

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.**

Студент: гр.153502
Миненков Я. А.

Проверил: Гриценко Н. Ю.

Минск 2024

СОДЕРЖАНИЕ

1 Формулировка задачи	3
2 Теоретические сведения.....	4
3 Описание функций программы	5
Список использованных источников.....	6
Приложение А (обязательное) Исходный код программы.....	7

1 ФОРМУЛИРОВКА ЗАДАЧИ

Целью выполнения лабораторной работы является изучение методов и средств обработки текстовой информации, включая регулярные выражения, и использующих их утилит.

В качестве задачи требуется написать скрипт для оболочки *shell*, реализующий «Табличный калькулятор», обращающийся к необходимым программам, для обработки входных данных (файлов). Входные данные – таблицы (матрицы) в файлах в формате *CSV*.

2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

AWK – си-подобный сценарный язык строчного разбора и обработки входного потока (например, текстового файла) по заданным шаблонам (регулярным выражениям). Может использоваться в сценариях командной строки.

AWK рассматривает входной поток как список записей. Каждая запись делится на поля. На основе этой информации выполняется некоторый определённый программистом алгоритм обработки. По умолчанию разделителем записей является символ новой строки (то есть записи — это то же самое, что строки), разделителем полей — символ пробела или табуляции, или последовательность таких символов. Символы-разделители можно явно определить в программе. Символ-разделитель полей можно определить и в командной строке. *AWK*-программа состоит из операторов (правил), имеющих вид шаблон {действие}.

Каждая запись поочерёдно сравнивается со всеми шаблонами, и каждый раз, когда она соответствует шаблону, выполняется указанное действие. Если шаблон не указан, то действие выполняется для любой записи. Если не указано действие, то запись выводится. В *AWK* также существует 2 предопределённых шаблона *BEGIN* и *END*. *BEGIN* выполняется до начала обработки входного потока. *END* – после обработки последней записи входного потока. Действие может состоять из последовательности операторов, разделяемых точкой с запятой, переводом строки или закрывающей скобкой [1].

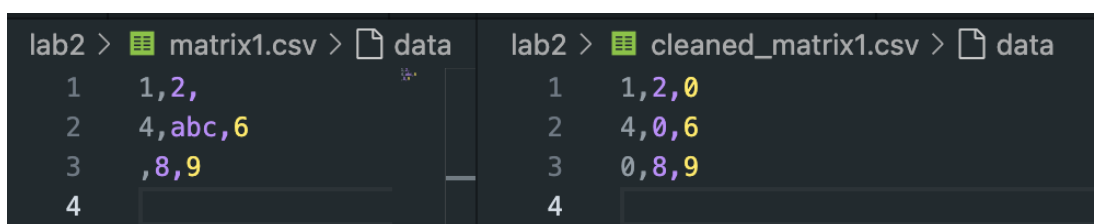
Регулярное выражение, или *regex*, – это способ описания набора строк. Поскольку регулярные выражения являются фундаментальной частью *awk* программирования.

Регулярное выражение, заключённое в косую черту (') – это *awk* шаблон, который соответствует каждой входной записи, текст которой принадлежит этому набору. Простейшее регулярное выражение – это последовательность букв, цифр или того и другого. Такое регулярное выражение соответствует любой строке, содержащей эту последовательность. Таким образом, регулярное выражение «*foo*» соответствует любой строке, содержащей «*foo*». Таким образом, шаблон */foo/* соответствует любой входной записи, содержащей три соседних символа «*foo*» в любом месте записи. Другие виды регулярных выражений позволяют указывать более сложные классы строк [2].

3 ОПИСАНИЕ ФУНКЦИЙ ПРОГРАММЫ

Скрипт обеспечивает выполнение математических операций над двумя матрицами, представленными в формате *CSV*. Он принимает три аргумента командной строки: два файла матрицы и тип операции. На рисунке 5 произведен подсчет контрольной суммы заданных файлов.

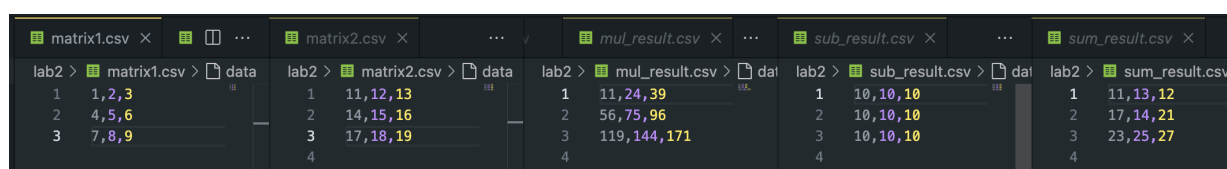
Скрипт сначала очищает обе матрицы от нечисловых значений, заменяя их на ноль. Это делается с помощью функции *clean_matrix*, которая использует *awk* для обработки каждого поля в каждой строке и замены нечисловых значений на ноль (рисунок 1).



lab2 > matrix1.csv > data	lab2 > cleaned_matrix1.csv > data
1 1,2,	1 1,2,0
2 4,abc,6	2 4,0,6
3 ,8,9	3 0,8,9
4	4

Рисунок 1 – Очистка матрицы

Далее выполняется указанная операция над двумя матрицами. Операция может быть одной из следующих: сложение (*add*), вычитание (*sub*) или умножение (*mul*). Это делается с помощью функций *add_matrices*, *subtract_matrices* и *multiply_matrices*, которые также используют *awk* для выполнения соответствующих математических операций над полями в каждой строке (рисунок 2).



matrix1.csv	matrix2.csv	mul_result.csv	sub_result.csv	sum_result.csv
1 1,2,3	1 11,12,13	1 11,24,39	1 10,10,10	1 11,13,12
2 4,5,6	2 14,15,16	2 56,75,96	2 10,10,10	2 17,14,21
3 7,8,9	3 17,18,19	3 119,144,171	3 10,10,10	3 23,25,27
4	4	4	4	4

Рисунок 2 – Результаты умножения, вычитания, сложения матриц

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Awk [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/AWK>. – Дата доступа: 27.02.2024.

[2] The GNU Awk User's Guide [Электронный ресурс]. – Режим доступа: <https://www.gnu.org/software/gawk/manual/gawk.html#Regexp>. – Дата доступа: 27.02.2024.

ПРИЛОЖЕНИЕ А

(обязательное)

Исходный код программы

```
#!/bin/zsh

print_help() {
    echo "Usage: ./lab2.zsh [matrix1] [matrix2] [operation]"
    echo "operation: The operation to perform on the matrices.
Can be 'add', 'sub', or 'mul'."
    echo ""
    echo "Example: ./lab2.zsh matrix1.csv matrix2.csv add"
}

clean_matrix() {
    awk -F, '{
        for(i=1; i<=NF; i++) {
            if ($i ~ /^[^0-9.]/ || $i == "") {
                $i = 0
            }
        }
    }1' OFS=, "$1" > cleaned_"$1"
}

add_matrices() {
    awk -F,
'NR==FNR{a[NR]=$0;next}{split(a[FNR],b,",");for(i=1;i<=NF;i++)$i=b[i]+
$i}1' OFS=, cleaned_"$1" cleaned_"$2" > add_result.csv
}

subtract_matrices() {
    awk -F,
'NR==FNR{a[NR]=$0;next}{split(a[FNR],b,",");for(i=1;i<=NF;i++)$i=$i-
b[i]}1' OFS=, cleaned_"$1" cleaned_"$2" > sub_result.csv
}

multiply_matrices() {
    awk -F,
'NR==FNR{a[NR]=$0;next}{split(a[FNR],b,",");for(i=1;i<=NF;i++)$i=b[i]*
$i}1' OFS=, cleaned_"$1" cleaned_"$2" > mul_result.csv
}

clean_matrix "$1"
clean_matrix "$2"
if [ "$3" = "add" ]; then
    add_matrices "$1" "$2"
elif [ "$3" = "sub" ]; then
    subtract_matrices "$1" "$2"
fi
```

```
elif [ "$3" = "mul" ]; then
    multiply_matrices "$1" "$2"
elif [ "$3" = "help" ]; then
    print_help
else
    echo "Invalid operation. Please choose 'add', 'sub', or
'mul'."
    print_help
fi
```