오픈소스기초설계

[셸프로그래밍] 20211577 김호준

1. 3-0.sh

- 출력 결과 스크린샷

```
hojun@hojun-VMware-Virtual-Platform:~/opensource$ ./3-0.sh
Hello shell
```

- 코드

```
# 3-0.sh
#!/bin/bash

grep -qFx 'export MYENV="Hello Shell"' ~/.bashrc || echo 'export MYENV="Hello
Shell"' >> ~/.bashrc

source ~/.bashrc
bash -c 'echo $MYENV'

sed -i '\#export MYENV="Hello Shell"#d' ~/.bashrc
unset MYENV
bash -c 'echo $MYENV'
```

2. 3-1.sh

- 출력 결과 스크린샷

```
hojun@hojun-VMware-Virtual-Platform:~/opensource$ vi 3-1.sh
hojun@hojun-VMware-Virtual-Platform:~/opensource$ ./3-1.sh
두 개의 숫자: 12 4
입력: num1=12, num2=4
12 + 4 = 16
12 - 4 = 8
12 * 4 = 48
12 / 4 = 3
hojun@hojun-VMware-Virtual-Platform:~/opensource$
```

- 코드

```
- #!/bin/bash
-
- read -p "두 개의 숫자: " num1 num2
- if [ -z "$num1" ] || [ -z "$num2" ]; then
-     echo "error: 두 개의 숫자 입력."
-     exit 1
- fi
-
- echo "입력: num1=$num1, num2=$num2"
-
- #덧셈
- sum=$(($num1 + $num2))
- echo "$num1 + $num2 = $sum"
-
- # 뺄셈
- diff=$(($num1 - $num2))
- echo "$num1 - $num2 = $diff"
-
- # 곱셈
- prod=$(($num1 * $num2))
- echo "$num1 * $num2 = $prod"
-
- if [ $num2 -eq 0 ]; then
-     echo "$num1 / $num2 = 0, error:divide by zero"
- else
-     quot=$(($num1 / $num2))
-     echo "$num1 / $num2 = $quot"
- fi
```

3. 3-2.sh

- 출력 결과 스크린샷

```
hojun@hojun-VMware-Virtual-Platform:~/opensource$ vi 3-2.sh
hojun@hojun-VMware-Virtual-Platform:~/opensource$ ./3-2.sh
두 개 이상의 숫자 입력: 2 3.5 7
y=1/2*x^2
x = 2 => y = 2.0
x = 3.5 => y = 6.125
x = 7 => y = 24.5
hojun@hojun-VMware-Virtual-Platform:~/opensource$
```

- 코드

```
#3-2.sh
#!/bin/bash

read -p "두 개 이상의 숫자 입력: " -a x_array

count=${#x_array[@]} # 배열의 총 개수
if [ $count -lt 2 ]; then
    echo "error: 두 개 이상의 숫자를 입."
    exit 1
fi

echo "y=1/2*x^2"

for x in "${x_array[@]}"
do
    y=$(echo "scale=4; 0.5 * $x * $x" | bc)

    echo "x = $x => y = $y"
done
```

4. 3-3.sh

- 출력 결과 스크린샷

```
hojun@hojun-VMware-Virtual-Platform:~/opensource$ vi 3-3.sh
hojun@hojun-VMware-Virtual-Platform:~/opensource$ ./3-3.sh
점수 입력 : 100 90 80 70
각각 등급
점수: 100  =>  등급: A
점수: 90  =>  등급: A
점수: 80  =>  등급: B
점수: 70  =>  등급: B
평균 등급
총점: 340
과목 수: 4
평균 점수: 85
평균 등급: B
hojun@hojun-VMware-Virtual-Platform:~/opensource$
```

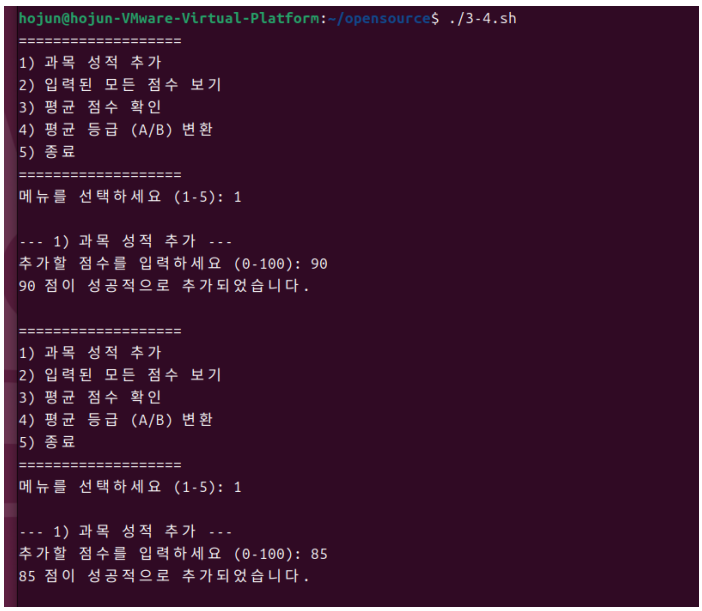
- 코드

```
- #3-3.sh
- #!/bin/bash
-
- read -p "점수 입력 : " -a score_array
-
- count=${#score_array[@]}
- if [ $count -lt 2 ]; then
-     echo "error: 두 개 이상의 점수를 입력."
-     exit 1
- fi
-
- total_sum=0
-
- echo "각각 등급"
-
- for score in "${score_array[@]}"
- do
-     if ! [[ "$score" =~ ^[0-9]+$ ]] || [ $score -lt 0 ] || [ $score -gt
  100 ]; then
-         echo "error: '$score' - 0<= score <=100"
-         exit 1
-     fi
-
-     if [ $score -ge 90 ]; then
-         grade="A"
-     else
-         grade="B"
```

```
-       fi
-
-       echo "점수: $score  =>  등급: $grade"
-       total_sum=$(($total_sum + $score))
-   done
-
-   echo "평균 등급"
-   avg_score=$(($total_sum / $count))
-
-   echo "총점: $total_sum"
-   echo "과목 수: $count"
-   echo "평균 점수: $avg_score"
-   if [ $avg_score -ge 90 ]; then
-       avg_grade="A"
-   else
-       avg_grade="B"
-   fi
-
-   echo "평균 등급: $avg_grade"
-
```

5. 3-4.sh

- 출력 결과 스크린샷



```
==================
1) 과목 성적 추가
2) 입력된 모든 점수 보기
3) 평균 점수 확인
4) 평균 등급 (A/B) 변환
5) 종료
==================
메뉴를 선택하세요 (1-5): 2

--- 2) 입력된 모든 점수 보기 ---
총 2개 점수: 90 85

==================
1) 과목 성적 추가
2) 입력된 모든 점수 보기
3) 평균 점수 확인
4) 평균 등급 (A/B) 변환
5) 종료
==================
메뉴를 선택하세요 (1-5): 3

--- 3) 평균 점수 확인 ---
총점: 175
과목 수: 2
평균 점수 (정수): 87

==================
1) 과목 성적 추가
2) 입력된 모든 점수 보기
3) 평균 점수 확인
4) 평균 등급 (A/B) 변환
5) 종료
==================
메뉴를 선택하세요 (1-5): 4

--- 4) 평균 등급 (A/B) 변환 ---
평균 점수 87 는 등급으로 'B' 입니다.
```

```
====================
메뉴를 선택하세요 (1-5): 5

프로그램을 종료합니다.
```

- 코드

```
- #3-4.sh
- #!/bin/bash
-
- scores=()
-
- calculate_average() {
-     count=${#scores[@]}
-
```

```
-       if [ $count -eq 0 ]; then
-           echo "error: 입력된 점수가 없음."
-           return 1 # 실패 반환
-       fi
-
-       total_sum=0
-       for s in "${scores[@]}"; do
-           total_sum=$(($total_sum + $s))
-       done
-
-       avg_score=$(($total_sum / $count))
-
-       return 0
-   }
-
-   while true
-   do
-       # --- 1. 메뉴 출력 ---
-       echo "===================="
-       echo "1) 과목 성적 추가"
-       echo "2) 입력된 모든 점수 보기"
-       echo "3) 평균 점수 확인"
-       echo "4) 평균 등급 (A/B) 변환"
-       echo "5) 종료"
-       echo "===================="
-
-       read -p "메뉴를 선택하세요 (1-5): " choice
-       echo ""
-
-       case $choice in
-           1)
-           echo "--- 1) 과목 성적 추가 ---"
-               read -p "추가할 점수를 입력하세요 (0-100): " new_score
-
-           if ! [[ "$new_score" =~ ^[0-9]+$ ]] || [ $new_score -lt 0 ] ||
    [ $new_score -gt 100 ]; then
-                   echo "오류: '$new_score' - 점수는 0 에서 100 사이의 정수여야
    합니다."
-               else
-               scores+=($new_score)
-                   echo "$new_score 점이 성공적으로 추가되었습니다."
-               fi
```

```
-           ;;
-       2)
-           echo "--- 2) 입력된 모든 점수 보기 ---"
-               count=${#scores[@]}
-               if [ $count -eq 0 ]; then
-                   echo "입력된 점수가 없습니다."
-               else
-                   # ${scores[@]} : 배열의 모든 요소를 출력
-                   echo "총 ${count}개 점수: ${scores[@]}"
-               fi
-           ;;
-       3)
-           echo "--- 3) 평균 점수 확인 ---"
-               # 헬퍼 함수 호출, 'if'로 성공(0) 여부 확인
-               if calculate_average; then
-                   echo "총점: $total_sum"
-                   echo "과목 수: $count"
-                   echo "평균 점수 (정수): $avg_score"
-               fi
-           ;;
-       4)
-           echo "--- 4) 평균 등급 (A/B) 변환 ---"
-               if calculate_average; then
-                   if [ $avg_score -ge 90 ]; then
-                       avg_grade="A"
-                   else
-                       avg_grade="B"
-                   fi
-                   echo "평균 점수 $avg_score 는 등급으로 '$avg_grade'
    입니다."
-               fi
-               ;;
-       5)
-           echo "프로그램을 종료합니다."
-               break
-               ;;
-       *)
-           echo "잘못된 입력입니다. 1 에서 5 사이의 숫자를 입력하세요."
-               ;;
-       esac
-
-       echo ""
```

```
done
```

6. 3-5.sh

- 출력 결과 스크린샷

```
hojun@hojun-VMware-Virtual-Platform:~/opensource$ ./3-5.sh
스크립트에 전달된 인자:

--- eval로 실행될 명령어 문자열 ---
ls
3-0.sh  3-1.sh  3-2.sh  3-3.sh  3-4.sh  3-5.sh
```

- 코드

```
#3-5.sh
#!/bin/bash

run_ls_with_options() {
    local command_to_run="ls $@"

    echo "--- eval로 실행될 명령어 문자열 ---"
        echo "$command_to_run"

    eval $command_to_run
}

echo "스크립트에 전달된 인자: $@"
echo ""

run_ls_with_options "$@"
```

7. 3-6.sh

- 출력 결과 스크린샷

```
hojun@hojun-VMware-Virtual-Platform:~/opensource$ ./3-6.sh
전달할 2개 이상의 인자를 입력: hello world shell script
[셸] Python(3-6.py)를 호출
[셸] Python에 전달할 인자: hello world shell script
--- [Python] 실행 파일 시작 ---
[Python] 수신된 인자 리스트: ['hello', 'world', 'shell', 'script']
[Python] 수신된 인자 개수: 4
--- [Python] 실행 파일 종료 ---
--- [셸] 3-6.sh 종료 ---
hojun@hojun-VMware-Virtual-Platform:~/opensource$
```

- 코드

Python 코드

```python
#!/usr/bin/env python3
import sys

print("--- [Python] 실행 파일 시작 ---")
arguments = sys.argv[1:]

print(f"[Python] 수신된 인자 리스트: {arguments}")
print(f"[Python] 수신된 인자 개수: {len(arguments)}")

print("--- [Python] 실행 파일 종료 ---")
```

shell 코드

```bash
#3-6.sh
#!/bin/bash

read -p "전달할 2개 이상의 인자를 입력: " -a input_array

count=${#input_array[@]}
if [ $count -lt 2 ]; then
    echo "error: 2개 이상의 인자가 필요."
    exit 1
fi

echo "[셸] Python(3-6.py)를 호출"
echo "[셸] Python에 전달할 인자: ${input_array[@]}"
python3 3-6.py "${input_array[@]}"

echo "--- [셸] 3-6.sh 종료 ---"
```

8. 3-7.sh

- 출력 결과 스크린샷

```
hojun@hojun-VMware-Virtual-Platform:~/opensource$ ./3-7.sh
============================
Linux 시스템 상태 확인
============================
1) 사용자 정보 (w)
2) GPU / CPU 사용률 (nvidia-smi / top)
3) 메모리 사용량 (free)
4) 디스크 사용량 (df)
5) 종료
============================
메뉴를 선택하세요 (1-5): 1

--- 1) 사용자 정보 ---
현재 시스템에 접속된 사용자 정보를 표시합니다.
 22:02:32 up  1:36,  1 user,  load average: 0.07, 0.15, 0.15
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU  WHAT
hojun    tty2     -                20:25    1:36분  0.07초  0.07초 /usr/libexec/gnome-session-binary --session=ubuntu
```

```
메뉴를 선택하세요 (1-5): 2

--- 2) GPU / CPU 사용률 ---
[CPU 사용률 (1회 실행)]
```

```
메뉴를 선택하세요 (1-5): 3

--- 3) 메모리 사용량 ---
시스템의 전체, 사용 중, 여유 메모리 (MB/GB 단위)
               총계          사용          여분       공유     버퍼/캐시      가용
메모리:          15Gi        2.4Gi         11Gi       142Mi        2.5Gi         13Gi
스  왑:         4.0Gi           0B        4.0Gi
```

```
메뉴를 선택하세요 (1-5): 4

--- 4) 디스크 사용량 ---
마운트된 파일 시스템별 디스크 사용량 (MB/GB 단위)
파일 시스템     크기  사용  가용 사용% 마운트위치
tmpfs           1.6G  2.1M  1.6G    1% /run
/dev/sda2        98G   35G   59G   38% /
tmpfs           7.8G  106M  7.7G    2% /dev/shm
tmpfs           5.0M  8.0K  5.0M    1% /run/lock
tmpfs           1.6G  136K  1.6G    1% /run/user/1000
/dev/sr0         96M   96M     0  100% /media/hojun/CDROM
/dev/sr1        6.0G  6.0G     0  100% /media/hojun/Ubuntu 24.04.3 LTS amd64
```

- 코드

```
- #3-7.sh
- #!/bin/bash
```

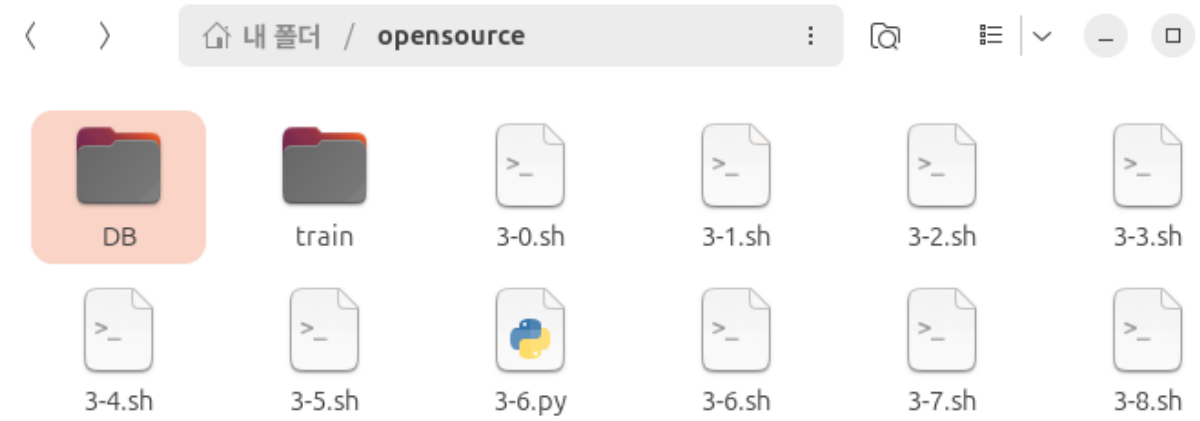
```
-
-  while true
-  do
-      echo "============================="
-      echo "Linux 시스템 상태 확인"
-      echo "============================="
-      echo "1) 사용자 정보 (w)"
-      echo "2) GPU / CPU 사용률 (nvidia-smi / top)"
-      echo "3) 메모리 사용량 (free)"
-      echo "4) 디스크 사용량 (df)"
-      echo "5) 종료"
-      echo "============================="
-
-      read -p "메뉴를 선택하세요 (1-5): " choice
-      echo ""
-
-      case $choice in
-          1)
-              echo "--- 1) 사용자 정보 ---"
-              echo "현재 시스템에 접속된 사용자 정보를 표시합니다."
-              w
-              ;;
-
-          2)
-              echo "--- 2) GPU / CPU 사용률 ---"
-              if command -v nvidia-smi &> /dev/null
-              then
-                  echo "[NVIDIA GPU 사용률]"
-                  nvidia-smi
-              else
-                  echo "[CPU 사용률 (1회 실행)]"
-                  top -b -n 1 | grep "Cpu(s)"
-              fi
-              ;;
-
-          3)
-              echo "--- 3) 메모리 사용량 ---"
-              echo "시스템의 전체, 사용 중, 여유 메모리 (MB/GB 단위)"
-              free -h
-              ;;
-
-          4)
```

```
-               echo "--- 4) 디스크 사용량 ---"
-               echo "마운트된 파일 시스템별 디스크 사용량 (MB/GB 단위)"
-               df -h
-               ;;
-
-           5)
-               echo "프로그램을 종료합니다."
-               break
-               ;;
-
-           *)
-               echo "잘못된 입력입니다. 1 에서 5 사이의 숫자를 입력하세요."
-               ;;
-       esac
-       echo ""
-
-   done
```

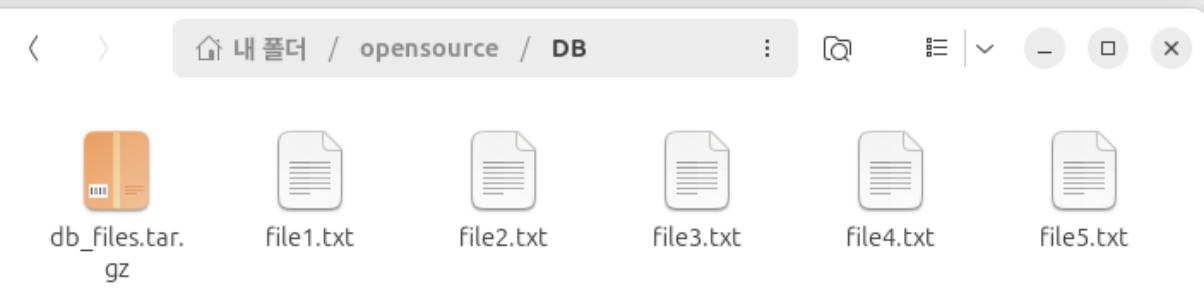
9. 3-8.sh

- 출력 결과 스크린샷

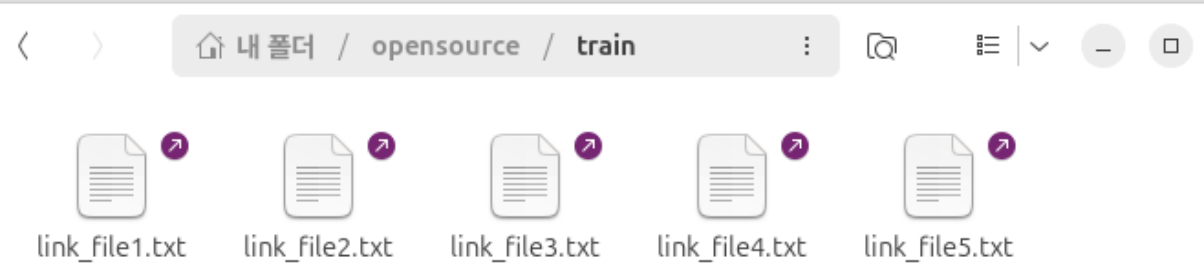
```
hojun@hojun-VMware-Virtual-Platform:~/opensource$ vi 3-8.sh
hojun@hojun-VMware-Virtual-Platform:~/opensource$ ./3-8.sh
--- 1. 'DB' 폴더 확인 및 생성 ---
'DB' 폴더가 준비되었습니다.

DB 폴더에 5개 파일 생성 중 (file1.txt ~ file5.txt)...
DB 폴더 내에서 파일 압축 중 (db_files.tar.gz)...
file1.txt
file2.txt
file3.txt
file4.txt
file5.txt
압축 완료.

--- 3. 'train' 폴더 생성 및 5개 파일 링크 ---
'train' 폴더에 5개 원본 파일의 심볼릭 링크 생성...
[DB 폴더 내용]
합계 24
-rw-rw-r-- 1 hojun hojun 226 10월  29 22:11 db_files.tar.gz
-rw-rw-r-- 1 hojun hojun  36 10월  29 22:11 file1.txt
-rw-rw-r-- 1 hojun hojun  36 10월  29 22:11 file2.txt
-rw-rw-r-- 1 hojun hojun  36 10월  29 22:11 file3.txt
-rw-rw-r-- 1 hojun hojun  36 10월  29 22:11 file4.txt
-rw-rw-r-- 1 hojun hojun  36 10월  29 22:11 file5.txt

[train 폴더 내용]
합계 0
lrwxrwxrwx 1 hojun hojun 35 10월  29 22:11 link_file1.txt -> /home/hojun/opensource/DB/file1.txt
lrwxrwxrwx 1 hojun hojun 35 10월  29 22:11 link_file2.txt -> /home/hojun/opensource/DB/file2.txt
lrwxrwxrwx 1 hojun hojun 35 10월  29 22:11 link_file3.txt -> /home/hojun/opensource/DB/file3.txt
lrwxrwxrwx 1 hojun hojun 35 10월  29 22:11 link_file4.txt -> /home/hojun/opensource/DB/file4.txt
lrwxrwxrwx 1 hojun hojun 35 10월  29 22:11 link_file5.txt -> /home/hojun/opensource/DB/file5.txt

[링크된 파일 내용 확인 (예: link_file1.txt)]
이것은 file1의 내용입니다.
hojun@hojun-VMware-Virtual-Platform:~/opensource$
```



DB 파일 내에 파일



Train에 링크 된 파일



- 코드

```
- #3-8.sh
- #!/bin/bash
-
- echo "--- 1. 'DB' 폴더 확인 및 생성 ---"
- mkdir -p "DB"
- echo "'DB' 폴더가 준비되었습니다."
- echo ""
-
- echo "DB 폴더에 5 개 파일 생성 중 (file1.txt ~ file5.txt)..."
- for i in {1..5}
- do
-     echo "이것은 file${i}의 내용입니다." > "DB/file${i}.txt"
- done
-
- echo "DB 폴더 내에서 파일 압축 중 (db_files.tar.gz)..."
- tar -czvf "DB/db_files.tar.gz" -C "DB" file1.txt file2.txt file3.txt
  file4.txt file5.txt
- echo "압축 완료."
- echo ""
-
- echo "--- 3. 'train' 폴더 생성 및 5 개 파일 링크 ---"
- mkdir -p "train"
-
```

```
-   CURRENT_DIR=$(pwd)
-   echo "'train' 폴더에 5 개 원본 파일의 심볼릭 링크 생성..."
-   for i in {1..5}
-   do
-       local_target="$CURRENT_DIR/DB/file${i}.txt"
-       local_link="train/link_file${i}.txt"
-       ln -sf "$local_target" "$local_link"
-   done
-
-   echo "[DB 폴더 내용]"
-   ls -l DB
-   echo ""
-   echo "[train 폴더 내용]"
-   ls -l train
-   echo ""
-   echo "[링크된 파일 내용 확인 (예: link_file1.txt)]"
-   cat "train/link_file1.txt"
```

10. 3-9.sh

- 출력 결과 스크린샷

```
hojun@hojun-VMware-Virtual-Platform:~/opensource$ ./3-9.sh
==================
1) 팀원 정보 추가
2) 팀원과 한 일 기록
3) 팀원 검색 (이름)
4) 수행 내용 검색 (날짜/키워드)
5) 종료
==================
메뉴를 선택하세요 (1-5): 1

--- 1) 팀원 정보 추가 ---
추가할 팀원 이름: 김호준
생일 또는 전화번호: 010-1234-5678
[김호준] 님의 정보를 DB.txt 에 저장했습니다.
==================
1) 팀원 정보 추가
2) 팀원과 한 일 기록
3) 팀원 검색 (이름)
4) 수행 내용 검색 (날짜/키워드)
5) 종료
==================
메뉴를 선택하세요 (1-5): 2

--- 2) 팀원과 한 일 기록 ---
날짜 (기본값: 2025-10-29):
수행한 일 내용: shell study
[2025-10-29] 활동을 DB.txt 에 저장했습니다.
```

```
메뉴를 선택하세요 (1-5): 3

--- 3) 팀원 검색 (이름) ---
검색할 팀원 이름: 김호준

[김호준] (으)로 검색된 팀원 정보:
MEMBER | 김호준 | 010-1234-5678
====================
1) 팀원 정보 추가
2) 팀원과 한 일 기록
3) 팀원 검색 (이름)
4) 수행 내용 검색 (날짜/키워드)
5) 종료
====================
메뉴를 선택하세요 (1-5): 4

--- 4) 수행 내용 검색 (날짜/키워드) ---
검색할 날짜(YYYY-MM-DD) 또는 키워드: 2025-10-29

[2025-10-29] (으)로 검색된 수행 내용:
LOG | 2025-10-29 | shell study
====================
1) 팀원 정보 추가
2) 팀원과 한 일 기록
3) 팀원 검색 (이름)
4) 수행 내용 검색 (날짜/키워드)
5) 종료
====================
메뉴를 선택하세요 (1-5): 5

프로그램을 종료합니다.
```

- 코드

```
- #3-9.sh
- #!/bin/bash
-
- DB_FILE="DB.txt"
- touch "$DB_FILE"
-
- while true
- do
```

```
-       echo "==================="
-       echo "1) 팀원 정보 추가"
-       echo "2) 팀원과 한 일 기록"
-       echo "3) 팀원 검색 (이름)"
-       echo "4) 수행 내용 검색 (날짜/키워드)"
-       echo "5) 종료"
-       echo "==================="
-       read -p "메뉴를 선택하세요 (1-5): " choice
-       echo ""
-
-       case $choice in
-           1)
-               echo "--- 1) 팀원 정보 추가 ---"
-               read -p "추가할 팀원 이름: " name
-               read -p "생일 또는 전화번호: " info
-
-               echo "MEMBER | $name | $info" >> "$DB_FILE"
-               echo "[$name] 님의 정보를 $DB_FILE 에 저장했습니다."
-               ;;
-
-           2)
-               echo "--- 2) 팀원과 한 일 기록 ---"
-               default_date=$(date +%Y-%m-%d)
-               read -p "날짜 (기본값: $default_date): " log_date
-
-               if [ -z "$log_date" ]; then
-                   log_date=$default_date
-               fi
-
-               read -p "수행한 일 내용: " activity
-
-               echo "LOG | $log_date | $activity" >> "$DB_FILE"
-               echo "[$log_date] 활동을 $DB_FILE 에 저장했습니다."
-               ;;
-
-           3)
-               echo "--- 3) 팀원 검색 (이름) ---"
-               read -p "검색할 팀원 이름: " search_name
-               echo ""
-               echo "[$search_name] (으)로 검색된 팀원 정보:"
-               result=$(grep "^MEMBER" "$DB_FILE" | grep -i "$search_name")
```

```
-
-             if [ -z "$result" ]; then
-                 echo "일치하는 팀원 정보가 없습니다."
-             else
-                 echo "$result"
-             fi
-             ;;
-
-         4)
-             echo "--- 4) 수행 내용 검색 (날짜/키워드) ---"
-             read -p "검색할 날짜(YYYY-MM-DD) 또는 키워드: " search_keyword
-             echo ""
-             echo "[$search_keyword] (으)로 검색된 수행 내용:"
-
-             result=$(grep "^LOG" "$DB_FILE" | grep -i "$search_keyword")
-
-             if [ -z "$result" ]; then
-                 echo "일치하는 수행 내용이 없습니다."
-             else
-                 echo "$result"
-             fi
-             ;;
-
-         5)
-             echo "프로그램을 종료합니다."
-             break
-             ;;
-
-         *)
-             echo "잘못된 입력입니다. 1 에서 5 사이의 숫자를 입력하세요."
-             ;;
-     esac
-
- done
```