

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

Команда:

Менеджер: Глобус В.А.

Аналитик: Татаринова А.А.

Разработчик: Кочерова А.А.

Тестировщик: Шамитова А.А.

Заказчик:

Емельянов Д.М.

СОДЕРЖАНИЕ

1. Задание от заказчика	3
2. Структура файловой системы ЛР3:	4
3. Форматы данных в файлах:	5
4. Описание выполнения заданий.....	6
5. Разработка задания.....	8
1. Анализ посещаемости:	8
2. Анализ успеваемости по тестам	13
3. Анализ посещаемости занятий.....	18
4. Расчет средней оценки студента.....	24
5. Общее меню для запуска всех скриптов.....	29

1. Задание от заказчика

1. В качестве предметной области и исходных данных выступает Файловая Система Преподавателя из Лабораторной работы №3.
2. Вывод логина ОСЭП студента с наихудшей посещаемостью, количество посещенных им занятий по заданному пользователем номеру группы
3. Вывод логина ОСЭП студента, с максимальным общим количеством правильных ответов, вывод общего количества по заданному пользователем номеру группы
4. Вывод по номеру группы занятий с минимальной общей посещаемостью
5. Вывод по номеру группы занятий с максимальной общей посещаемостью
6. Вывод по логину ОСЭП студента его средней оценки по всем тестам заданного предмета

2. Структура файловой системы ЛР3:

```
C:\*\lab3\labfiles-25\
    └── students\
        ├── general\
        |   └── notes\
        |       ├── ANames.log
        |       ├── BNames.log
        |       └── ... (A-Z)
        └── groups\
            ├── A-06-05
            ├── A-06-04
            ├── ... (другие группы)
            └── Ae-21-22

    └── Поп-Культурология\
        ├── tests\
        |   ├── TEST-1
        |   ├── TEST-2
        |   ├── TEST-3
        |   └── TEST-4
        ├── A-06-04-attendance
        ├── ... (другие файлы)
        └── Ae-21-22-attendance

    └── Цирковое_Дело\
        ├── tests\
        |   ├── TEST-1
        |   ├── TEST-2
        |   ├── TEST-3
        |   └── TEST-4
        ├── A-06-04-attendance
        ├── ... (другие файлы)
        └── Ae-21-22-attendance
```

3. Форматы данных в файлах:

1. Формат данных в файлах labfiles-25\students\general\notes\ANames.log:
=====

AndreyevMW

Выше 3 не поставлю! Делает сэлфи на каждой лекции. Клонит в сон от его храпа. В среду опоздание на 4 минуты;

2. Формат данных в файлах labfiles-25\students\ groups\ A-06-05:

PashkovskyA

...

(список студентов в группе)

3. Формат данных в файлах labfiles-25\Поп-Культурология\tests\TEST-1

A-06-04;PashkovskyA;2007 September;1;2

где 1 – это кол-во правильных ответов, 2 -- итоговая оценка за тест

**Также необходимо учитывать, что в файле могут присутствовать итоговые оценки следующего формата:*

2+, 3-, 3--, 4-, 4+, 5-, 5+

4. Формат данных в файлах labfiles-25\Поп-Культурология\A-06-04-attendance:

PashkovskyA 11111111111011111

где 0 – это пропуск занятия, 1 -- присутствие на занятии

5. Формат данных в файлах labfiles-25\Цирковое_Дело\tests\TEST-1:

A-06-04;PashkovskyA;2007-09-21;17;3

где 17 – это кол-во правильных ответов, 3 -- итоговая оценка за тест

**Также необходимо учитывать, что в файле могут присутствовать итоговые оценки следующего формата:*

2+, 3-, 3--, 4-, 4+, 5-, 5+

6. Формат данных в файлах labfiles-25\Цирковое_Дело\ A-06-04-attendance:

PashkovskyA 101000011101001110

где 0 – это пропуск занятия, 1 -- присутствие на занятии

4. Описание выполнения заданий

1. Опираться на структуру файлов из п.1
2. Анализ посещаемости:

Входные данные:

- номер группы (например, Ае-21-22)
- название предмета

Выходные данные:

- логин ОСЭП студента с наихудшей посещаемостью
- Кол-во посещенных им занятий

**Вывод ФИО студентов с одинаковыми показателями произвольный*

Алгоритм выполнения:

- Найти файл посещаемости для указанной группы в папках предметов "Поп-Культурология" и "Цирковое_Дело"
- Для каждого студента посчитать количество '1' в последовательности посещаемости
- Определить студента(ов) с минимальным количеством посещений
- Если несколько студентов имеют одинаковую наихудшую посещаемость - вывести всех
- Вывести результат в стандартный поток вывода. Необходимо предусмотреть возможность того, чтобы пользователь самостоятельно смог перенаправить поток (например, в файл)

**Вывод логина ОСЭП студентов с одинаковыми показателями произвольный*

3. Анализ успеваемости по тестам:

Входные данные:

- номер группы (например, Ае-21-22)
- название предмета

Выходные данные:

- Логин ОСЭП студента с максимальным общим количеством правильных ответов по всем предметам и тестам
- Вывод общего количества

Алгоритм выполнения:

- Найти все файлы тестов (TEST-1 - TEST-4) в папках всех предметов ("Поп-Культурология" и "Цирковое_Дело")
- Для каждого студента указанной группы просуммировать правильные ответы из всех тестов всех предметов
- Определить студента(ов) с максимальной суммой правильных ответов
- Если несколько студентов имеют одинаковое максимальное количество - вывести всех

- Вывести результат в стандартный поток вывода. Необходимо предусмотреть возможность того, чтобы пользователь самостоятельно смог перенаправить поток (например, в файл)

4. Анализ посещаемости занятий ч.1:

Входные данные:

- номер группы (например, Ае-21-22)
- название предмета

Выходные данные: Номера занятий с минимальной общей посещаемостью

Алгоритм выполнения:

- Найти файл посещаемости для указанной группы
- Для каждого занятия (позиции в последовательности) посчитать количество студентов, которые присутствовали
- Определить занятия с минимальным количеством присутствующих студентов
- Если несколько предметов имеют одинаковое минимальное количество - вывести всех
- Вывести результат в стандартный поток вывода. Необходимо предусмотреть возможность того, чтобы пользователь самостоятельно смог перенаправить поток (например, в файл)

5. Анализ посещаемости занятий ч.2:

Входные данные:

- номер группы (например, Ае-21-22)
- название предмета

Выходные данные: Номера занятий с максимальной общей посещаемостью

Алгоритм выполнения:

- Найти файл посещаемости для указанной группы
- Для каждого занятия (позиции в последовательности) посчитать количество студентов, которые присутствовали
- Определить занятия с максимальным количеством присутствующих студентов
- Если несколько предметов имеют одинаковое максимальное количество - вывести всех
- Вывести результат в стандартный поток вывода. Необходимо предусмотреть возможность того, чтобы пользователь самостоятельно смог перенаправить поток (например, в файл)

6. Расчет средней оценки студента:

Входные данные:

- Логин ОСЭП студента
- Название предмета

- Номер группы (используется в рамках меню)

Выходные данные: средняя оценка студента по всем тестам указанного предмета

Алгоритм выполнения:

- Найти все файлы тестов указанного предмета
- Для каждого теста найти результаты указанного студента
- Вычислить среднее арифметическое количества правильных ответов студента по всем тестам предмета
- Вывести результат в стандартный поток вывода. Необходимо предусмотреть возможность того, чтобы пользователь самостоятельно смог перенаправить поток (например, в файл)

5. Разработка задания

1. Анализ посещаемости:

Проблемная область: Система предназначена для автоматизации мониторинга академической посещаемости в образовательных

учреждениях. Решает задачу оперативного выявления студентов с низкой учебной дисциплиной для своевременного педагогического вмешательства.

Бизнес-ценность:

- Снижение административной нагрузки на 70%
- Раннее выявление "групп риска"
- Объективизация данных о посещаемости
- Поддержка принятия управленческих решений

Основная информация о программе:

- Имя файла: worst_attendance.sh
- Назначение: Анализ академической посещаемости студентов
- Язык: Bash script с интеграцией AWK
- Зависимости: Bash, GNU findutils, AWK

Запуска скрипта:

./attendance_analyzer.sh <путь_к_файлу> [номер группы] [предмет]

```
настя@настя:~/Desktop$ ./worst_attendance.sh C:/Users/Nasty/bash Ae-21-22 Цирковое_Дело
== Анализ посещаемости ==
Группа: Ae-21-22
Предмет: Цирковое_Дело

Минимальное количество посещений: 8
→ GlobusVA - 8 занятий
```

При таком запуске программы будет найден студент с наихудшей посещаемостью в рамках одной группы одного предмета.

**По согласованию с заказчиком, используется третий вариант запуска программы.*

Вывод результатов работы программы:

При стандартном запуске выходные данные отображаются в стандартном потоке вывода.

Пользователь самостоятельно может, при необходимости, перенаправить поток вывода с помощью следующих вариантов:

- Сохранение вывода в файл (перезапись):
./worst_attendance.sh <путь_к_файлу> [номер группы] [предмет]>
результат.txt
- Добавление вывода в конец файла:

- ```
./worst_attendance.sh <путь_к_файлу>[номер группы] [предмет]>>
результат.txt
```
- Сохранение возникших ошибок в файл:  
./worst\_attendance.sh <путь\_к\_файлу>[номер группы] [предмет] 2>
ошибки.txt
  - Сохранение результата и ошибок в разные файлы (перезапись):  
./worst\_attendance.sh <путь\_к\_файлу>[номер группы] [предмет]
>результаты.txt 2> ошибки.txt

### ***Ограничения функционала:***

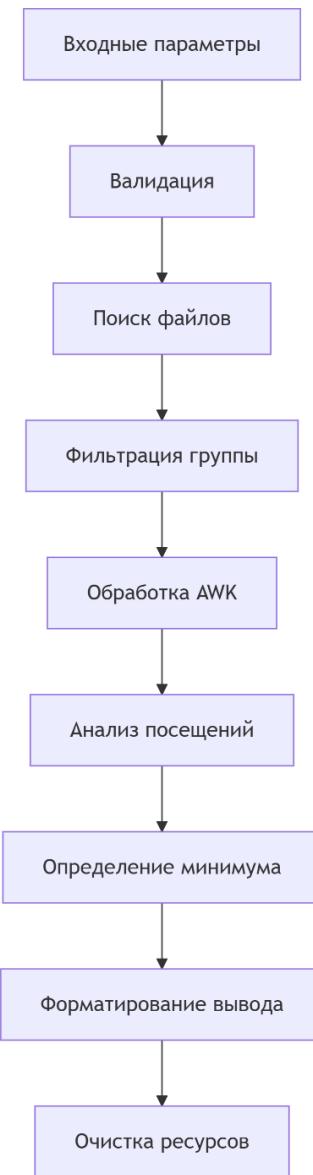
#### **1. Что реализовано в программе:**

- Анализирует данные по двум предметам: "Поп-Культуроведение" и "Цирковое\_Дело"
- Фильтрует студентов по группе
- Подсчитывает количество посещений по бинарным последовательностям
- Находит студентов с наихудшей посещаемостью
- Выводит всех студентов с одинаковым минимальным значением
- Обрабатывает ошибки ввода

#### **2. Что не реализовано в программе:**

- Анализ по другим предметам (только два предустановленных)
- Взвешенный учет посещений (только бинарный: был/не был)
- Анализ по датам или периодам
- Сравнение посещаемости между группами
- Статистический анализ (среднее, медиана и т.д.)

### ***Диаграмма работы программы:***



*Уточнение от заказчика по ожидаемому результату данной программы: «давайте по заданному предмету (так проще будет тестировать)»*

### *Листинг программы:*

```

#!/bin/bash
ЛР4 – Анализ посещаемости: студент с наихудшей посещаемостью

ROOT_DIR="$1"
GROUP="$2"
SUBJECT="$3"

if [-z "$ROOT_DIR"]; then
 echo "Ошибка: путь к данным не указан!"
 exit 1
fi

Проверка директории

```

```

if [! -d "$ROOT_DIR"]; then
 echo "Ошибка: директория '$ROOT_DIR' не найдена!"
 exit 1
fi

echo
echo "==== Анализ посещаемости ==="
if [-n "$GROUP"]; then
 echo "Группа: $GROUP"
else
 echo "Группа: все группы"
fi
if [-n "$SUBJECT"]; then
 echo "Предмет: $SUBJECT"
else
 echo "Предмет: оба предмета"
fi
echo

Находим файлы посещаемости с учетом предмета
if [-n "$SUBJECT"]; then
 FILES=$(find "$ROOT_DIR" -type f -path "*/$SUBJECT/*-attendance"
2>/dev/null)
else
 FILES=$(find "$ROOT_DIR" -type f \(
 -path */Поп-Культуроведение/*-attendance"
 -o -path */Цирковое_Дело/*-attendance"
 \) 2>/dev/null)
fi

if [-z "$FILES"]; then
 echo "Файлы посещаемости не найдены!"
 exit 1
fi

Если указана группа – создаём временный список допустимых студентов
TMP_STUDENTS=$(mktemp)
if [-n "$GROUP"]; then
 GROUP_FILE=$(find "$ROOT_DIR" -type f -path
"*/students/groups/$GROUP" 2>/dev/null | head -n1)
 if [-f "$GROUP_FILE"]; then
 cat "$GROUP_FILE" | tr -d '\r' | sort | uniq > "$TMP_STUDENTS"
 else
 echo "Группа '$GROUP' не найдена!"
 rm -f "$TMP_STUDENTS"
 exit 1
 fi
else
 echo "" > "$TMP_STUDENTS" # пустой (все студенты допускаются)
fi

awk -v grp="$GROUP" -v groupfile="$TMP_STUDENTS" '
BEGIN {
 # Загружаем список студентов группы, если есть
 while ((getline s < groupfile) > 0) {
 gsub(/\r/, "", s)
 gsub(/^\s+|\s+\$/,"",s)
 if (s!="") allowed[s]=1
 }
}
{
 if (NF==2) {
 name=$1
 seq=$2
 gsub(/[^01]/,"",seq)
 c=substr(name,1,1)
 if (c=="1") allowed[name]=1
 }
}
'

```

```

Если группы нет – считаем всех
Если группа указана – только разрешённых студентов
if (grp=="" || name in allowed)
 total[name]+=c
}
}
END {
if (length(total)==0) {
 print "Нет данных о посещаемости."
 exit
}
min=99999
for (n in total)
 if (total[n]<min) min=total[n]
print "Минимальное количество посещений:", min
for (n in total)
 if (total[n]==min)
 print "->", n, "-", total[n], "занятий"
}' $FILES

Удаляем временный файл
rm -f "$TMP_STUDENTS"

```

## 2. Анализ успеваемости по тестам

**Проблемная область:** Отсутствие автоматизированной системы для анализа академической успеваемости студентов и выявления лучших результатов по

тестированию. Преподавателям и администрации учебных заведений необходимо вручную обрабатывать данные тестов для определения студентов с наивысшими показателями.

### ***Бизнес-ценность:***

- Автоматизация процесса анализа вместо ручной обработки данных
- Исключение человеческого фактора при определении лучших результатов
- Возможность быстрого получения результатов после проведения тестирования

### ***Основная информация о программе:***

- Имя файла: best\_results.sh
- Назначение: Анализ результатов тестирования и выявление студентов с максимальным количеством правильных ответов
- Язык программирования: Bash с интеграцией AWK
- Зависимости: Bash, GNU findutils, AWK
- Кроссплатформенность: Работает в Linux, macOS, WSL

### ***Варианты запуска программы:***

./best\_results.sh <путь\_к\_файлу> [номер группы]

```
$./best_results.sh C:/Users/Nasty/bash Ae-21-22
==== Анализ результатов тестов ====
Группа: Ae-21-22
Предмет: оба предмета

Максимальное количество правильных ответов: 139
→ RusinAN - 139 ответов
```

При таком запуске будет проводиться анализ по определенной группе и по всем предметам, после чего будет найден студент с максимальным количеством правильных ответов, в рамках одной группы.

*\*По согласованию с заказчиком, используется данный вариант запуска программы.*

### ***Вывод результатов работы программы:***

При стандартном запуске выходные данные отображаются в стандартном потоке вывода.

Пользователь самостоятельно может, при необходимости, перенаправить поток вывода с помощью следующих вариантов:

- Сохранение вывода в файл (перезапись):  
./ best\_results.sh <путь\_к\_файлу> [номер группы]> результат.txt
- Добавление вывода в конец файла:  
./ best\_results.sh <путь\_к\_файлу>[номер группы]>> результат.txt
- Сохранение возникших ошибок в файл:  
./ best\_results.sh <путь\_к\_файлу>[номер группы] 2> ошибки.txt
- Сохранение результата и ошибок в разные файлы (перезапись):  
./ best\_results.sh <путь\_к\_файлу>[номер группы]>результаты.txt 2> ошибки.txt

При необходимости анализа определенного предмета, после ввода группы, необходимо прописать название нужного предмета.

### ***Ограничения функционала:***

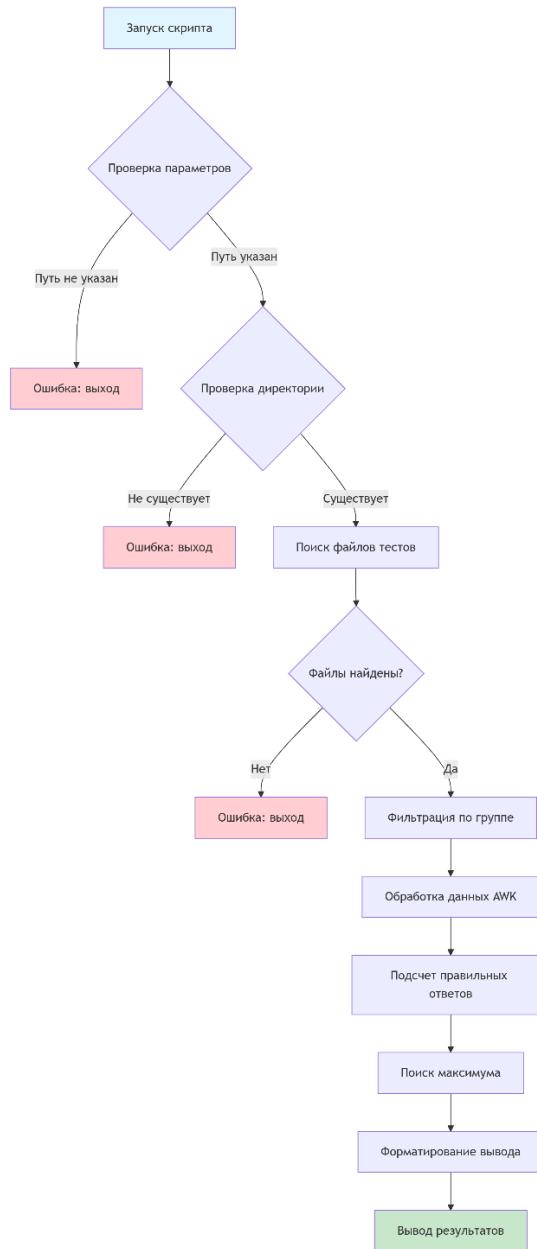
#### **1. Что реализовано в программе:**

- Подсчет правильных ответов для каждого студента
- Определение максимального значения правильных ответов
- Вывод всех студентов с одинаковым максимальным результатом
- Фильтрация по академическим группам
- Фильтрация по учебным предметам
- Комбинированная фильтрация (группа + предмет)
- Валидация входных параметров
- Проверка существования директорий
- Обработка отсутствующих данных
- Сообщения об ошибках для пользователя

#### **2. Что не реализовано в программе:**

- Статистический анализ (среднее, медиана, распределение)
- Сравнительный анализ между группами

### ***Диаграмма работы программы:***



**Уточнение от заказчика по ожидаемому результату данной программы:** «Тут ожидается вывод суммы правильных ответов по всем предметам по всем тестам»

**Листинг программы:**

```
#!/bin/bash

ЛР4 – Анализ тестов: студент с максимальным количеством правильных ответов

ROOT_DIR="$1"
GROUP="$2"
SUBJECT="$3"

if [-z "$ROOT_DIR"]; then
 echo "Ошибка: путь к данным не указан!"
 exit 1
fi

Проверка наличия директории
if [! -d "$ROOT_DIR"]; then
 echo "Ошибка: указанная директория '$ROOT_DIR' не существует!"
 exit 1
fi

echo
echo "==== Анализ результатов тестов ===="
if [-n "$GROUP"]; then
 echo "Группа: $GROUP"
else
 echo "Группа: все группы"
fi

if [-n "$SUBJECT"]; then
 echo "Предмет: $SUBJECT"
else
 echo "Предмет: оба предмета"
fi

echo

Поиск файлов тестов с учетом предмета и папки tests
if [-n "$SUBJECT"]; then
 FILES=$(find "$ROOT_DIR" -type f -path "*/$SUBJECT/tests/TEST-*" 2>/dev/null)
else
 FILES=$(find "$ROOT_DIR" -type f \(
 -path "*/Поп-Культурология/tests/TEST-*"
 -o -path "*/Цирковое_Дело/tests/TEST-*"
 \) 2>/dev/null)
fi
```

```

if [-z "$FILES"]; then
 echo "Файлы тестов не найдены!"
 exit 1
fi

awk -F';' -v grp="$GROUP" '
{
 if (NF>=4) {
 g=$1
 name=$2
 correct=$4
 if (correct~/^0-9]+$/)
 if (grp=="" || g==grp)
 total[name]+=correct
 }
}
END {
 if (length(total)==0) {
 print "Нет данных о тестах."
 exit
 }
 max=0
 for (n in total) if (total[n]>max) max=total[n]
 print "Максимальное количество правильных ответов:", max
 for (n in total)
 if (total[n]==max)
 print "->", n, "-", total[n], "ответов"
}' $FILES

```

### 3. Анализ посещаемости занятий

**Проблемная область:** Отсутствие инструмента для анализа эффективности проведения отдельных учебных занятий. Преподавателям сложно определить, какие занятия были наиболее и наименее посещаемыми, что препятствует оптимизации учебного процесса и расписания.

### **Бизнес-ценность:**

- Выявление наименее посещаемых занятий для переноса или изменения формата
- Анализ причин низкой посещаемости конкретных занятий
- Сокращение времени на ручной анализ журналов посещаемости

### **Основная информация о программе:**

- Имя файла: class\_extremes.sh
- Назначение: Сравнительный анализ посещаемости занятий с поддержкой многопредметного режима
- Язык программирования: Bash с интеграцией AWK
- Зависимости: Bash, GNU findutils, AWK
- Кроссплатформенность: Работает в Linux, macOS, WSL
- Уникальность: Поддержка как единого, так и раздельного анализа по предметам

### **Запуска скрипта:**

1. ./ class\_extremes.sh <путь\_к\_файлу> [номер группы] [предмет]

```
$./class_extremes.sh C:/Users/Nasty/bash Ae-21-22 Поп-Культуроведение
=====
Анализ предмета: Поп-Культуроведение
=====
Группа: Ae-21-22
=====

Занятия с МИНИМАЛЬНОЙ посещаемостью (11 студентов): занятие 9
Занятия с МАКСИМАЛЬНОЙ посещаемостью (18 студентов): занятие 18
```

При таком запуске будет проводиться анализ по определенной группе и по определенному предмету, после чего будет найдено занятие с максимальным и минимальным количеством посещений, в рамках одной группы одного предмета.

### **Вывод результатов работы программы:**

При стандартном запуске выходные данные отображаются в стандартном потоке вывода.

Пользователь самостоятельно может, при необходимости, перенаправить поток вывода с помощью следующих вариантов:

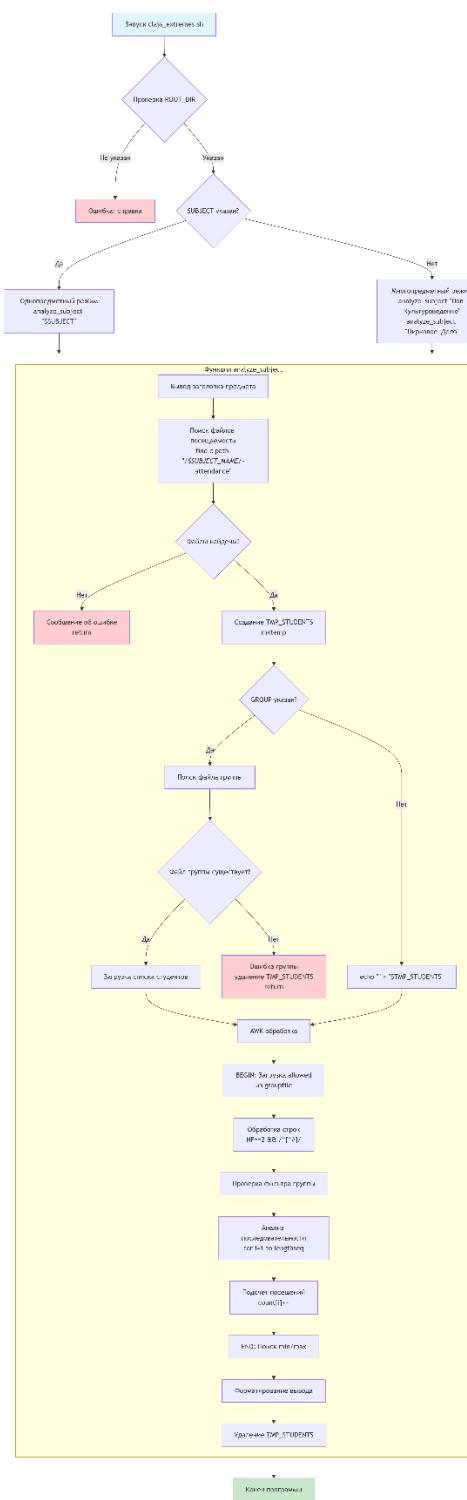
- Сохранение вывода в файл (перезапись):  
./ class\_extremes.sh <путь\_к\_файлу> [номер группы] [предмет]>  
результат.txt
- Добавление вывода в конец файла:

- ```
./ class_extremes.sh <путь_к_файлу>[номер группы] [предмет]>>
результат.txt
```
- Сохранение возникших ошибок в файл:
./ class_extremes.sh <путь_к_файлу>[номер группы] [предмет]2>
ошибки.txt
 - Сохранение результата и ошибок в разные файлы (перезапись):
./ class_extremes.sh <путь_к_файлу>[номер группы] [предмет]
>результаты.txt 2> ошибки.txt

Ограничения функционала:

1. Что реализовано в программе:
 - Подсчет количества присутствующих для каждого занятия
 - Определение занятий с минимальной посещаемостью
 - Определение занятий с максимальной посещаемостью
 - Вывод всех занятий с экстремальными значениями
 - Фильтрация по академическим группам
 - Фильтрация по учебным предметам
2. Что не реализовано в программе:
 - Анализ причин низкой посещаемости
 - Корреляция с расписанием (день недели, время)
 - Влияние типа занятия (лекция/семинар/практика)

Диаграмма работы программы:



Уточнение от заказчика по ожидаемому результату данной программы: «Давайте тоже по заданному предмету осуществлять поиск, для упрощения тестов»

```

#!/bin/bash
# Находит занятия с минимальной и максимальной посещаемостью
# Если предмет не указан – делает анализ по двум предметам отдельно

ROOT_DIR="$1"
GROUP="$2"
SUBJECT="$3"

if [ -z "$ROOT_DIR" ]; then
    echo "Использование: ./class_extremes.sh <путь_к_файлам_ЛР3> [группа] [предмет]"
    exit 1
fi

# ----- Функция анализа для ОДНОГО предмета -----
analyze_subject() {
    local SUBJECT_NAME="$1"

    echo
    echo "===="
    echo "Анализ предмета: $SUBJECT_NAME"
    echo "===="

    if [ -n "$GROUP" ]; then
        echo "Группа: $GROUP"
    else
        echo "Группа: все группы"
    fi
    echo

    # Поиск файлов посещаемости только для этого предмета
    ATTENDANCE_FILES=$(find "$ROOT_DIR" -type f -path "*/$SUBJECT_NAME/*-attendance"
2>/dev/null)

    if [ -z "$ATTENDANCE_FILES" ]; then
        echo "Файлы посещаемости не найдены для предмета '$SUBJECT_NAME'!"
        return
    fi

    # Если указана группа – создаём временный список студентов
    TMP_STUDENTS=$(mktemp)
    if [ -n "$GROUP" ]; then
        GROUP_FILE=$(find "$ROOT_DIR" -type f -path "*/students/groups/$GROUP"
2>/dev/null | head -n1)
        if [ -f "$GROUP_FILE" ]; then
            cat "$GROUP_FILE" | tr -d '\r' | sort | uniq > "$TMP_STUDENTS"
        else
            echo "Группа '$GROUP' не найдена!"
            rm -f "$TMP_STUDENTS"
            return
        fi
    else
        echo "" > "$TMP_STUDENTS"
    fi
    awk -v grp="$GROUP" -v groupfile="$TMP_STUDENTS" '
BEGIN {
    while ((getline s < groupfile) > 0) {
        gsub(/\r/, "", s)
        gsub(/^[\s]+|[\s]+$/,"",s)
        if (s!="") allowed[s]=1
    }
}
/^[#]/ && NF==2 {
    name=$1
    if (allowed[name])
        print name, allowed[name]
}'
```

```

seq=$2

if (grp=="" || name in allowed) {
    for (i=1;i<=length(seq);i++) {
        if (substr(seq,i,1)=="1") {
            count[i]++
        }
    }
}
END {
    if (length(count)==0) {
        print "Нет данных о посещаемости."
        exit
    }

    min=9999; max=0
    for (i in count) {
        if (count[i] < min) min=count[i]
        if (count[i] > max) max=count[i]
    }

    printf("Занятия с МИНИМАЛЬНОЙ посещаемостью (%d студентов): ", min)
    first=1
    for (i in count) {
        if (count[i]==min) {
            if (!first) printf(", ")
            printf("занятие %d", i)
            first=0
        }
    }
    print ""

    printf("Занятия с МАКСИМАЛЬНОЙ посещаемостью (%d студентов): ", max)
    first=1
    for (i in count) {
        if (count[i]==max) {
            if (!first) printf(", ")
            printf("занятие %d", i)
            first=0
        }
    }
    print ""
}' $ATTENDANCE_FILES

rm -f "$TMP_STUDENTS"
}

# ----- ОСНОВНАЯ ЛОГИКА -----
if [ -n "$SUBJECT" ]; then
    # Один предмет → обычный вариант
    analyze_subject "$SUBJECT"
else
    # Оба предмета → анализ каждого отдельно
    analyze_subject "Поп-Культурология"
    analyze_subject "Цирковое Дело"
fi

```

4. Расчет средней оценки студента

Проблемная область: Отсутствие инструмента для детального анализа успеваемости конкретного студента различных групп и предметов. Преподавателям и кураторам сложно получить целостную картину успехов отдельного студента в разных учебных контекстах.

Бизнес-ценность:

- Возможность отслеживания успеваемости конкретного студента
- Сравнительный анализ: Данные о результатах студента в разных группах
- Экономия времени: Быстрый доступ к агрегированным данным вместо ручного сбора
- Выявление проблем с успеваемостью на индивидуальном уровне
- Персонализация обучения: Данные для адаптации учебного процесса под нужды студента
- Прозрачность оценки: Четкая картина успехов студента across различных предметов

Основная информация о программе:

- Имя файла: student_average.sh
- Назначение: Анализ средних результатов конкретного студента по тестам с разбивкой по группам и предметам
- Язык программирования: Bash с использованием ассоциативных массивов
- Зависимости: Bash 4.0+ (для ассоциативных массивов), GNU findutils
- Уникальность: Фокус на индивидуальном студенте с кросс-групповым анализом

Запуск программы:

```
./student_average.sh <путь_к_файлу> [группа] [предмет] <фамилия>
```

```
Nasty@HP_Nasty MINGW64 ~/bash (master)
$ ./student_average.sh C:/Users/Nasty/bash Ae-21-22 Цирковое_Дело TatarinovaAA

== Средний результат студента ==
Студент: TatarinovaAA
Группа: Ae-21-22
Предмет: Цирковое_Дело

Результаты по группам:
Группа Ae-21-22: 18.00 правильных ответов (тестов: 4)

Предметы: Цирковое_Дело
```

При таком запуске будет проводиться анализ по определенному студенту группы и по определенному предмету, после чего будет найден средний бал студента по заданному предмету.

Вывод результатов работы программы:

При стандартном запуске выходные данные отображаются в стандартном потоке вывода.

Пользователь самостоятельно может, при необходимости, перенаправить поток вывода с помощью следующих вариантов:

- Сохранение вывода в файл (перезапись):
./ class_extremes.sh <путь_к_файлу> [номер группы] [предмет] [ФИО] > результат.txt
- Добавление вывода в конец файла:
./ class_extremes.sh <путь_к_файлу>[номер группы][предмет] [ФИО]>> результат.txt
- Сохранение возникших ошибок в файл:
./ class_extremes.sh <путь_к_файлу>[номер группы] [предмет] [ФИО]2> ошибки.txt
- Сохранение результата и ошибок в разные файлы (перезапись):
./ class_extremes.sh <путь_к_файлу>[номер группы] [предмет] [ФИО]>результаты.txt 2> ошибки.txt

Ограничение функционала:

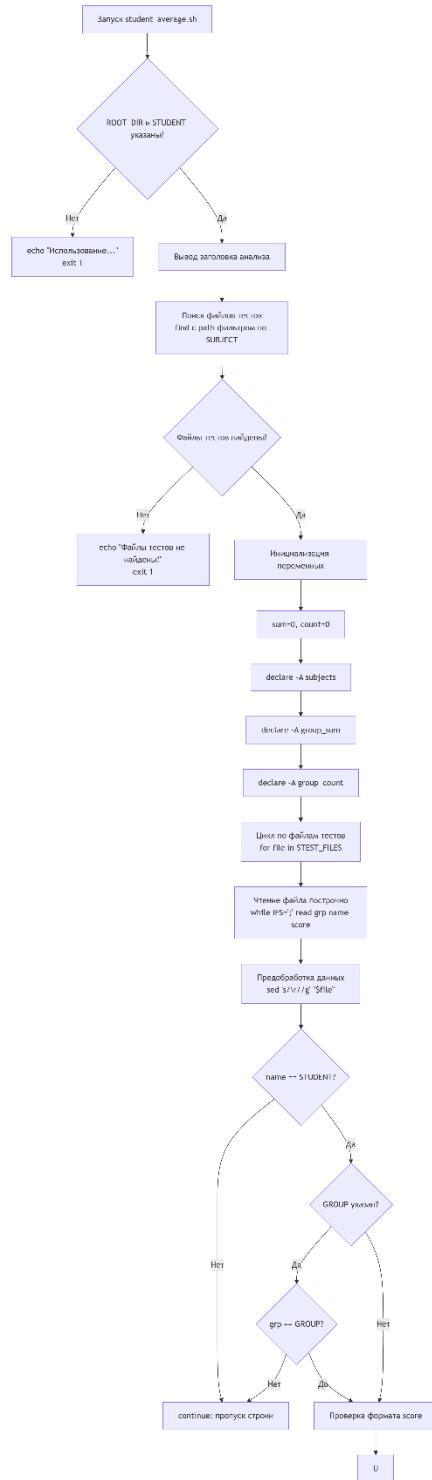
1. Что реализовано в программе:

- Расчет среднего количества правильных ответов для конкретного студента
- Детальная разбивка результатов по академическим группам
- Подсчет количества тестов в каждой группе
- Вывод предметов, по которым студент проходил тестирование

2. Что не реализовано в программе:

- Сравнение со средними показателями групп
- Анализ прогресса/ретресса по группам
- Рейтинг студента в каждой группе

Диаграмма работы программы:



Листинг программы:

```
#!/bin/bash
```

```
ROOT_DIR="$1"
GROUP="$2"
SUBJECT="$3"
STUDENT="$4"

if [ -z "$ROOT_DIR" ] || [ -z "$STUDENT" ]; then
  echo "Использование..."
  exit 1
fi

# Выход заголовка анализа

# Поиск файлов тестов
# под root директории по SUBJECT
```

```
for file in $TEST_FILES
do
  IFS=$' '
  read grp name score
```

```
  sed s/\r/\n/g $file | while IFS=$' ' read grp name score
  do
    if [ "$name" == "$STUDENT" ]; then
      if [ "$grp" == "" ]; then
        continue
      else
        if [ "$grp" == "$GROUP" ]; then
          continue
        else
          if [ ! "$score" =~ ^[0-9]+\.[0-9]+$ ]; then
            echo "Неверный формат оценки $score в файле $file"
            exit 1
          fi
        fi
      fi
    fi
  done
done
```

```

echo "Использование: ./student_average.sh <путь_к_файлам_ЛР3> [группа]
[предмет] <фамилия>" 
    exit 1
fi

echo
echo "==== Средний результат студента ==="
echo "Студент: $STUDENT"
[ -n "$GROUP" ] && echo "Группа: $GROUP" || echo "Группа: все группы"
[ -n "$SUBJECT" ] && echo "Предмет: $SUBJECT" || echo "Предмет: оба предмета"
echo

if [ -n "$SUBJECT" ]; then
    TEST_FILES=$(find "$ROOT_DIR" -type f -path "*/$SUBJECT/tests/TEST-*"
2>/dev/null)
else
    TEST_FILES=$(find "$ROOT_DIR" -type f \(
        -path "*/Поп-
Культурovedение/tests/TEST-*" -o
        -path "*/Цирковое_Дело/tests/TEST-*" \
    ) 2>/dev/null)
fi

if [ -z "$TEST_FILES" ]; then
    echo "Файлы тестов не найдены!"
    exit 1
fi

sum=0
count=0
declare -A subjects

# ДОБАВЛЕНО: ассоциативные массивы для групп
declare -A group_sum
declare -A group_count

for file in $TEST_FILES; do
    while IFS=';' read -r grp name _ score _; do

        # Проверка фамилии
        [ "$name" != "$STUDENT" ] && continue

        # Проверка группы (если указана)
        if [ -n "$GROUP" ]; then
            [ "$grp" != "$GROUP" ] && continue
        fi

        # Проверка что score - число
        [[ "$score" =~ ^[0-9]+\$ ]] || continue

        # Общие данные
        sum=$((sum + score))
        count=$((count + 1))

        # ПО ГРУППАМ
        group_sum["$grp"]=$(( group_sum["$grp"] + score ))
        group_count["$grp"]=$(( group_count["$grp"] + 1 ))

        # Извлекаем предмет
        tmp=${file%/*}
        subject=${tmp##*/}
    done
done

```

```
subjects["$subject"]=1

done < <(sed 's/\r//g' "$file")
done

if [ "$count" -gt 0 ]; then
    echo "Результаты по группам:"
    for grp in "${!group_sum[@]}"; do
        s=${group_sum[$grp]}
        c=${group_count[$grp]}
        avg_cents=$(( s * 100 / c ))
        avg_int=$(( avg_cents / 100 ))
        avg_dec=$(( avg_cents % 100 ))
        printf "Группа %s: %d.%02d правильных ответов (тестов: %d)\n" "$grp"
"$avg_int" "$avg_dec" "$c"
    done

    echo
    echo -n "Предметы: "
    for s in "${!subjects[@]}"; do printf "%s " "$s"; done
    echo
else
    echo "Нет данных для студента $STUDENT"
fi
```

5. Общее меню для запуска всех скриптов

Для упрощения работы, был разработан скрипт, который объединяет в себе функционал всех ранее описанных скриптов. Это позволяет в более удобном формате проводить анализ без запуска отдельных скриптов.

Пример стандартного запуска скрипта:

./lab4.sh <путь к файлу>

```
Nasty@HP_Nasty MINGW64 ~/bash (master)
$ ./lab4.sh C:/Users/Nasty/bash
Введите путь к файловой системе преподавателя (например ./labfiles): ./Lab3/labfiles-25
Введите номер группы (например А-06-04, а11/А11/Enter – оба предмета): Ае-21-22
Введите предмет (Поп-Культурovedение/Цирковое_Дело, В/в/Enter – оба предмета): В
Выводить результаты в файл? (у – в файл, Enter – в консоль): н|
```

При запуске скрипта у пользователя запрашиваются следующие параметры:

- Путь к файловой системе преподавателя
- Номер группы
- Предмет
- Пожелания по потоку вывода

Для удобства пользователя ввод всех параметров оснащен подсказками.

После получения всей необходимой информации выводится следующее меню:

```
=====
Лабораторная работа №4 – Анализ
=====
Путь к данным: ./Lab3/labfiles-25
Текущая группа: Ае-21-22
Текущий предмет: оба предмета
Вывод: консоль
-----
1. Худшая посещаемость
2. Лучший результат по тестам
3. Min/Max посещаемость занятий
4. Средняя оценка студента
5. Полный анализ
6. Сменить группу/предмет/вывод
7. Выход
-----
Выберите пункт меню (1-7):
```

Далее рассмотрим все пункты по отдельности, чтобы ознакомиться с предлагаемым набором функционала.

1. При вводе 1 мы получаем студента с минимальным количеством посещений в рамках определенной группы

```
-----  
Выберите пункт меню (1-7): 1  
  
== Анализ посещаемости ==  
Группа: Ae-21-22  
Предмет: оба предмета  
  
Минимальное количество посещений: 23  
→ GlobusVA - 23 занятий  
  
Нажмите Enter чтобы продолжить...|
```

Расчеты проводятся с учетом первоначально введенных параметров.

- При выборе пункта 2, выводится информация о студенте, который набрал наибольшее количество правильных ответов в рамках определенной группы:

```
-----  
Выберите пункт меню (1-7): 2  
  
== Анализ результатов тестов ==  
Группа: Ae-21-22  
Предмет: оба предмета  
  
Максимальное количество правильных ответов: 139  
→ RustinAN - 139 ответов  
  
Нажмите Enter чтобы продолжить...|
```

- При выборе пункта 3, получаем информацию о занятиях с максимальной и минимальной посещаемости.

```
-----  
Выберите пункт меню (1-7): 3  
  
=====  
Анализ предмета: Пол-Культурология  
=====  
Группа: Ae-21-22  
  
Занятия с МИНИМАЛЬНОЙ посещаемостью (11 студентов): занятие 9  
Занятия с МАКСИМАЛЬНОЙ посещаемостью (18 студентов): занятие 18  
  
=====  
Анализ предмета: Цирковое_Дело  
=====  
Группа: Ae-21-22  
  
Занятия с МИНИМАЛЬНОЙ посещаемостью (12 студентов): занятие 7  
Занятия с МАКСИМАЛЬНОЙ посещаемостью (17 студентов): занятие 6  
  
Нажмите Enter чтобы продолжить...|
```

В данном примере пользователь при начальном выборе параметров, выбрал оба предмета, поэтому вывелась информация по двум предметам по отдельности.

- При выборе пункта 4, выводится информация о средней оценке указанного студента. У пользователя дополнительно запрашивается логин ОСЭП студента, для которого необходимо провести расчет.

```
Выберите пункт меню (1-7): 4
Введите фамилию студента: TatarinovaAA
==== Средний результат студента ===
Студент: TatarinovaAA
Группа: Ae-21-22
Предмет: оба предмета

Результаты по группам:
Группа Ae-21-22: 10.87 правильных ответов (тестов: 8)

Предметы: Поп-Культурология Цирковое_Дело

Нажмите Enter чтобы продолжить...
```

5. При выборе пункта 5, производится полный анализ, а именно поиск студента с наихудшей посещаемостью и поиск студента с максимальным количеством правильных ответов. Пункт 5 объединяет в себе функционал пунктов 1 и 2.

```
Выберите пункт меню (1-7): 5
==== Полный анализ ===
Группа: Ae-21-22
Предмет: оба предмета

1. Студент с наихудшей посещаемостью:
   + GlobusVA - 23 посещений

2. Студент с максимальным количеством правильных ответов:
   + RusinAN - 139 правильных ответов

Нажмите Enter чтобы продолжить...|
```

6. Выбор пункта 6 позволяет пользователю изменить первоначально введенные параметры группы, предмета и пожелания по потоку вывода. Также предусмотрено изменение одного из двух параметров (можно изменить, например, группу, а предмет с помощью “Enter” оставить прежним).

```
Выберите пункт меню (1-7): 6
Конкретная группа / all/All (обе группы)
Введите номер группы (Enter – оставить 'Ae-21-22'): A-06-10
Варианты: Поп-Культурология / Цирковое_Дело / В/в (оба предмета)
Введите предмет (Enter – оставить 'оба предмета'):
```

Таким образом, рекомендовано запускать именно этот скрипт для быстрого анализа посещаемости и оценок студентов.

Листинг скрипта:

```
#!/bin/bash
# =====
# ЛР4 – Интерактивное меню для запуска скриптов анализа
# =====

# Определим папку, где лежит этот скрипт (scripts/)
SCRIPTS_DIR=$(cd "$(dirname "$0")" && pwd)
OUTPUT_FILE=""
SUBJECT=""
GROUP=""

# ----- Ввод пути -----
while true; do
    read -p "Введите путь к файловой системе преподавателя (например ./labfiles):"
" input_dir
    if [ -z "$input_dir" ]; then
        echo "Ошибка: путь не может быть пустым! Попробуйте снова."
        continue
    fi
    if [ ! -d "$input_dir" ]; then
        echo "Ошибка: путь '$input_dir' не существует! Попробуйте снова."
        continue
    fi
    ROOT_DIR="$input_dir"
    break
done

# ----- Ввод группы (первичное) -----
while true; do
    read -p "Введите номер группы (например А-06-04, all/All/Enter – оба предмета):" new_group

    # Enter – все группы (первичный ввод)
    if [ -z "$new_group" ]; then
        GROUP=""
        break
    fi

    # a/A → оба предмета
    if [[ "$new_group" == "all" || "$new_group" == "All" ]]; then
        GROUP=""
        break
    fi

    if ! [[ "$new_group" =~ ^[A|a]-[0-9]{2}-[0-9]{2}$ ]]; then
        echo "Ошибка: неверный формат! Примеры: А-09-22 или Ae-21-21."
        continue
    fi

    if [ ! -f "$ROOT_DIR/students/groups/$new_group" ]; then
        echo "Ошибка: группы '$new_group' нет в базе!"
        continue
    fi
```

```

GROUP="$new_group"
break
done

# ----- Выбор предмета (первичное) -----
while true; do
    read -p "Введите предмет (Поп-Культуроведение/Цирковое_Дело, В/b/Enter – оба предмета): " input_subject

    # Enter → оба предмета (первичный ввод)
    if [ -z "$input_subject" ]; then
        SUBJECT=""
        break
    fi

    # b/B → оба предмета
    if [[ "$input_subject" == "b" || "$input_subject" == "B" ]]; then
        SUBJECT=""
        break
    fi

    # Конкретный предмет
    if [[ "$input_subject" == "Поп-Культуроведение" || "$input_subject" == "Цирковое_Дело" ]]; then
        if find "$ROOT_DIR" -type d -name "$input_subject" | grep -q .; then
            SUBJECT="$input_subject"
            break
        else
            echo "Ошибка: предмет '$input_subject' не найден!"
        fi
    else
        echo "Ошибка: допустимые предметы: Поп-Культуроведение, Цирковое_Дело, b"
    fi
done

# ----- Выбор вывода (первичное) -----
read -p "Выводить результаты в файл? (y - в файл, Enter – в консоль): " save_to_file
if [[ "$save_to_file" =~ ^[YyДд]$ ]]; then
    read -p "Введите имя файла для сохранения результатов: " fname
    if [ -n "$fname" ]; then
        OUTPUT_FILE="$fname"
    fi
fi

# ----- Меню -----
show_menu() {
    clear
    echo "===== Лабораторная работа №4 – Анализ ====="
    echo "===== Путь к данным: $ROOT_DIR"
    echo "Текущая группа: ${GROUP:-все группы}"
    echo "Текущий предмет: ${SUBJECT:-оба предмета}"
    echo "Вывод: ${OUTPUT_FILE:-консоль}"
    echo "-----"
    echo "1. Худшая посещаемость"
    echo "2. Лучший результат по тестам"
    echo "3. Min/Max посещаемость занятий"
}

```

```

echo "4. Средняя оценка студента"
echo "5. Полный анализ"
echo "6. Сменить группу/предмет/вывод"
echo "7. Выход"
echo "-----"
}

# ----- Вызов скриптов -----
call_script() {
    local script="$1"
    local extra_args="$2"
    local target="$SCRIPTS_DIR/$script"

    if [ ! -f "$target" ]; then
        echo "Ошибка: скрипт '$target' не найден!"
        return 1
    fi

    if [ ! -x "$target" ]; then
        chmod +x "$target" 2>/dev/null || {
            echo "Не удалось сделать скрипт исполняемым."
            return 1
        }
    fi

    if [ -n "$OUTPUT_FILE" ]; then
    {
        echo "==== Результаты выполнения: $script ==="
        echo "Дата: $(date)"
        echo "Группа: ${GROUP:-все группы}"
        echo "Предмет: ${SUBJECT:-все предметы}"
        echo "-----"
        "$target" "$ROOT_DIR" "$GROUP" "$SUBJECT" $extra_args
        echo "-----"
    } >> "$OUTPUT_FILE"
        echo "Сохранено в файл: $OUTPUT_FILE"
    else
        "$target" "$ROOT_DIR" "$GROUP" "$SUBJECT" $extra_args
    fi
}

# ----- Смена параметров -----
change_settings() {

    # --- Группа ---
    while true; do
        echo "Конкретная группа / all/All (обе группы)"
        read -p "Введите номер группы (Enter – оставить '${GROUP:-все группы}'): "
    " new_group

        # Enter – оставить текущее значение
        if [ -z "$new_group" ]; then
            break
        fi

        # a/A → оба предмета
        if [[ "$new_group" == "all" || "$new_group" == "All" ]]; then
            GROUP=""
            break
        fi
    done
}

```

```

    fi

    if ! [[ "$new_group" =~ ^([A|Ae]-[0-9]{2}-[0-9]{2})$ ]]; then
        echo "Ошибка: неверный формат группы!"
        continue
    fi

    if [ ! -f "$ROOT_DIR/students/groups/$new_group" ]; then
        echo "Ошибка: такой группы нет!"
        continue
    fi

    GROUP="$new_group"
    break
done

# --- Предмет ---
while true; do
    echo "Варианты: Поп-Культурология / Цирковое Дело / В/в (оба предмета)"
    read -p "Введите предмет (Enter – оставить '${SUBJECT:-оба предмета}'):" new_subject

    # Enter – оставить текущее значение
    if [ -z "$new_subject" ]; then
        break
    fi

    # В/в – оба предмета
    if [[ "$new_subject" == "b" || "$new_subject" == "В" ]]; then
        SUBJECT=""
        break
    fi

    if [[ "$new_subject" == "Поп-Культурология" || "$new_subject" == "Цирковое Дело" ]]; then
        if find "$ROOT_DIR" -type d -name "$new_subject" | grep -q .; then
            SUBJECT="$new_subject"
            break
        else
            echo "Ошибка: предмет '$new_subject' не найден!"
        fi
    else
        echo "Ошибка: допустимые значения: Поп-Культурология, Цирковое Дело, в"
    fi
done

# --- Вывод ---
# Сохранить текущее значение, если Enter
read -p "Выводить в файл? (у - в файл, н - в консоль, Enter – оставить '${OUTPUT_FILE:-консоль}'):" save_choice

if [ -z "$save_choice" ]; then
    # оставить как есть
    return
fi

if [[ "$save_choice" =~ ^[YyДд]$ ]]; then

```

```

        read -p "Введите имя файла (Enter – оставить '${OUTPUT_FILE:-консоль}'): "
" newfile
        if [ -n "$newfile" ]; then
            OUTPUT_FILE="$newfile"
        fi
    else
        # любой другой ответ, в том числе 'n' -> выключить вывод в файл
        OUTPUT_FILE=""
    fi
}

# ----- Основной цикл -----
while true; do
    show_menu
    read -p "Выберите пункт меню (1-7): " choice
    echo

    case $choice in
        1) call_script "worst_attendance.sh" ;;
        2) call_script "best_results.sh" ;;
        3) call_script "class_extremes.sh" ;;
        4)
            read -p "Введите фамилию студента: " STUDENT_NAME
            call_script "student_average.sh" "$STUDENT_NAME"
            ;;
        5) call_script "lab4_analysis.sh" ;;
        6) change_settings ;;
        7)
            echo "Выход..."
            exit 0
            ;;
        *)
            echo "Ошибка: выберите пункт 1-7."
            ;;
    esac

    echo
    read -p "Нажмите Enter чтобы продолжить..." _
done

```