

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 №1  
на тему

Скрипты *shell*

Студент  
Преподаватель

Н. В. Климкович  
Н. Ю. Гриценко

Минск 2024

## СОДЕРЖАНИЕ

1 Цель работы .....	3
2 Теоретические сведения .....	4
3 Результат выполнения .....	5
Заключение .....	6
Список использованных источников .....	7
Приложение А (обязательное) Листинг кода.....	8

## 1 ЦЕЛЬ РАБОТЫ

Цель работы изучение элементов и конструкций скриптов *shell*: переменных, параметров, ветвлений, циклов, вычислений, команд *shell* и вызовов внешних программ для решения достаточно сложной задачи, имеющей практическое значение, а также принципов интеграции *Unix*-программ скриптами *shell*. Написать скрипт для оболочки *shell*, которые обеспечат получение заданным образом организованной выходной информации. Для реализации данной цели будет создана консольная версия интерактивной игры «2048» с хранением лучших результатов в файле.

## 2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Скрипты *shell* представляют собой исполняемые файлы, содержащие набор команд, предназначенных для выполнения в интерпретаторе оболочки (*shell*). Они являются средством автоматизации задач и выполнения последовательности команд в операционных системах *Unix* и аналогичных, таких как *Linux*.

Скрипты *shell* составляются на основе команд и операторов, обрабатываемых интерпретатором оболочки. Они могут включать переменные, условные операторы, циклы, функции и другие конструкции. Переменные в *shell* используются для хранения данных и передачи значений между различными частями скрипта, их можно использовать без объявления явного типа данных.

Управление выполнением команд в скриптах *shell* осуществляется через условные операторы (*if-then-else*) и циклы (*for*, *while*), что позволяет автоматизировать задачи в зависимости от условий и повторять определенные действия.

Использование функций в скриптах *shell* позволяет организовывать и структурировать код, делая его более читаемым и модульным. Функции могут вызываться из различных частей скрипта.

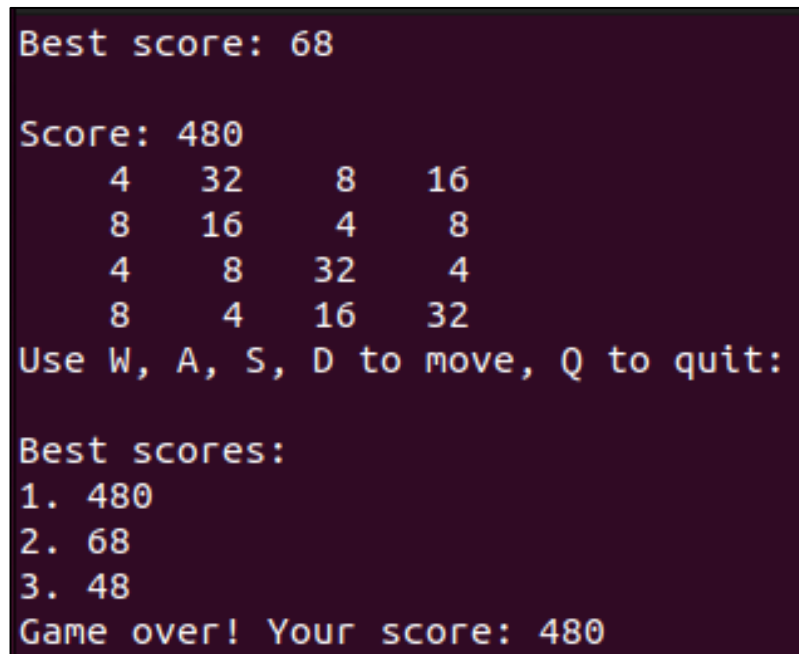
Скрипты *shell* могут включать как встроенные команды, предоставляемые оболочкой, так и внешние программы, вызываемые из скрипта. Они могут принимать аргументы из командной строки, что обеспечивает гибкость и универсальность в настройке скриптов для выполнения различных задач.

Оболочка действует как интерактивный интерфейс между пользователем и операционной системой, принимая команды от пользователя и передавая их ядру операционной системы для выполнения. В скриптах *shell* используется интерпретатор оболочки для выполнения команд. Существует несколько различных оболочек, таких как *Bash*, *Zsh*, *Csh*, *Ksh*, каждая из которых имеет свои особенности и синтаксис.

Механизмы обработки ошибок и исключений в скриптах *shell* позволяют создавать устойчивые и надежные сценарии, способные корректно реагировать на различные ситуации и условия.

### 3 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ

В результате выполнения лабораторной работы была создана консольная версия интерактивной игры «2048», которая сохраняет три лучших результата в файл (см. Рисунок 1).



```
Best score: 68

Score: 480
  4   32   8   16
  8   16   4    8
  4    8  32    4
  8    4  16   32
Use W, A, S, D to move, Q to quit:

Best scores:
1. 480
2. 68
3. 48
Game over! Your score: 480
```

Рисунок 1 – Окно приложения

В начале игры программа создает игровое поле и заполняет две ячейки, а также выводит лучший результат из предыдущих игр, сохраненных в файле. Пользователь может перемещать ячейки в различные направления или выйти из программы. Если пользователь завершает игру или проигрывает, программа выводит результаты предыдущих игр и текущий результат. Кроме того, программа сохраняет новый результат в файл, если он лучше, чем худший из трех предыдущих.

## ЗАКЛЮЧЕНИЕ

В результате лабораторной работы были изучены элементы и конструкции скриптов *shell*: переменных, параметров, ветвлений, циклов, вычислений, команд *shell* и вызовов внешних программ для решения достаточно сложной задачи, имеющей практическое значение, а также принципы интеграции *Unix*-программ скриптами *shell*. Для реализации был написан скрипт для оболочки *shell*, который обеспечил получение заданным образом организованной выходной информации. Была создана консольная версия интерактивной игры «2048» с хранением лучших результатов в файле.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] *Bash documentation* [Электронный ресурс]. — Режим доступа: <https://devdocs.io/bash/>

[2] Создание классических приложений для *Ubuntu* [Электронный ресурс]. — Режим доступа: <https://learn.ubuntu.com>

## ПРИЛОЖЕНИЕ А

### (обязательное)

### Листинг кода

#### Листинг 1 – Файл lab1.sh

```
#!/bin/bash

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

function read_best_scores {
    if [[ -f "$BEST_SCORE_FILE" ]]; then
        readarray -t best_scores < "$BEST_SCORE_FILE"
    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}"

        printf "%s\n" "${best_scores[@]}" > "$BEST_SCORE_FILE"
    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
        write_best_scores
    fi
}
```



```

        echo "Game over! Your score: $score"
        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 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
}

```

```

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

    read -n 1 -s direction

    case $direction in
        w)
            for ((i=size-1; i>=0; i--)); do
                for ((j=size-1; j>=0; j--)); do
                    move $i $j "up"
                done
            done
            ;;
        s)
            for ((i=0; i<size; i++)); do
                for ((j=0; j<size; j++)); do
                    move $i $j "down"
                done
            done
            ;;
        a)
            for ((j=size-1; j>=0; j--)); do
                for ((i=size-1; i>=0; i--)); do
                    move $i $j "left"
                done
            done
            ;;
        d)
            for ((j=0; j<size; j++)); do
                for ((i=0; i<size; i++)); do
                    move $i $j "right"
                done
            done
            ;;
        q)
            write_best_scores
            printf "\nYour score: $score\n"
            exit
            ;;
    esac

    generate_random
done

```