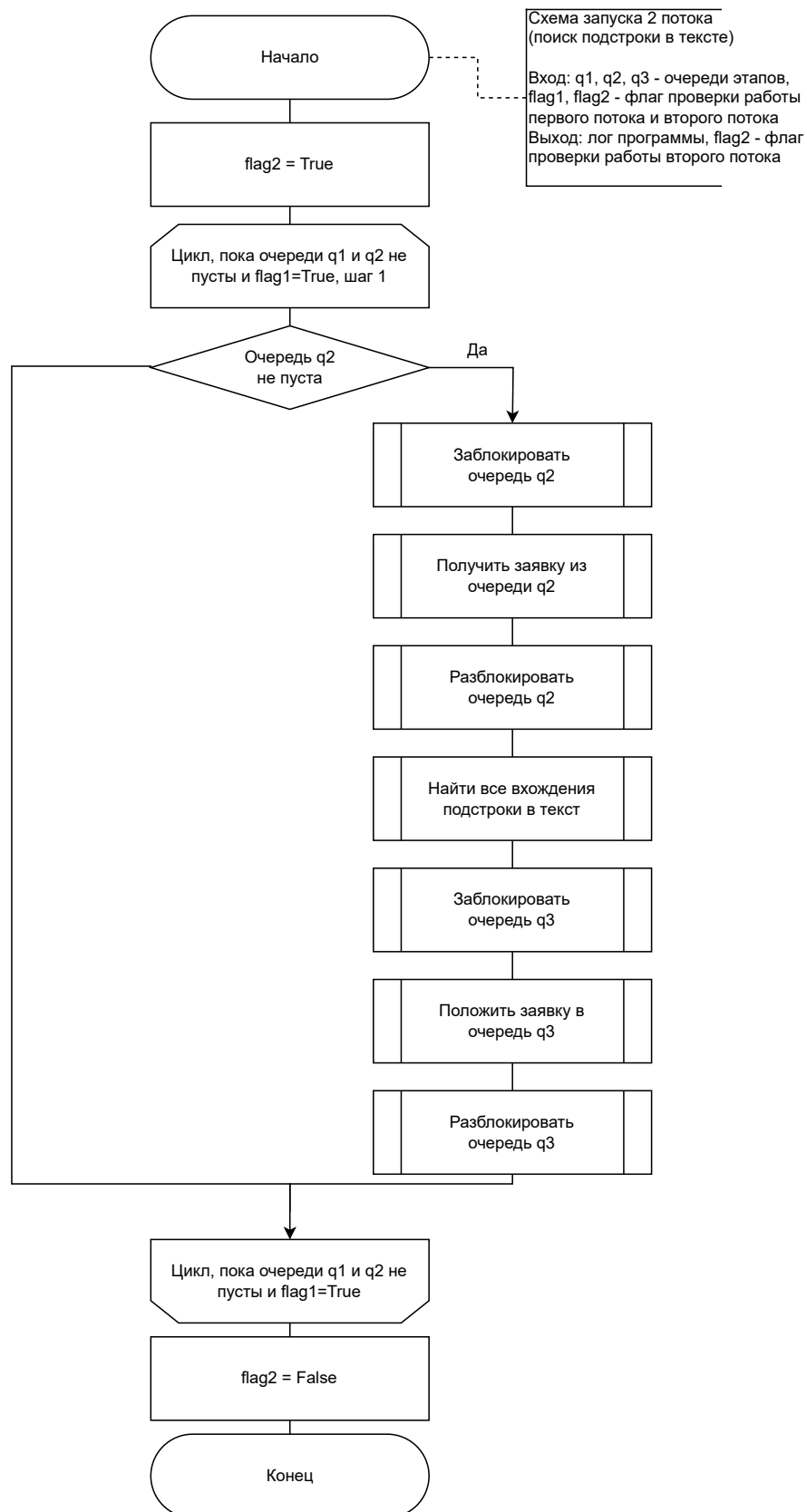


Рисунок 2.4 – Схема алгоритма потока 2



Рисунок 2.5 – Схема алгоритма потока 3

Вывод

В данном разделе были представлены схемы последовательной и параллельной работы стадий конвейера.

3 Технологическая часть

В этом разделе предоставляются листинги реализованных алгоритмов и осуществляется выбор средств реализации.

3.1 Средства реализации

Для реализации программного обеспечения был выбран язык программирования C++ [3]. Данный выбор обусловлен наличием у языка встроенной библиотекой измерения процессорного времени и предоставлением возможности:

- работа с потоками ядра предоставляется классом *thread* [4];
- работа с очередями предоставляется классом *queue* [5].

3.2 Реализация алгоритмов

На листинге 3.1 представлены используемые типы очередей.

Листинг 3.1 – Реализация последовательной конвейерной обработки

```
1 std::queue<std::shared_ptr<Report>> q1;  
2 std::queue<std::shared_ptr<Report>> q2;  
3 std::queue<std::shared_ptr<Report>> q3;
```

На листинге 3.2 представлена последовательная реализация конвейерной обработки.

Листинг 3.2 – Реализация последовательной конвейерной обработки

```
1  void Conveyor::run_linear(size_t reports_cnt, bool log_fl)
2  {
3      for (size_t i = 0; i < reports_cnt; i++)
4      {
5          std::string n = Conveyor::names[rand() %
6 (names.size())];
7          std::shared_ptr<Report> new_report(new Report(n));
8          reports.push_back(new_report);
9          q1.push(new_report);
10     }
11     for (size_t i = 0; i < reports_cnt; i++)
12     {
13         std::shared_ptr<Report> report = q1.front();
14         if (log_fl) report->get_text(i + 1);
15         else report->get_text_meassure(i + 1);
16         q2.push(report);
17         q1.pop();
18         report = q2.front();
19         if (log_fl) report->find_subst(i + 1);
20         else report->find_subst_meassure(i + 1);
21         q3.push(report);
22         q2.pop();
23         report = q3.front();
24         if (log_fl) report->output_res(i + 1);
25         else report->output_res_meassure(i + 1);
26         q3.pop();
27     }
```

На листинге 3.3 представлена реализация главного потока при параллельной работе конвейера.

Листинг 3.3 – Реализация основного потока для конвейерной обработки

```
1  void Conveyor::run_parallel(size_t reports_cnt, bool log_fl)
2  {
3      for (size_t i = 0; i < reports_cnt; i++)
4      {
5          std::string n = Conveyor::names[rand() %
6 (names.size())];
7          std::shared_ptr<Report> new_report(new Report(n));
8          reports.push_back(new_report);
9          q1.push(new_report);
10     }
11     if (log_fl)
12     {
13         this->threads[0] = std::thread(&Conveyor::get_text,
14 this);
15         this->threads[1] = std::thread(&Conveyor::find_subst,
16 this);
17         this->threads[2] = std::thread(&Conveyor::output_res,
18 this);
19     }
20     else
21     {
22         this->threads[0] =
23 std::thread(&Conveyor::get_text_measure, this);
24         this->threads[1] =
25 std::thread(&Conveyor::find_subst_measure, this);
26         this->threads[2] =
27 std::thread(&Conveyor::output_res_measure, this);
28     }
29     for (int i = THRD_CNT-1; i >=0; i--)
30     {
31         this->threads[i].join();
32     }
33 }
```

На листинге 3.4 представлена реализация алгоритма извлечения текста из файла.

Листинг 3.4 – Реализация вспомогательного потока

```
1  void Conveyor::get_text()
2  {
3      size_t task_num = 0;
4      while (!this->q1.empty())
5      {
6          std::shared_ptr<Report> report = q1.front();
7          q1.pop();
8          report->get_text(++task_num);
9          mutex_q2.lock();
10         q2.push(report);
11         mutex_q2.unlock();
12     }
13 }
```

На листинге 3.5 представлена реализация алгоритма работы 2 рабочего потока, который выполняет поиск вхождений подстроки в строку.

Листинг 3.5 – Реализация вспомогательного потока

```
1  void Conveyor::find_subst()
2  {
3      size_t task_num = 0;
4      do
5      {
6          if (!this->q2.empty())
7          {
8              mutex_q2.lock();
9              std::shared_ptr<Report> report = q2.front();
10             mutex_q2.unlock();
11             report->find_subst(++task_num);
12             mutex_q3.lock();
13             q3.push(report);
14             mutex_q3.unlock();
15             mutex_q2.lock();
16             q2.pop();
17             mutex_q2.unlock();
18         }
19     } while (!q1.empty() || !q2.empty());
20 }
```

На листинге 3.6 представлена реализация алгоритма сохранения результатов поиска в файл.

Листинг 3.6 – Реализация вспомогательного потока

```
1  void Conveyor::output_res()  
2  {  
3      size_t task_num = 0;  
4      do  
5      {  
6          if (!this->q3.empty())  
7          {  
8              mutex_q3.lock();  
9              std::shared_ptr<Report> report = q3.front();  
10             mutex_q3.unlock();  
11             report->output_res(++task_num);  
12             mutex_q3.lock();  
13             q3.pop();  
14             mutex_q3.unlock();  
15         }  
16     } while (!q1.empty() || !q2.empty() || !q3.empty());  
17 }
```

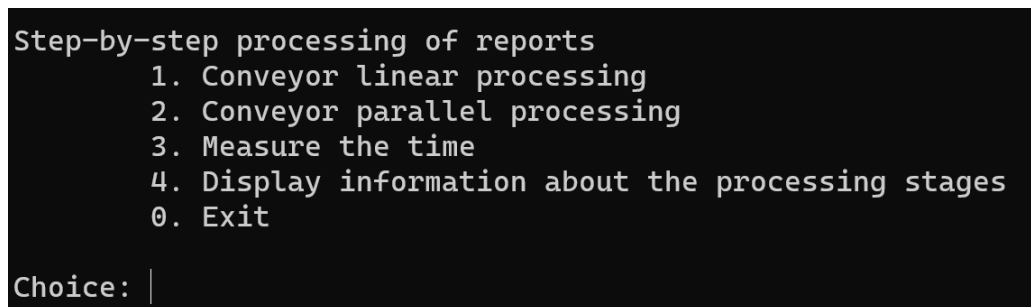
Вывод

В данном разделе была разработана реализация конвейерных вычислений. Также в данном разделе была приведена информация о выбранных средствах для разработки алгоритмов.

4 Исследовательская часть

4.1 Интерфейс приложения

На рисунке 4.1 представлен интерфейс приложения.



```
Step-by-step processing of reports
  1. Conveyor linear processing
  2. Conveyor parallel processing
  3. Measure the time
  4. Display information about the processing stages
  0. Exit

Choice: |
```

Рисунок 4.1 – Интерфейс приложения

4.2 Технические характеристики

Технические характеристики устройства:

- операционная система — Windows 11 Pro 64 – разрядная система [6];
- оперативная память — 16 Гбайт;
- процессор — 11th Gen Intel(R) Core(TM) i7-1165G7 с тактовой частотой 2.8 ГГц.

4.3 Временные характеристики

В таблице 4.1 приведено сравнение времени выполнения параллельной обработки данных в зависимости от количества входных задач. Линия №1 — чтение файла, линия №2 — поиск подстроки в тексте файла алгоритмом Бойера — Мура, линия №3 — запись результата в файл. Время указано в секундах.

Таблица 4.1 – Таблица времени выполнения параллельной обработки данных, время в секундах

К-во задач	Линия №1, с	Линия №2, с	Линия №3, с	Время работы, с
50	0.03	0.16	0.01	0.27
100	0.06	0.34	0.02	0.47
200	0.13	0.63	0.06	0.90
400	0.30	1.32	0.15	1.86
800	0.63	2.45	0.31	3.45

На рисунке 4.2 представлен график зависимости времени от количества задач для линейной и параллельной обработки конвейера.

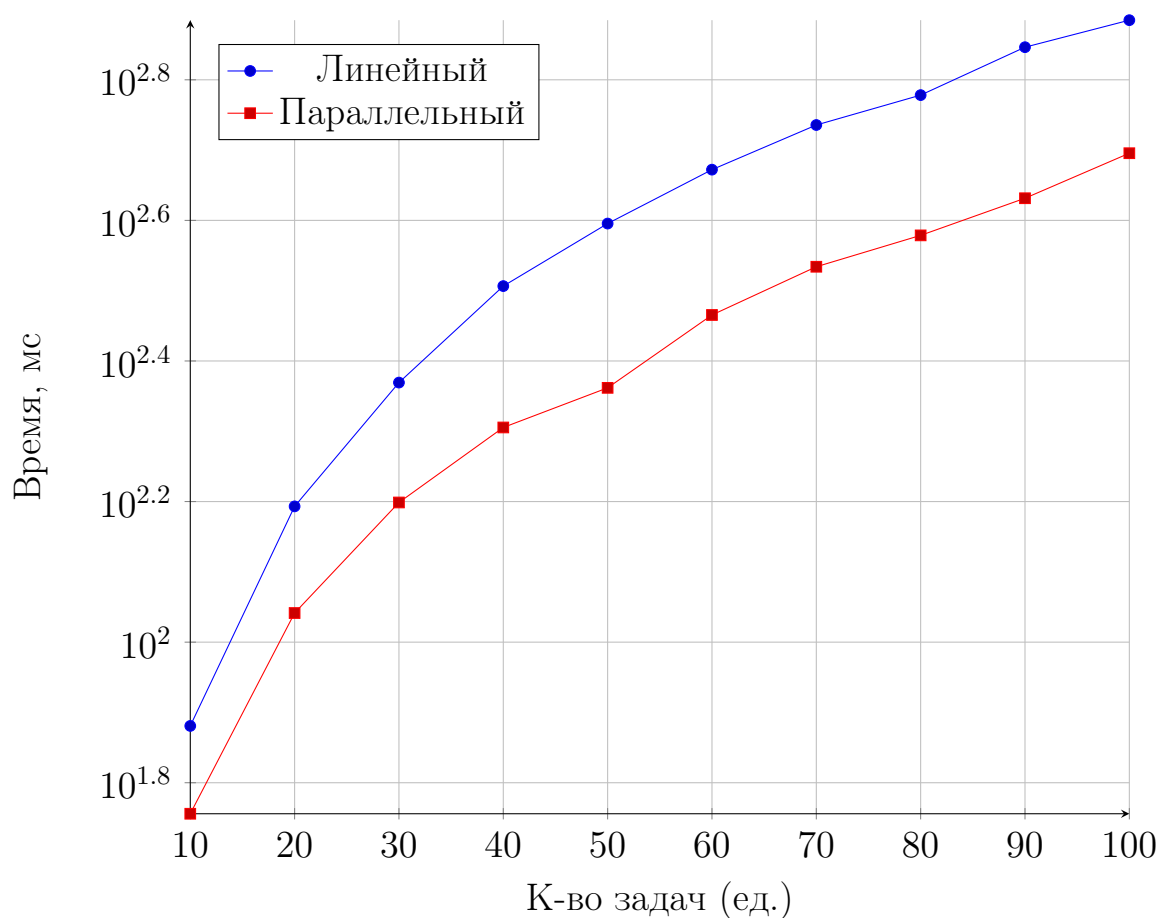


Рисунок 4.2 – Зависимость времени работы реализации конвейеров от количества задач

4.4 Вывод

В исследовательском разделе были приведены результаты замеров зависимости времени выполнения от количества задач для параллельной и линейной обработки конвейера.

В результате исследования было получено, что параллельная реализация конвейерной обработки данных выполняется быстрее, чем последовательная приблизительно в 1.1 раза.

ЗАКЛЮЧЕНИЕ

Поставленная цель достигнута — исследованы параллельная и последовательная реализации конвейерной обработки данных.

В ходе выполнения лабораторной работы были решены все задачи:

- описана конвейерная обработка данных;
- реализованы параллельная и линейная версии конвейерных вычислений;
- исследована зависимость времени выполнения от количества задач для параллельной и линейной обработки конвейера;
- сделаны выводы на основе проделанной работы.

В результате исследования было получено, что параллельная реализация конвейерной обработки данных выполняется быстрее, чем последовательная приблизительно в 1.1 раза.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Конвейерные вычисления: определение и особенности [Электронный ресурс]. — Режим доступа: <https://xn--870-iddfg5dar7d.xn--p1ai/faq/konveiernye-vycisleniya-opredelenie-i-osobennosti> (дата обращения: 11.02.2024).
- 2 Конвейерные вычисления [Электронный ресурс]. — Режим доступа: <https://helpdoma.ru/faq/cto-takoe-konveiernye-vycisleniya> (дата обращения: 30.01.2024).
- 3 C++ Standard [Электронный ресурс]. — Режим доступа: <https://isocpp.org/> (дата обращения: 11.02.2024).
- 4 Класс thread [Электронный ресурс]. — Режим доступа: <https://learn.microsoft.com/ru-ru/cpp/standard-library/thread-class> (дата обращения: 11.02.2024).
- 5 Класс queue [Электронный ресурс]. — Режим доступа: [:https://learn.microsoft.com/ru-ru/cpp/standard-library/queue-class](https://learn.microsoft.com/ru-ru/cpp/standard-library/queue-class) (дата обращения: 11.02.2024).
- 6 Windows 11 Pro [Электронный ресурс]. — Режим доступа: <https://www.microsoft.com/en-us/windows/business/windows-11-pro#windows11security> (дата обращения: 30.01.2024).