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»

ОТЧЕТ
К лабораторной работе № 2
на тему «Обработка текстовой информации. Регулярные выражения»

Выполнил Н. В. Климкович

Проверил Н. Ю. Гриценко

Минск 2024

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ

Цель работы изучение методов и средств обработки текстовой информации, включая регулярные выражения, и использующих их утилит. Для реализации необходимо написать скрипт (скрипты) для *sed*, *awk* и так далее, либо скрипт *shell*, обращающийся к необходимым программам, для обработки входных данных, также будет предусмотрено поведение скрипта при ошибочных входных данных.

2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Использование регулярных выражений при обработке текстовой информации представляет собой эффективный метод для поиска, извлечения и изменения текстовых данных. Регулярные выражения представляют собой шаблоны символов, используемые для поиска и сопоставления текста в строках. Они могут содержать обычные символы, такие как буквы и цифры, а также метасимволы с особым значением, например, `^` для начала строки, `$` для конца строки и `.` для любого символа.

Эти выражения определяют количество вхождений предыдущего символа или группы, например, `* ` для нуля или более вхождений, `+` для одного или более вхождений, `?` для нуля или одного вхождения. Выражения можно группировать в скобках для создания подвыражений, что позволяет применять операции и квантификаторы к группам символов.

Определение наборов символов позволяет указать диапазоны символов, например, `[*a*-z*A*-*Z*]` означает любую букву верхнего или нижнего регистра. С использованием оператора | можно указать альтернативные варианты для сопоставления, например, `*cat*|*dog*` соответствует либо *cat*, либо *dog*. Якоря, такие как ^ для начала строки и $ для конца строки, используются для указания положения в тексте.

Shell, *Awk* и *Sed* являются мощными инструментами командной строки в *Unix*-подобных операционных системах.

Оболочка (*Shell*) предоставляет интерфейс для взаимодействия пользователя с операционной системой, позволяя выполнять команды и управлять процессами и файлами.

Awk является утилитой и языком программирования, которые специализированы для обработки и анализа текстовых данных. Он основан на работе с полями и строками, что облегчает извлечение и манипуляцию данными.

Sed (*Stream Editor*) предназначен для поточной обработки текстовых данных. Он применяет правила к каждой строке входного потока, обеспечивая поиск, замену и другие манипуляции.

3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ

В результате выполнения лабораторной работы была создана консольная версия интерактивной игры «2048», которая сохраняет три лучших результата и действия игрока во время игры в файлы (Рисунок 3.1).

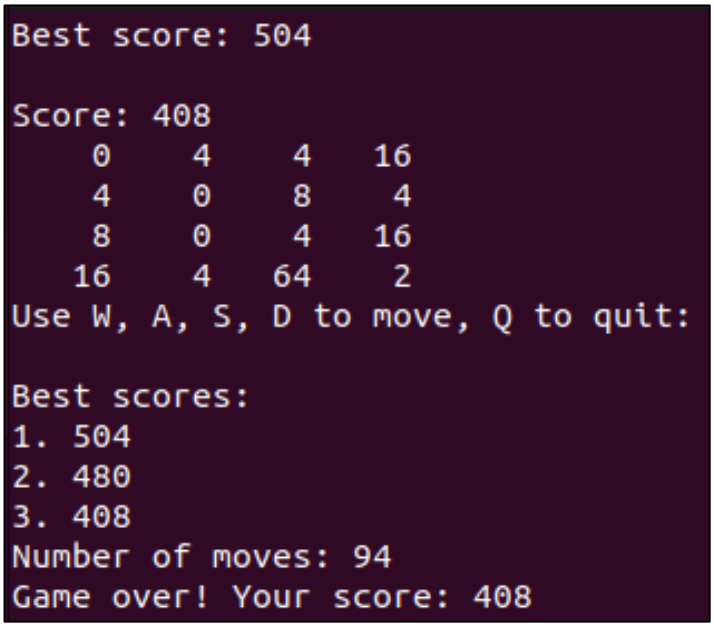

Рисунок 3.1 – Окно приложения

Программа, при запуске, считывает данные из файла, где хранится информация о лучших результатах. В ходе игры, информация о ходах игрока записывается во второй файл. По завершении игры программа считывает данные из второго файла и выводит на экран количество сделанных ходов. Если данные в файлах повреждены, например, добавлены вручную случайные символы, программа проанализирует их и не будет их считывать.

ЗАКЛЮЧЕНИЕ

В результате лабораторной работы были изучены методы и средства обработки текстовой информации, включая регулярные выражения, и использующих их утилит. библиотек. Была разработана консольная версия игры "2048", сохраняющая лучшие результаты и ходы игрока в файлы. Программа обеспечивает надежное считывание данных, а также предотвращает ошибки при обработке поврежденных файлов, обеспечивая стабильность приложения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] *Bash documentation* [Электронный ресурс]. ─ Режим доступа: https://devdocs.io/bash. – Дата доступа: 07.02.2024.

[2] Создание классических приложений для *Ubuntu* [Электронный ресурс]. – Режим доступа: https://learn.ubuntu.com. – Дата доступа: 06.02.2024.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг кода

Листинг 1 – Файл lab2.sh

```
#!/bin/bash

BEST_SCORE_FILE="best_score.txt"
MOVE_LOG="MOVE_LOG.txt"
declare -A board
size=4
score=0

function read_best_scores {
    if [[ -f "$BEST_SCORE_FILE" ]]; then
        best_scores=($(awk '/^[0-9]+$/ {print $1}' "$BEST_SCORE_FILE" | sort
-nr | head -n 3))
    else
        best_scores=()
    fi
}
function write_best_scores {
    if [[ ${#best_scores[@]} -lt 3 || $score -gt ${best_scores[-1]} ]]; then
        best_scores+=("$score")
        best_scores=($(echo "${best_scores[@]}" | tr ' ' '\n' | sort -nr))
        best_scores=("${best_scores[@]:0:3}")

        if [[ -e "$BEST_SCORE_FILE" ]]; then
            printf "%s\n" "${best_scores[@]}" > "$BEST_SCORE_FILE"
        else
            touch "$BEST_SCORE_FILE"
            printf "%s\n" "${best_scores[@]}" > "$BEST_SCORE_FILE"
        fi

    fi

    printf "\nBest scores:\n"
    for ((i=0; i<${#best_scores[@]}; i++)); do
        echo "$(($i + 1)). ${best_scores[$i]}"
    done
}
function print {
    clear
    printf "Best score: ${best_scores[0]}\n\n"
    echo "Score: $score"
    for ((i=0; i<size; i++)); do
        for ((j=0; j<size; j++)); do
            printf "%5d" ${board[$i,$j]}
        done
        echo
    done
}
function generate_random {
    local empty_cells=()
    for ((i=0; i<size; i++)); do
        for ((j=0; j<size; j++)); do
            if [[ ${board[$i,$j]} -eq 0 ]]; then
                empty_cells+=("$i,$j")
            fi
```

```
        done
    done

    if [[ ${#empty_cells[@]} -eq 0 ]]; then
        game_over
        exit
    fi

    local random_index=$((RANDOM % ${#empty_cells[@]}))
    local random_cell=${empty_cells[$random_index]}
    local random_value=$((RANDOM % 2 * 2 + 2))

    board[${random_cell}]=${random_value}
}
function initialize {
    for ((i=0; i<size; i++)); do
        for ((j=0; j<size; j++)); do
            board[$i,$j]=0
        done
    done
    generate_random
    generate_random
}
function game_over {
    write_best_scores
    lines=$(awk '/^(left|down|right|up)$/ {count++} END {print count}'
"$MOVE_LOG")
    echo "Number of moves: ${lines:-0}"
    echo "Game over! Your score: $score"
}
function move {
    local row=$1
    local col=$2
    local direction=$3
    local new_row=$row
    local new_col=$col

    case $direction in
        "up")
            new_row=$((row - 1))
            ;;
        "down")
            new_row=$((row + 1))

            ;;
        "left")
            new_col=$((col - 1))
            ;;
        "right")
            new_col=$((col + 1))
            ;;
    esac

    if [[ $new_row -ge 0 && $new_row -lt $size && $new_col -ge 0 && $new_col
-lt $size && ${board[$row,$col]} -ne 0 ]]; then
        if [[ ${board[$new_row,$new_col]} -eq 0 ]]; then
          board[$new_row,$new_col]=${board[$row,$col]}
          board[$row,$col]=0

        elif [[ ${board[$row,$col]} -eq ${board[$new_row,$new_col]} ]]; then
          board[$new_row,$new_col]=$((board[$new_row,$new_col] +
board[$row,$col]))
```

```
          board[$row,$col]=0
          score=$((score + board[$new_row,$new_col]))
        fi
    fi
}
read_best_scores
initialize
truncate -s 0 $MOVE_LOG

while true; do
    print
    echo "Use W, A, S, D to move, Q to quit:"

    read -n 1 -s direction

    case $direction in
        w|W)
            echo "up" >> "$MOVE_LOG" || touch "$MOVE_LOG" && echo "up" >>
"$MOVE_LOG"
            for ((i=size-1; i>=0; i--)); do
                for ((j=size-1; j>=0; j--)); do
                    move $i $j "up"
                done
            done
            generate_random
            ;;
        s|S)
            echo "down" >> "$MOVE_LOG" || touch "$MOVE_LOG" && echo "down" >>
"$MOVE_LOG"
            for ((i=0; i<size; i++)); do
                for ((j=0; j<size; j++)); do
                    move $i $j "down"
                done
            done
            generate_random
            ;;
        a|A)
            echo "left" >> "$MOVE_LOG" || touch "$MOVE_LOG" && echo "left" >>
"$MOVE_LOG"
            for ((j=size-1; j>=0; j--)); do
                for ((i=size-1; i>=0; i--)); do
                    move $i $j "left"
                done
            done
            generate_random
            ;;
        d|D)
            echo "right" >> "$MOVE_LOG" || touch "$MOVE_LOG" && echo "right"
>> "$MOVE_LOG"
            for ((j=0; j<size; j++)); do
                for ((i=0; i<size; i++)); do
                    move $i $j "right"
                done
            done
            generate_random
            ;;
        q|Q)
            game_over
            exit
            ;;
    esac
done
```