

Министерство образования Республики Беларусь  
Учреждение образования «Белорусский государственный университет  
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЁТ  
к лабораторной работе №2  
на тему

**ОБРАБОТКА ТЕКСТОВОЙ ИНФОРМАЦИИ. РЕГУЛЯРНЫЕ  
ВЫРАЖЕНИЯ**

Выполнил: студент гр.253504  
Фроленко К.Ю.  
Проверил: ассистент кафедры информатики  
Гриценко Н.Ю.

Минск 2025

## СОДЕРЖАНИЕ

1	Формулировка задачи .....	3
2	Краткие теоритические сведения .....	4
3	Описание функций программы.....	5
4	Пример выполнения программы .....	6
	Вывод.....	7
	Список использованных источников .....	8
	Приложение А (обязательное) .....	9

## 1 ФОРМУЛИРОВКА ЗАДАЧИ

Целью данной лабораторной работы является освоение методов и средств обработки текстовой информации с использованием регулярных выражений и системных утилит *Unix*, таких как *sed* и *awk*, а также приобретение практических навыков написания надежных *shell*-скриптов. В рамках задания требуется разработать скрипт автокорректора, который автоматически преобразует входной текст, исправляя регистровые ошибки оформления. Основная задача скрипта заключается в замене строчных букв на заглавные в начале всего документа и в начале каждого нового предложения. Это означает, что первая буква файла, а также первая буква после любого знака окончания предложения (точки, восклицательного или вопросительного знака), должна быть автоматически преобразована в заглавную. При этом особое внимание уделяется случаям, когда предложение начинается на новой строке, то есть знак препинания и первая буква нового предложения могут находиться в разных строках файла.

Разрабатываемый скрипт должен работать корректно даже при наличии грязных входных данных, таких как лишние пробелы, случайные табуляции или непредвиденные переносы строки, обеспечивая устойчивость и предсказуемость поведения программы при ошибочном или неочищенном вводе. Решение может быть реализовано с использованием возможностей GNU *sed* или *awk*, что позволит не только автоматизировать процесс исправления оформления текстовых документов, но и продемонстрировать практическое применение регулярных выражений для анализа и модификации текста. Такой подход демонстрирует эффективность инструментов *Unix* для обработки данных и позволяет глубже понять принципы работы *shell*-программирования.

В результате выполнения лабораторной работы будет создан рабочий скрипт автокорректора, способный автоматически исправлять регистровые ошибки в текстовых файлах, что улучшит качество их оформления и повысит удобство последующей обработки данных. При этом студент приобретет ценный опыт разработки и отладки *shell*-скриптов, освоит методы работы с регулярными выражениями и научится интегрировать внешние утилиты в свои программные решения, что является важным аспектом автоматизации процессов в *Unix*-среде.

## 2 КРАТКИЕ ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

Обработка текстовой информации является одной из ключевых задач в современной информационной среде. В этой области особое место занимают регулярные выражения, которые представляют собой компактный и выразительный язык описания шаблонов, используемых для поиска, замены и извлечения данных из текстовых потоков. Применение регулярных выражений позволяет создавать универсальные алгоритмы для решения широкого круга задач: от простых замен символов до сложного анализа и трансформации данных. Возможность описывать закономерности в тексте с помощью регулярных выражений делает их незаменимым инструментом при автоматизации рутинных операций, обработке лог-файлов и генерации отчетов [1].

Для реализации обработки текстовой информации в Unix-среде широко применяются утилиты, такие как `sed` и `awk`. `Sed`, являясь потоковым редактором, позволяет выполнять преобразования текста «на лету», изменяя входной поток согласно заданным правилам без необходимости загрузки всего файла в память. Это делает `sed` эффективным для обработки больших объемов данных и применения сложных наборов команд для поиска, замены и удаления строк. `Awk`, напротив, представляет собой специализированный язык программирования для обработки текстовых файлов, особенно полезный при работе с табличными данными, где требуется разделение строк на поля, проведение арифметических операций и агрегация информации. Возможности `awk` по фильтрации и форматированию данных позволяют создавать компактные скрипты, способные генерировать отчеты и анализировать входной текст [2].

При разработке скриптов, использующих эти утилиты, особое внимание уделяется обработке ошибочных или «неочищенных» входных данных. Надежное решение должно предусматривать проверку корректности входного файла, его существование, а также анализ содержимого на наличие лишних пробелов, неожиданных символов или некорректных форматов. В случае обнаружения ошибок скрипт должен выдавать понятные сообщения, информируя пользователя о причинах сбоя или предлагая варианты корректировки данных. Такой подход позволяет обеспечить устойчивость работы системы и минимизировать риск аварийного завершения обработки данных в реальных условиях эксплуатации [3].

Таким образом, использование регулярных выражений в сочетании с мощными утилитами Unix, такими как `sed` и `awk`, предоставляет гибкий и эффективный инструмент для автоматизации обработки текстовой информации. Это позволяет создавать надежные скрипты, способные адаптироваться к различным форматам входных данных, выполнять сложные преобразования и обеспечивать высокое качество итоговых результатов, что является важным аспектом в современных информационных системах.

### 3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ

В данной программе автокорректора реализована автоматическая обработка текстового файла, направленная на исправление ошибок регистрового оформления. Скрипт начинается с проверки наличия входного файла (*input.txt*). Если файл отсутствует, выводится сообщение об ошибке и программа завершается, что обеспечивает надежную работу даже при ошибочных или неполных входных данных.

При наличии входного файла скрипт использует утилиту *sed* с включением расширенных регулярных выражений (опция *-E*). Сначала применяется конструкция «*:a;N;\$!ba;*», которая объединяет весь файл в один блок. Это позволяет корректно работать с переносами строк и обрабатывать случаи, когда знак окончания предложения и первая буква нового предложения располагаются на разных строках.

Первая команда замены «*s/^[[:space:]]\*([[:lower:]])/\U\1/*» ищет в начале текста возможные пробельные символы, за которыми следует первая строчная буква, и преобразует её в заглавную. Такой подход гарантирует правильное оформление самого начала документа.

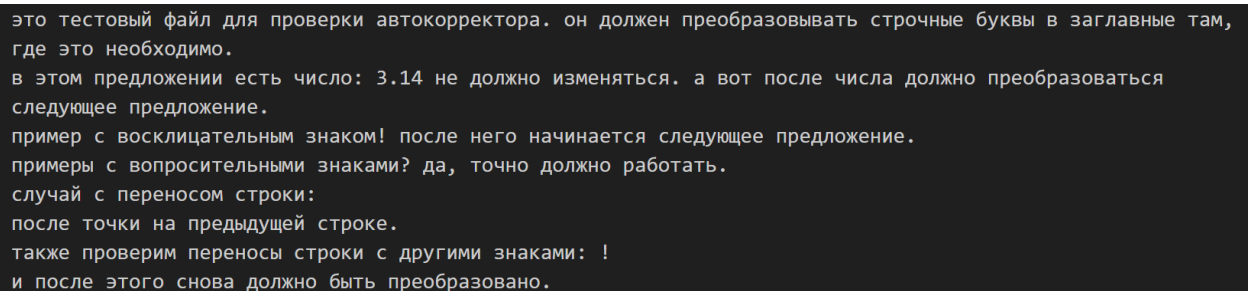
Вторая команда замены «*s/([.!?])([[:space:]]\n+)([[:lower:]])/\1\2\U\3/g*» предназначена для поиска знаков окончания предложения (точки, восклицательного или вопросительного знака), после которых следуют пробельные символы или символ перевода строки, а затем — строчная буква. Эта замена преобразует найденную строчную букву в заглавную, что позволяет обеспечить корректное оформление всех предложений, даже если начало предложения расположено на новой строке.

Обработанный текст сохраняется в выходном файле (*output.txt*), что позволяет получить документ с правильным регистровым оформлением, где каждое предложение начинается с заглавной буквы. Такой подход демонстрирует эффективное использование регулярных выражений и возможностей *Unix*-утилит для автоматизации обработки текстовой информации.

## 4 ПРИМЕР ВЫПОЛНЕНИЯ ПРОГРАММЫ

При запуске разработанного автокорректора на *Bash* программа обрабатывает входной текстовый файл, преобразуя его. Скрипт начинает работу с проверки наличия файла (*input.txt*) и, при его отсутствии, выводит сообщение об ошибке. Если файл найден, программа объединяет его содержимое в единый блок с помощью утилиты *sed* и применяет регулярные выражения для корректировки регистра. Обработанный результат записывается в выходной файл (*output.txt*).

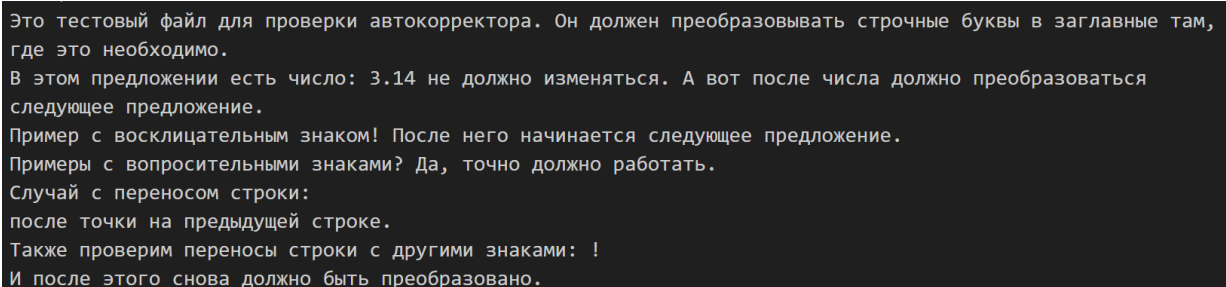
На рисунке 4.1 представлено исходное состояние текстового файла до обработки автокорректором.



```
это тестовый файл для проверки автокорректора. он должен преобразовывать строчные буквы в заглавные там,
где это необходимо.
в этом предложении есть число: 3.14 не должно изменяться. а вот после числа должно преобразоваться
следующее предложение.
пример с восклицательным знаком! после него начинается следующее предложение.
примеры с вопросительными знаками? да, точно должно работать.
случай с переносом строки:
после точки на предыдущей строке.
также проверим переносы строки с другими знаками: !
и после этого снова должно быть преобразовано.
```

Рисунок 4.1 – Исходный текстовый файл (до обработки)

После выполнения скрипта автокорректора текст преобразуется таким образом, что первая буква документа и первая буква после каждого знака окончания становятся заглавными. Результат работы программы отображается в файле *output.txt*, как показано на рисунке 4.2. Преобразование осуществляется корректно даже при наличии лишних пробелов и переносов строк между знаком препинания и первой буквой следующего предложения, что обеспечивает высокую точность обработки входных данных.



```
Это тестовый файл для проверки автокорректора. Он должен преобразовывать строчные буквы в заглавные там,
где это необходимо.
В этом предложении есть число: 3.14 не должно изменяться. А вот после числа должно преобразоваться
следующее предложение.
Пример с восклицательным знаком! После него начинается следующее предложение.
Примеры с вопросительными знаками? Да, точно должно работать.
Случай с переносом строки:
после точки на предыдущей строке.
Также проверим переносы строки с другими знаками: !
И после этого снова должно быть преобразовано.
```

Рисунок 4.2 – Текстовый файл после обработки автокорректором

Таким образом, разработанный автокорректор демонстрирует эффективность использования регулярных выражений и возможностей *Unix*-утилит для автоматизации обработки текстовой информации, обеспечивая стабильную работу даже при ошибочном или неочищенном вводе.

## ВЫВОД

В ходе выполнения лабораторной работы по разработке автокорректора, реализованного с использованием *Bash*-скриптов и утилиты *sed*, были успешно решены поставленные задачи по автоматическому исправлению регистровых ошибок в текстовых файлах. Разработанная программа корректно обрабатывает входной файл, проверяя его наличие, и применяет регулярные выражения для преобразования первой строчной буквы всего документа, а также первой строчной буквы после знаков окончания предложения. Благодаря использованию конструкции для объединения строк, скрипт способен работать с многострочным вводом, обеспечивая корректное преобразование даже при наличии лишних пробелов и неожиданных переносов строки.

Реализация автокорректора позволила не только автоматизировать процесс форматирования текстовых данных, но и продемонстрировала эффективность применения регулярных выражений и *Unix*-утилит в решении практических задач обработки информации. Особое внимание было уделено обработке ошибочных или неочищенных входных данных, что обеспечивает стабильную и надежную работу программы в любых условиях. Полученные знания и навыки в области *shell*-программирования, работы с текстовыми потоками и применения регулярных выражений имеют высокую практическую ценность и могут быть использованы для разработки других утилит автоматизации.

Таким образом, поставленные задачи лабораторной работы были успешно выполнены, что позволило создать эффективное и устойчивое решение для автоматического исправления оформления текстовых документов. Разработанная программа демонстрирует практическое применение возможностей *Unix*-среды для обработки данных и является хорошим примером интеграции теоретических знаний в практическую реализацию.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] LinuxConfig.org: Bash Scripting Tutorial: How to Write a Bash Script [Электронный ресурс]. – Режим доступа: <https://linuxconfig.org/bash-scripting-tutorial>. – Дата доступа: 30.01.2025.

[2] freeCodeCamp.org: Bash Scripting Tutorial – Linux Shell Script and Command Line for Beginners [Электронный ресурс]. – Режим доступа: <https://www.freecodecamp.org/news/bash-scripting-tutorial-linux-shell-script-and-command-line-for-beginners/>. – Дата доступа: 30.01.2025.

[3] CHARMM-GUI: Unix Tutorial [Электронный ресурс]. – Режим доступа: <https://charmm-gui.org/?doc=lecture&lesson=10&module=unix>. – Дата доступа: 30.01.2025.



# **ПРИЛОЖЕНИЕ А**

## **(обязательное)**

### **Исходный код программы**

```
#!/bin/bash

input_file="input.txt"
output_file="output.txt"

if [ ! -f "$input_file" ]; then
    echo "Error: File '$input_file' not found." >&2
    exit 1
fi

sed -E ':a;N;$!ba;
s/^[[:space:]]*([[:lower:]])/\U\1/;
s/([.!?])([[:space:]]\n+)([[:lower:]])/\1\2\u\3/g' "$input_file" >
"$output_file"
```