Министерство Образования Российской Федерации
Пермский Государственный Национальный Исследовательский университет
Государственного и муниципального управления

Отчет по лабораторной работе №6

Выполнили студенты:
ИТ-9-10, 1 курса
Суворова Полина
Палкина Агата
Смирнов Павел
Руководитель:
Иванова Е. А

г. Пермь, 2025

Оглавление

Ссылка на репозиторий

https://github.com/a1cogolik/Lab6

Структура программы

Архитектура программы

Программа состоит из основных функций и главного игрового цикла, реализующего логику игры "Быки и Коровы" с возможностью нескольких попыток и системой рекордов.

1. **Основные функции программы**

Функция difficult()

Отвечает за генерацию случайного числа в соответствии с выбранным уровнем сложности.
• Для уровня 1 (Низкий) генерирует числа в диапазоне от 1 до 10
• Для уровня 2 (Средний) генерирует числа в диапазоне от -100 до 100
• Для уровня 3 (Высокий) генерирует числа в диапазоне от -1000 до 1000

```
difficult(){ # Функция для генерации числа в соответствии с выбранной сложностью
 case $(($1)) in
 "1") local num=$((RANDOM % 10 + 1));;
 "2") local num=$((RANDOM % 200 - 100));;
 "3") local num=$((RANDOM % 2001 - 1000));;
 esac
 echo $(($num))
}
```

Функция test_input()

Выполняет проверку введенного пользователем числа на принадлежность к заданному диапазону.
• Возвращает 0 если число находится в диапазоне, 1 - если выходит за его границы

```
test_input(){  # Проверка числа на нахождение в диапазоне
 local input=$1
 local min=$2
 local max=$3
 if [[ $input -ge $min && $input -le $max ]]; then
  return 0
 fi
 return 1
}
```

Функция show_records()

Отображает таблицу лучших результатов из файла рекордов.
• Сортирует результаты по количеству попыток (чем меньше - тем лучше)
• Выводит топ-5 результатов для каждого уровня сложности
• Если файл рекордов отсутствует или пуст, сообщает об отсутствии записей

```
# Функция для показа рекордов
show_records() {
    if [[ ! -f "$RECORDS_FILE" ]] || [[ ! -s "$RECORDS_FILE" ]]; then
        echo "Рекордов пока нет!"
        return
    fi

    echo
    echo -e "${YELLOW}=====ТАБЛИЦА РЕКОРДОВ=====${NC}"
    echo -e "Уровень    Попытки  Время"
    echo -e "-------------------------"


    # Сортируем по количеству попыток (чем меньше, тем лучше)
    sort -t'|' -k2,2n "$RECORDS_FILE" | head -5 | while IFS='|' read difficulty attempts time; do
        case $difficulty in
            "1") diff_name="Низкий  ";;
            "2") diff_name="Средний ";;
            "3") diff_name="Высокий ";;
        esac
        printf "%-10s %-8d %s\n" "$diff_name" "$attempts" "${time}c"
    done
    echo -e "${YELLOW}=========================${NC}"
}
```

Функция save_record()

Сохраняет результат игры в файл рекордов.
• Записывает данные в формате:
уровень сложности | количество попыток | время игры

```
# Функция для сохранения рекорда
save_record() {
    local difficulty=$1
    local attempts=$2
    local time=$3
    echo "$difficulty|$attempts|$time" >> "$RECORDS_FILE"
}
```

Переменная play_again

Отвечает за возможность повторно сыграть игру.

• После завершения игры предлагает сыграть еще раз
• Обрабатывает ответ пользователя ("да"/"нет"). При положительном ответе запускает новую игру, при отрицательном - завершает работу программы.

```
play_again=true

while $play_again; do
echo -n "Хотите сыграть еще раз? (да/нет): "
read answer
if [[ $answer == "да" || $answer == "д" ||  $answer == "y" || $answer == "yes" ]]; then
 play_again=true
 echo "Начинаем новую игру!"
else
 play again=false
```

2. Инициализация программы

• Определяются настройки оформления интерфейса

```
# Цвета
YELLOW='\e[36m'
GREEN='\e[32m'
BLUE='\e[34m'
BOLD='\e[1m'
NC='\e[0m'
```

• Задается путь к файлу для хранения рекордов

```
# Файл для хранения рекордов
RECORDS_FILE="records.txt"
```

• Устанавливаются настройки для каждого уровня сложности

```
if [[ $user_choice -eq 1 ]]; then
 min=1
 max=10
elif [[ $user_choice -eq 2 ]]; then
 min=-100
 max=100
else
 min=-1000
 max=1000
fi
```

3. Главный игровой цикл

• Приветственный интерфейс

• Выводится название игры "Быки и Коровы"

```
echo -e "${BLUE}-------------${NC}${BOLD}Быки и Коровы${NC}${BLUE}--------------${NC}"
```

• Объясняются правила и цель игры - угадать загаданное число

```
echo -e "${GREEN}Цель игры:${NC} угадать загаданное мной число"
echo -e "${GREEN}Правила игры:${NC} компьютер загадывает число, игрок вводит числа для отгадывания. После каждой попытки дается подсказка "Больше!" или
'Меньше!"."
```

• Предлагается выбрать один из трех уровней сложности с описанием диапазонов

```
echo "Выберите уровень сложности:"
echo -e "1-${GREEN}Низкий${NC}(от 1 до 10)\n2-${GREEN}Средний${NC}(от -100 до 100)\n3-${GREEN}Высокий${NC}(-1000 до 1000)"
```

Цикл выбора сложности

• Реализован через цикл while true для обработки неверного ввода
• Пользователь вводит цифру от 1 до 3 для выбора уровня сложности
• При неверном вводе выводится сообщение об ошибке и предлагается повторить ввод
• После успешного ввода компьютер загадывает число соответствующего диапазона

```
while true; do  # Выбор сложности
 read user_choice  # Ввод пользователем (уровень сложности)
 if test_input "$user_choice" "1" "3"; then
  ai_num=$(difficult $user_choice)  # Генерация числа в соответствии с выбранной сложностью
  break
 else
  echo -n "Такой сложности нет! Выберите 1-3:"
 fi
done
```

Подготовка к игре

• Инициализируется счетчик попыток
• Фиксируется время начала игры для последующего расчета длительности

```
attemps=0  # Кол-во попыток
start_time=$(date +%s)
```

Основной игровой цикл

```
echo -e "${BLUE}------${NC}Игра началась${BLUE}------${NC}"
echo -e "(чтобы выйти введите ${GREEN}stop${NC})"
echo
```

• Пользователю предлагается ввести число или команду "stop" для выхода
• При вводе "stop" игра завершается с показом загаданного числа

```
while true; do
 echo -n "Введите число:"
 read user_try  # Попытка пользователя
 if [[ $user_try == "stop" ]]; then
  end_time=$(date +%s)
  total_time=$((end_time - start_time))
  echo "Очень жаль :c"
  echo -e "Загаданное число: ${YELLOW}$ai_num${NC}"
  break
 fi
```

• Счетчик попыток увеличивается после каждой попытки

```
((attemps++))  # +1 в счетчик попыток
```

• Даются подсказки "больше" или "меньше" относительно загаданного числа

```
if test_input "$user_try" "$min" "$max"; then
 if [[ $user_try -lt $ai_num ]]; then
  echo -e "Загаданное число ${GREEN}больше${NC}!"
 elif [[ $user_try -gt $ai_num ]]; then
  echo -e "Загаданное число ${GREEN}меньше${NC}!"
```

• Проверяется соответствие введенного числа диапазону сложности

```
else
 echo -e "Ваше число ${GREEN}вне${NC} диапазона :c"
```

• При угадывании числа выводится поздравление и результат сохраняется в таблицу рекордов

```
elif [[ $user_try -eq $ai_num ]]; then
 echo
 echo "!!! Поздравляю, вы угадали !!!"

 # Сохраняем рекорд
 end_time=$(date +%s)
 total_time=$((end_time - start_time))
 save_record "$user_choice" "$attemps" "$total_time"
```

4. Завершение игры

• Выводится статистика: затраченное время и количество попыток

```
echo
echo -e "${YELLOW}=========Статистика=========${NC}"
echo -e "Вы играли ${YELLOW}$total_time${NC} секунд"
echo -e "Колличество попыток: ${YELLOW}$attemps${NC}"
echo -e "${YELLOW}============================${NC}"
```

• Отображается таблица рекордов с лучшими результатами

```
# Показываем таблицу рекордов
show_records
```

• Предлагается сыграть еще раз или завершить программу

```
# Показываем таблицу рекордов
show_records

# Предложение сыграть еще раз
echo
echo -n "Хотите сыграть еще раз? (да/нет): "
read answer
if [[ $answer == "да" || $answer == "д" ||  $answer == "y" || $answer == "yes" ]]; then
 play_again=true
 echo "Начинаем новую игру!"
else
 play_again=false
 echo "Спасибо за игру! До свидания!"
fi
done
```

5. Особенности реализации

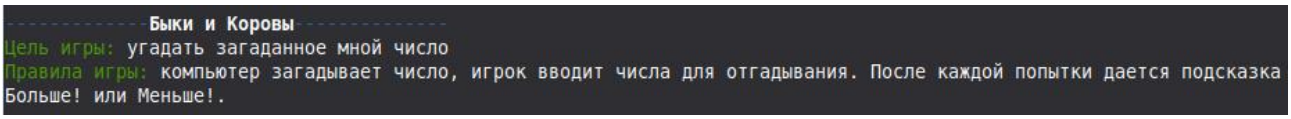
• Программа поддерживает повторные игры без перезапуска
• Рекорды сохраняются между сеансами игры
• Реализована обработка некорректного ввода
• Используется цветовое оформление для улучшения пользовательского опыта

Общий алгоритм работы с программой

Сценарий использования программы "Быки и Коровы"

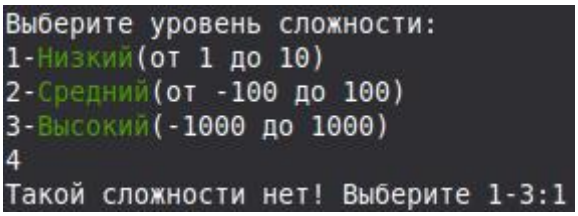
1. Запуск программы

- Пользователь запускает программу через терминал
- Система отображает главное меню с приветственным интерфейсом – цель и правила игры



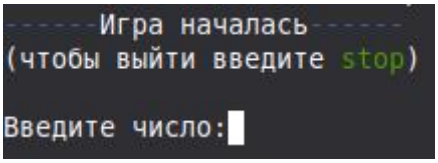
2. Выбор уровня сложности

- Пользователь изучает доступные уровни сложности:
  - Уровень 1: Низкий (диапазон 1-10)
  - Уровень 2: Средний (диапазон -100 до 100)
  - Уровень 3: Высокий (диапазон -1000 до 1000)
- Пользователь вводит цифру от 1 до 3 для выбора сложности
- Система проверяет корректность ввода
- При ошибке ввода система запрашивает повторный ввод



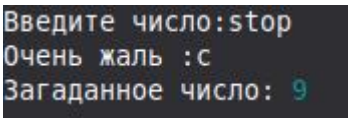
3. Начало игрового процесса

- Система генерирует случайное число в выбранном диапазоне
- Система фиксирует время начала игры
- Пользователь получает уведомление о начале игры

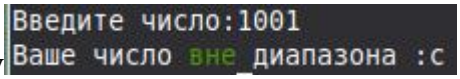


4. Процесс угадывания числа

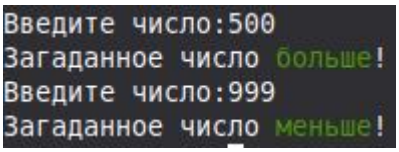
- Пользователь вводит предполагаемое число
- Система проверяет ввод на:
  - Команду "stop" для досрочного завершения

  - Принадлежность числу заданному диапазону

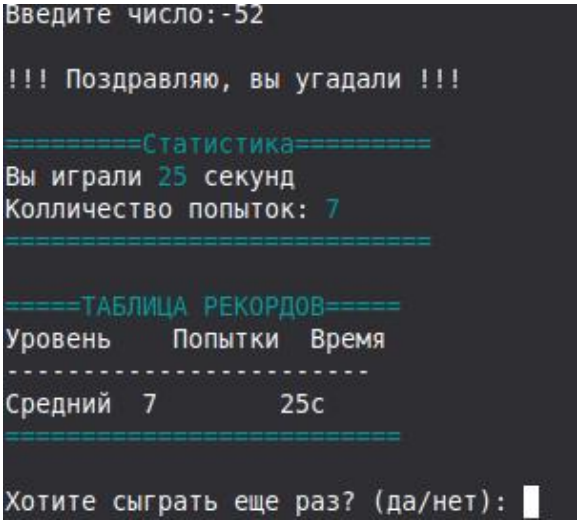
- Система предоставляет подсказки:
  - "Загаданное число больше!"
  - "Загаданное число меньше!"

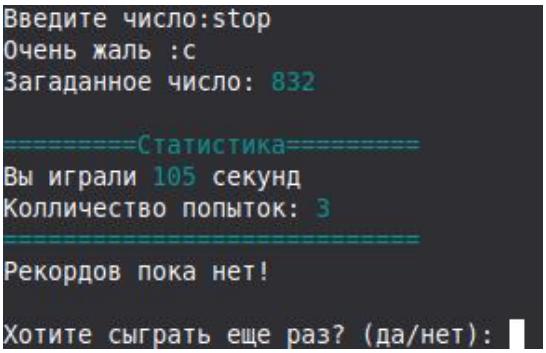
- Пользователь продолжает вводить числа, получая подсказки

5. Завершение игры

- При угадывании числа:
    - Система выводит поздравление
    - Система сохраняет результат в таблицу рекордов
    - Система выводит сообщение с предложением сыграть еще раз

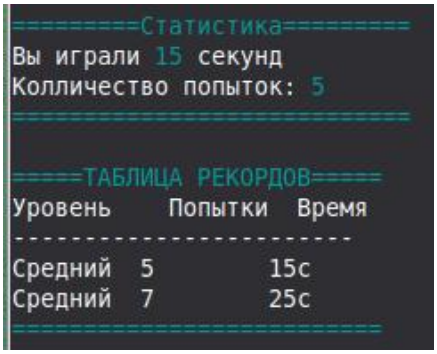


- При вводе "stop":
    - Игра завершается досрочно
    - Система показывает загаданное число, затраченное время, количество попыток
    - Система выводит сообщение с предложением сыграть еще раз



6. Просмотр статистики и рекордов

- Система отображает:
    - Затраченное время
    - Количество попыток
    - Таблицу лучших результатов



7. Повтор игры или завершение

- Система предлагает сыграть еще раз
- Пользователь выбирает:
    - "да" - начинается новая игра
    - "нет" - программа завершает работу
- При повторной игре процесс возвращается к этапу 2

Распределение ролей

Смирнов Павел

- Разработка:
  - Написание основной логики и тела скрипта игры
- Документирование:
  - Подготовка и оформление основного отчета по проделанной работе
- Администрирование:
  - Работа с репозиторием GitHub (коммиты, пуши)

Суворова Полина

- Тестирование:
  - Проверка работоспособности игры, поиск и исправление ошибок
- Доработка:
  - Реализация функций show_record(), save_records() и добавление возможности повторной игры
- Редактура:
  - Внесение правок в отчет

Палкина Агата

- Тестирование:
  - Проверка пользовательского опыта и функционала игры
- Дизайн:
  - Разработка рекомендаций по визуальному оформлению и улучшению интерфейса игры
- Презентация:
  - Подготовка презентации для представления проделанной работы