



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 № 4

по курсу «Анализ алгоритмов»

на тему: «Параллельные вычисления на основе нативных потоков»

Студент ИУ7-54Б
(Группа)

(Подпись, дата)

Булдаков М.
(И. О. Фамилия)

Преподаватель

(Подпись, дата)

Волкова Л. Л.
(И. О. Фамилия)

2023 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	3
1 Аналитический раздел	4
1.1 Многопоточность	4
1.2 Исправления орфографических ошибок в тексте	4
1.3 Использование потоков для исправления орфографических оши- бок	5
2 Конструкторский раздел	6
2.1 Требования к программному обеспечению	6
2.2 Описание используемых типов данных	6
2.3 Разработка алгоритмов	7
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	10

ВВЕДЕНИЕ

С развитием вычислительных систем появилась потребность в параллельной обработке данных для повышения эффективности систем, ускорения вычислений и более рационального использования имеющихся ресурсов. Благодаря совершенствованию процессоров стало возможно использовать их для выполнения множества одновременных задач, что привело к появлению понятия «многопоточность» [1].

Цель данной лабораторной работы — описать принципы параллельных вычислений на основе нативных потоков для исправления орфографических ошибок в тексте. Для достижения поставленной цели необходимо выполнить следующие задачи:

- описать алгоритм исправления орфографических ошибок в тексте;
- спроектировать программное обеспечение, реализующее алгоритм и его параллельную версию;
- выбрать инструменты для реализации и замера процессорного времени выполнения реализаций алгоритмов;
- проанализировать затраты реализаций алгоритмов по времени.

1 Аналитический раздел

В данном разделе будет представлена информация о многопоточности и исследуемом алгоритме исправления орфографических ошибок в тексте.

1.1 Многопоточность

Многопоточность [2] — это способность центрального процессора одновременно выполнять несколько потоков, используя ресурсы одного процессора. Каждый поток представляет собой последовательность инструкций, которые могут выполняться параллельно с другими потоками, созданными одним и тем же процессом.

Процессом называют программу в стадии выполнения [3]. Один процесс может иметь один или несколько потоков. Поток — это часть процесса, которая выполняет задачи, необходимые для выполнения приложения. Процесс завершается, когда все его потоки полностью завершены.

Одной из сложностей, связанных с использованием потоков, является проблема доступа к данным. Основным ограничением является невозможность одновременной записи в одну и ту же ячейку памяти из разных потоков. Это означает, что нужен механизм синхронизации доступа к данным, так называемый “мьютекс” (от англ. mutex - mutual exclusion, взаимное исключение). Мьютекс может быть захвачен одним потоком для работы в режиме монопольного использования или освобожден. Если два потока попытаются захватить мьютекс одновременно, то успех будет у одного потока, а другой будет блокирован, пока мьютекс не освободится.

1.2 Исправления орфографических ошибок в тексте

Для распознавания слов, написанных с ошибками, используется расстояние Левенштейна — минимальное количество ошибок, исправление которых приводит одно слово к другому [4]. Т. о. для введенного слова осуществляется проверка по корпусу, если данное слово не найдено в корпусе, то ищется ближайшее слово к данному по расстоянию Левенштейна.

Кроме того, следует вводить ограничение на количество ошибок, которые допускается допустить. Как говорит поговорка: «Если в слове хлеб допустить всего четыре ошибки, то получится слово пиво». Если фиксируется число

ошибок, то для коротких слов оно может оказаться избыточным. Верхнюю границу числа ошибок обычно ограничивают как процентным соотношением, так и фиксированным числом. Например, не более 30% букв входного слова, но не более 3. При этом все равно стараются найти слова с минимальным количеством ошибок [4].

1.3 Использование потоков для исправления орфографических ошибок

Поскольку задача сводится к поиску слова в корпусе, можно распараллелить поиск по этому корпусу. В таком случае каждый поток будет вычислять расстояние Левенштейна между заданным словом и некоторым словом из корпуса и в случае, если расстояние будет удовлетворять требованиям, то данное слово будет записано в массив. Для определения наилучших соответствий необходимо хранить минимальное количество ошибок на текущий момент, т. е. возможна ситуация, когда в одном потоке минимальное количество ошибок будет считано, а в другом в тот же момент изменено, следовательно возникает конфликт. То же касается и записи подходящих слов в массив, требуется отбирать лучшие k слов, поэтому возможна ситуация, когда значение длины массива считывается в одном потоке и в тот же момент изменяется в другом потоке, т. е. возникает конфликт. Для решения проблем синхронизации необходимо использовать мьютекс, чтобы обеспечить монополярный доступ к длине массива и текущему минимальному количеству ошибок.

Вывод

В данном разделе была представлена информация о многопоточности и исследуемом алгоритме.

2 Конструкторский раздел

В этом разделе будет представлено описание используемых типов данных, а также схемы алгоритмов исправления орфографических ошибок.

2.1 Требования к программному обеспечению

Программа должна поддерживать два режима работы: режим массового замера времени и режим исправления введенного слова.

Режим массового замера времени должен обладать следующей функциональностью:

- генерировать корпус слов;
- осуществлять массовый замер, используя сгенерированные данные;
- результаты массового замера должны быть представлены в виде таблицы и графика.

К режиму исправления введенного слова выдвигается следующий ряд требований:

- возможность вводить слова, которые отсутствуют в корпусе;
- наличие интерфейса для выбора действий;
- на выходе программы, набор из самых близких к введенному слов.

2.2 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие структуры и типы данных:

- слово — массив букв;
- корпус — массив слов, отсортированный в лексикографическом порядке;
- мьютекс — примитив синхронизации.

2.3 Разработка алгоритмов

На рисунке 2.1 представлена схема поиска ближайших слов в корпусе без использования потоков. На рисунке 2.2 представлена схема поиска ближайших слов в корпусе с использованием потоков. На рисунке ?? представлена схема алгоритма программы, выполняющейся в потоке.

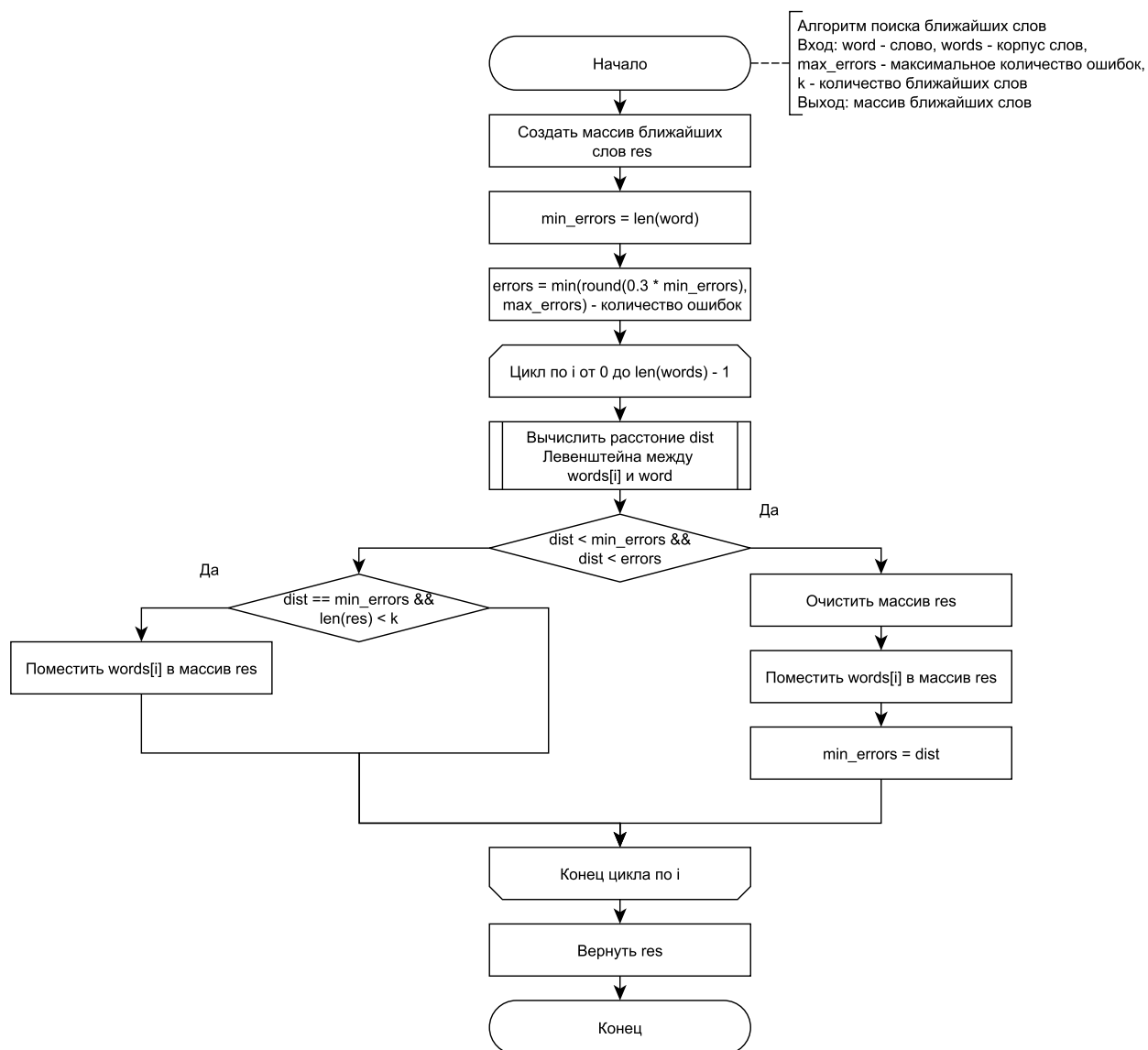


Рисунок 2.1 – Схема алгоритма поиска ближайших слов

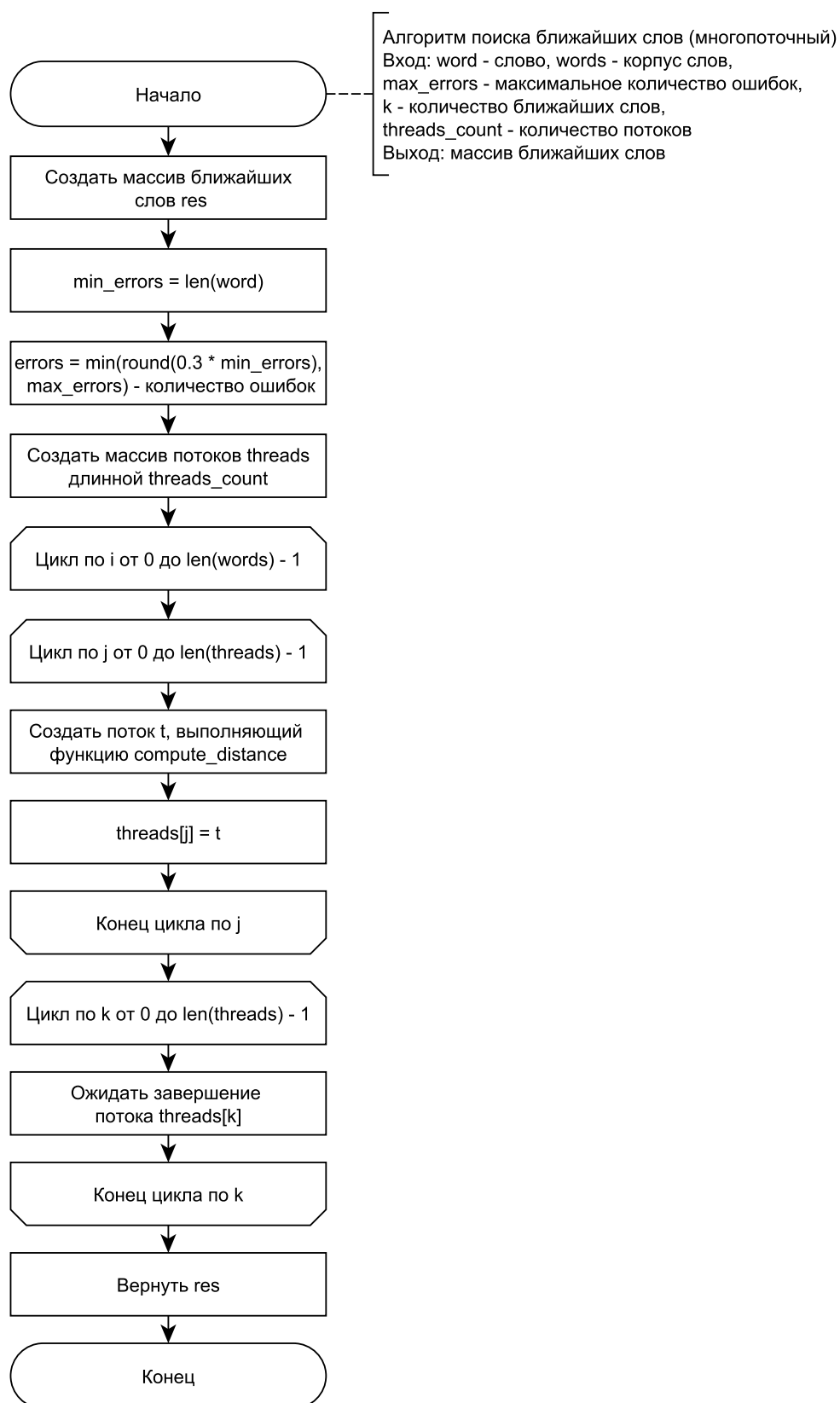


Рисунок 2.2 – Схема многопоточного алгоритма поиска ближайших слов

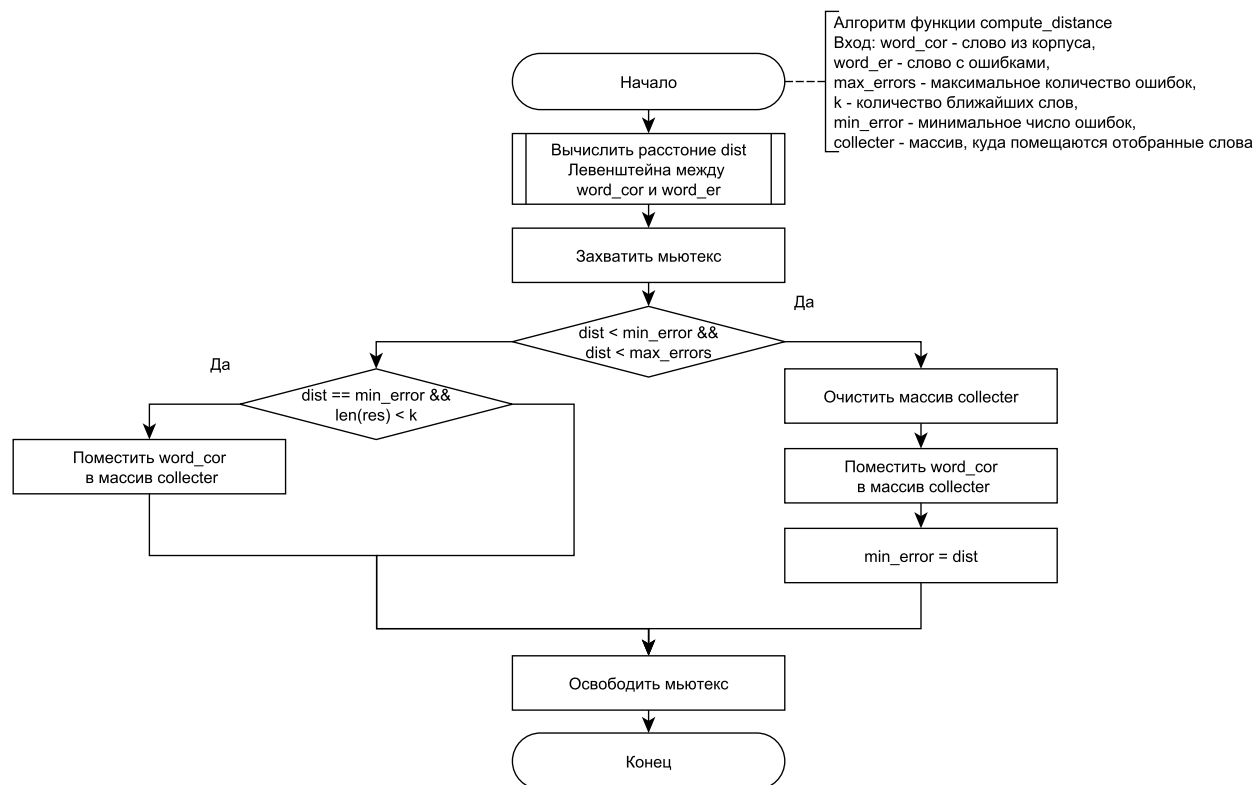


Рисунок 2.3 – Схема алгоритма потока

Поскольку в массиве в каждый момент времени содержатся слова, которые находятся на одинаковом расстоянии от введенного слова, то нет необходимости в какой-либо сортировке результатов, т. к. любое из найденных слов с одинаковой вероятностью может оказаться искомым. Поэтому шаг слияния результатов не требуется и достаточно помещать подходящее слово в массив.

Вывод

На основе теоретических данных, полученных из аналитического раздела были построены схемы требуемых алгоритмов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Гладышев Е.И. М. А.* Многопоточность в приложениях // Актуальные проблемы авиации и космонавтики. — 2012. — № 8.
2. *Stoltzfus J.* Multithreading. — — Режим доступа: <https://www.techopedia.com/definition/24297/multithreading-computer-architecture> (дата обращения: 07.12.2023).
3. *У. Ричард Стивенс С. А. Р.* UNIX. Профессиональное программирование. 3-е издание //. — — СПб.: Питер, 2018. — С. 994.
4. *Большакова Е.И. К. Э.* Автоматическая обработка текстов на естественном языке и компьютерная лингвистика //. — М.: МИЭМ, 2011. — С. 122—124.