

```
suyeon@suyeonpc:~/opensource/src$ cat Ex3-0.sh
#!/bin/bash
# Ex3-0.sh
BASHRC="$HOME/.bashrc"
VARLINE='export MYENV="Hello Shell"

echo "===" Ex3-0: 환경변수 등록/유지/해제 실습 ==="

echo "[1] .bashrc 등록 여부 확인"
if ! grep -q 'export MYENV="Hello Shell"' "$BASHRC"; then
    echo "$VARLINE" >> "$BASHRC"
    echo "-> .bashrc에 MYENV 추가 완료"
else
    echo "-> 이미 .bashrc에 MYENV 있음"
fi

echo
echo "[2] 현재 쉘에서 source로 반영"
source "$BASHRC"
echo "현재 MYENV 값: $MYENV"

echo
echo "[3] 새 쉘에서도 유지되는지 확인"
bash -lc 'echo "서브쉘 MYENV 값: $MYENV"'
```

```
echo
echo "[4] 환경변수 해제(unset)"
unset MYENV
echo "unset 후 MYENV 값: $MYENV"

echo
echo "[5] 서브쉘에서 다시 확인(해제하면 유지 x)"
bash -lc 'unset MYENV; echo "서브쉘 unset 후 MYENV 값: $MYENV"'
```

```
suyeon@suyeonpc:~/opensource/src$
```

```
suyeon@suyeonpc:~/opensource/src$ ./Ex3-0.sh  
== Ex3-0: 환경변수 등록/유지/해제 실습 ==  
[1] .bashrc 등록 여부 확인  
-> .bashrc에 MYENV 추가 완료
```

[2] 현재 쉘에서 source로 반영

현재 MYENV 값:

[3] 새 쉘에서도 유지되는지 확인

서브쉘 MYENV 값:

[4] 환경변수 해제(unset)

unset 후 MYENV 값:

[5] 서브쉘에서 다시 확인(해제하면 유지 X)

서브쉘 unset 후 MYENV 값:

```
suyeon@suyeonpc:~/opensource/src$
```

cat Ex3-0.sh

./Ex3-0.sh

- 1) ~/.bashrc에 export MYENV="Hello Shell"을 추가해 환경변수를 등록했다.
- 2) source ~/.bashrc로 현재 쉘에 반영하고, 서브쉘에서도 값이 유지되는지 확인했다.
- 3) unset MYENV로 해제 후 새 쉘에서는 값이 유지되지 않음을 출력으로 검증했다.

```
suyeon@suyeonpc:~/opensource/src$ cat Ex3-1.sh
#!/bin/bash
# Ex3-1.sh
if [ $# -ne 2 ]; then
    echo "사용법: $0 num1 num2"
    exit 1
fi
a=$1
b=$2
echo "a = $a, b = $b"
echo "a + b = $((a + b))"
echo "a - b = $((a - b))"
echo "a * b = $((a * b))"
if [ $b -ne 0 ]; then
    echo "a / b = $((a / b))"
else
    echo "a / b = (0으로 나눌 수 없음)"
fi
suyeon@suyeonpc:~/opensource/src$
```

```
suyeon@suyeonpc:~/opensource/src$ ./Ex3-1.sh
사용법: ./Ex3-1.sh num1 num2
suyeon@suyeonpc:~/opensource/src$
```

cat Ex3-1.sh

./Ex3-1.sh

- 1) 두 개의 정수 인자를 \$1, \$2로 받아 산술 연산을 수행하도록 했다.
- 2) \$() 정수 계산 문법으로 덧셈/뺄셈/곱셈/나눗셈을 출력했다.
- 3) 0으로 나누는 경우를 예외 처리해 오류 대신 안내 메시지를 출력했다.

```
suyeon@suyeonpc:~/opensource/src$ cat Ex3-2.sh
#!/bin/bash
# Ex3-2.sh
echo "==== Ex3-2: Y = 1/2 x^2 계산 ===="
echo "x 값을 2개 이상 입력하세요. 종료하려면 q"
count=0
while true; do
    read -p "x 입력: " x
    if [ "$x" = "q" ]; then
        break
    fi
    y=$(echo "0.5 * $x * $x" | bc -l)
    echo "x = $x -> y = $y"
    count=$((count + 1))
done
if [ $count -lt 2 ]; then
    echo "경고: 2개 이상 입력해야 과제 조건 만족!"
fi
suyeon@suyeonpc:~/opensource/src$
```

```
suyeon@suyeonpc:~/opensource/src$ ./Ex3-2.sh
==== Ex3-2: Y = 1/2 x^2 계산 ====
x 값을 2개 이상 입력하세요. 종료하려면 q
x 입력: 4
x = 4 -> y = 8.0
x 입력:
```

cat Ex3-2.sh

./Ex3-2.sh

- 1) 사용자에게 x 값을 반복 입력받아  $Y=1/2 x^2$ 를 계산하도록 구성했다.
- 2) bc -l을 사용해 실수 범위 계산이 가능하게 했고, 입력마다 결과를 출력했다.
- 3) 최소 2개 이상 입력해야 조건을 만족하므로, 입력 개수를 세어 부족하면 경고했다.

```

suyeon@suyeonpc:~/opensource/src$ cat Ex3-3.sh
#!/bin/bash
# Ex3-3.sh
scores=()
echo "==== Ex3-3: 점수 -> 등급(A/B) 변환 ==="
echo "점수 입력(0~100), 종료: q"
while true; do
    read -p "점수 입력: " s
    if [ "$s" = "q" ]; then break; fi
    if [ $s -lt 0 ] || [ $s -gt 100 ]; then
        echo "0~100 범위만 입력!"
        continue
    fi
    scores+=($s)
done
n=${#scores[@]}
if [ $n -lt 2 ]; then
    echo "과목 2개 이상 입력해야 합니다."
    exit 1
fi
sum=0
echo "----- 각 과목 등급 -----"
for s in "${scores[@]}"; do
    if [ $s -ge 90 ]; then grade="A"; else grade="B"; fi
    echo "점수 $s -> 등급 $grade"
    sum=$((sum + s))
done
avg=$((sum / n))

echo "----- 각 과목 등급 -----"
for s in "${scores[@]}"; do
    if [ $s -ge 90 ]; then grade="A"; else grade="B"; fi
    echo "점수 $s -> 등급 $grade"
    sum=$((sum + s))
done
avg=$((sum / n))
echo "-----"
echo "평균 점수: $avg"
if [ $avg -ge 90 ]; then avg_grade="A"; else avg_grade="B"; fi
echo "평균 등급: $avg_grade"
suyeon@suyeonpc:~/opensource/src$ 

```

```
suyeon@suyeonpc:~/opensource/src$ ./Ex3-3.sh
==== Ex3-3: 점수 -> 등급(A/B) 변환 ====
점수 입력(0~100), 종료: q
점수 입력: 86
점수 입력: 76
점수 입력: 99
점수 입력: 100
점수 입력: 3
점수 입력: q
----- 각 과목 등급 -----
점수 86 -> 등급 B
점수 76 -> 등급 B
점수 99 -> 등급 A
점수 100 -> 등급 A
점수 3 -> 등급 B
-----
평균 점수: 72
평균 등급: B
suyeon@suyeonpc:~/opensource/src$
```

cat Ex3-3.sh

./Ex3-3.sh

- 1) 여러 과목 점수를 입력받아 90 이상은 A, 그 미만은 B로 변환했다.
- 2) 각 점수의 등급을 출력하고, 전체 평균 점수를 계산했다.
- 3) 평균 점수도 동일 기준으로 A/B 등급으로 변환해 평균 등급을 출력했다.

```
suyeon@suyeonpc:~/opensource/src$ cat Ex3-4.sh
#!/bin/bash
# Ex3-4.sh
scores=()
to_gpa() {
    local s=$1
    if [ $s -ge 90 ]; then echo 4.0
    elif [ $s -ge 80 ]; then echo 3.0
    elif [ $s -ge 70 ]; then echo 2.0
    elif [ $s -ge 60 ]; then echo 1.0
    else echo 0.0
    fi
}
while true; do
    echo "===="
    echo "1. 과목 성적 추가"
    echo "2. 입력된 모든 점수 보기"
    echo "3. 평균 점수 확인"
    echo "4. 평균 등급 (GPA) 변환"
    echo "5. 종료"
    echo "===="
    read -p "메뉴 선택: " menu
    case $menu in
        1)
            read -p "추가할 점수(0~100): " s
            if [ $s -lt 0 ] || [ $s -gt 100 ]; then
                echo "범위 오류!"
            else
                scores+=($s)
            fi
        ;;
        2)
            echo ${scores[@]}
        ;;
        3)
            average=$(echo ${scores[@]} | awk '{sum=0; for(i=1;i<@+1;i++) sum+=\$i; print sum/@}')
            echo "평균 점수: $average"
        ;;
        4)
            gpa=$(echo ${scores[@]} | awk '{sum=0; for(i=1;i<@+1;i++) sum+=\$i; print sum/@; print "GPA: " sum/@}')
            echo "평균 등급 (GPA): $gpa"
        ;;
        5)
            exit 0
        ;;
        *)
            echo "선택 번호를 다시 확인하세요."
        ;;
    esac
done
```

```
else
    scores+=($s)
    echo "추가 완료."
fi
;;
2)
if [ ${#scores[@]} -eq 0 ]; then
    echo "점수 없음"
else
    echo "점수 목록: ${scores[@]}"
fi
;;
3)
if [ ${#scores[@]} -eq 0 ]; then
    echo "점수 없음"
else
    sum=0
    for s in "${scores[@]}"; do sum=$((sum + s)); done
    avg=$((sum / ${#scores[@]}))
    echo "평균 점수: $avg"
fi
;;
4)
if [ ${#scores[@]} -eq 0 ]; then
    echo "점수 없음"
else
    sum=0
    for s in "${scores[@]}"; do sum=$((sum + s)); done
```

```
4)
if [ ${#scores[@]} -eq 0 ]; then
    echo "점수 없음"
else
    sum=0
    for s in "${scores[@]}"; do sum=$((sum + s)); done
    avg=$((sum / ${#scores[@]}))
    gpa=$(to_gpa $avg)
    echo "평균 점수: $avg -> GPA: $gpa"
fi
;;
5)
echo "종료합니다."
break
;;
*)
echo "1~5 중 선택!"
;;
esac
echo
done
suyeon@suyeonpc:~/opensource/src$
```

```
suyeon@suyeonpc:~/opensource/src$ ./Ex3-4.sh
```

```
=====
```

1. 과목 성적 추가
2. 입력된 모든 점수 보기
3. 평균 점수 확인
4. 평균 등급 (GPA) 변환
5. 종료

```
=====
```

```
메뉴 선택: 1
```

```
추가할 점수(0~100): 33 44 55 66
```

```
||
```

```
./Ex3-4.sh: 행 25번: [: 인자가 너무 많음
```

```
./Ex3-4.sh: 행 25번: [: 인자가 너무 많음
```

```
추가 완료.
```

```
=====
```

1. 과목 성적 추가
2. 입력된 모든 점수 보기
3. 평균 점수 확인
4. 평균 등급 (GPA) 변환
5. 종료

```
=====
```

```
메뉴 선택: 2
```

```
점수 목록: 33 44 55 66
```

```
=====
```

1. 과목 성적 추가
2. 입력된 모든 점수 보기

```
2. 입력된 모든 점수 보기
3. 평균 점수 확인
4. 평균 등급 (GPA) 변환
5. 종료
=====
메뉴 선택: 3
평균 점수: 49

=====
1. 과목 성적 추가
2. 입력된 모든 점수 보기
3. 평균 점수 확인
4. 평균 등급 (GPA) 변환
5. 종료
=====
메뉴 선택: 4
평균 점수: 49 -> GPA: 0.0

=====
1. 과목 성적 추가
2. 입력된 모든 점수 보기
3. 평균 점수 확인
4. 평균 등급 (GPA) 변환
5. 종료
=====
메뉴 선택: 5
종료합니다.
```

```
cat Ex3-4.sh
```

```
./Ex3-4.sh
```

- 1) 1~5 메뉴를 반복 출력하고 case문으로 선택 기능을 분기했다.
- 2) 성적 추가/전체 보기/평균 계산/평균 GPA 변환 기능을 배열과 반복문으로 구현했다.
- 3) 5번 입력 전까지 계속 동작하도록 while true 루프를 사용했다.

```
suyeon@suyeonpc:~/opensource/src$ cat Ex3-5.sh
#!/bin/bash
# Ex3-5.sh
run_cmd() {
    local cmd=$1
    shift
    local opts="$@"
    echo "실행: $cmd $opts"
    eval "$cmd $opts"
}
run_cmd ls "$@"
suyeon@suyeonpc:~/opensource/src$
```

```
suyeon@suyeonpc:~/opensource/src$ ./Ex3-5.sh
실행: ls
Ex3-0.sh  Ex3-2.sh  Ex3-4.sh  Ex3-6.sh  Ex3-8.sh  args_app.py
Ex3-1.sh  Ex3-3.sh  Ex3-5.sh  Ex3-7.sh  Ex3-9.sh  test1
suyeon@suyeonpc:~/opensource/src$
```

cat Ex3-5.sh

./Ex3-5.sh

- 1) 내부 함수 run\_cmd()를 만들어 리눅스 명령어를 실행하도록 했다.
- 2) 입력받은 인자를 옵션 문자열로 조합해 함수 내부에서 전달했다.
- 3) eval을 이용해 동적으로 ls 옵션 형태의 명령이 실행되도록 구성했다.

```
suyeon@suyeonpc:~/opensource/src$ cat Ex3-6.sh
#!/bin/bash
# Ex3-6.sh
if [ $# -lt 2 ]; then
    echo "사용법: $0 arg1 arg2 [arg3 ...]"
    exit 1
fi
echo "셸에서 실행파일 시작"
./args_app.py "$@"
echo "셸에서 실행파일 종료"
suyeon@suyeonpc:~/opensource/src$
```

```
suyeon@suyeonpc:~/opensource/src$ ./Ex3-6.sh
사용법: ./Ex3-6.sh arg1 arg2 [arg3 ...]
suyeon@suyeonpc:~/opensource/src$
```

```
cat Ex3-6.sh
```

```
./Ex3-6.sh
```

- 1) 2개 이상 인자를 받아 시작/종료/인자 목록을 출력하는 Python 실행파일을 작성했다.
- 2) 셸 스크립트에서 그 실행파일을 호출하며 입력 인자를 그대로 넘겼다.
- 3) 셸에서 프로그램 시작/종료 메시지를 추가로 출력해 실행 흐름을 명확히 했다.

```
suyeon@suyeonpc:~/opensource/src$ cat Ex3-7.sh
#!/bin/bash
# Ex3-7.sh
while true; do
    echo "=====
    echo "1. 사용자 정보"
    echo "2. GPU 또는 CPU 사용률"
    echo "3. 메모리 사용량"
    echo "4. 디스크 사용량"
    echo "5. 종료"
    echo "=====
    read -p "메뉴 선택: " menu
    case $menu in
        1)
            echo "[사용자 정보]"
            whoami
            id
            uptime
            ;;
        2)
            echo "[GPU/CPU 사용률]"
            if command -v nvidia-smi >/dev/null 2>&1; then
                nvidia-smi
            else
                top -bn1 | head -20
            fi
            ;;
        3)
            exit
    esac
done
```

```
suyeon@suyeonpc:~/opensource/src$ ./Ex3-7.sh
=====
1. 사용자 정보
2. GPU 또는 CPU 사용률
3. 메모리 사용량
4. 디스크 사용량
5. 종료
=====
메뉴 선택: 1
[사용자 정보]
suyeon
uid=1000(suyeon) gid=1000(suyeon) groups=1000(suyeon),4(adm),24(cdrom),27(sudo),
30(dip),46(plugdev),110(lxd)
08:29:14 up 49 min, 1 user, load average: 0.01, 0.04, 0.07

=====
1. 사용자 정보
2. GPU 또는 CPU 사용률
3. 메모리 사용량
4. 디스크 사용량
5. 종료
=====
메뉴 선택: 2
[GPU/CPU 사용률]
top - 08:29:15 up 50 min, 1 user, load average: 0.01, 0.04, 0.07
작업: 203 총계, 1 실행, 202 대기, 0 멈춤, 0 좀비
%Cpu: 2.9 us, 2.9 sy, 0.0 ni, 94.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB 메모리 : 7833.3 total, 5453.1 free, 797.0 used, 1583.2 buff/cache
```

```
작업: 203 총계, 1 실행, 202 대기, 0 멈춤, 0 종비  
%Cpu: 2.9 us, 2.9 sy, 0.0 ni, 94.2 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st  
MiB 메모리 : 7833.3 total, 5453.1 free, 797.0 used, 1583.2 buff/cache  
MiB 스왑: 4096.0 total, 4096.0 free, 0.0 used. 6797.1 avail 메모리
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
1540	suyeon	20	0	5472772	388508	151056	S	11.8	4.8	2:28.01	gnome-s+
4174	suyeon	20	0	11368	3600	2976	R	5.9	0.0	0:00.02	top
1	root	20	0	168632	11896	7404	S	0.0	0.1	0:02.09	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par+
5	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	slub_fl+
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	netns
8	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker+
10	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_perc+
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tas+
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_tas+
13	root	20	0	0	0	0	S	0.0	0.0	0:00.08	ksoftirq+

```
=====
```

1. 사용자 정보
2. GPU 또는 CPU 사용률
3. 메모리 사용량
4. 디스크 사용량
5. 종료

```
=====
```

```
메뉴 선택: 3
```

```
=====
메뉴 선택: 3
[메모리 사용량]
      총계      사용      여분      공유      버퍼/캐시      가용
메모리:    7.6Gi    796Mi    5.3Gi    61Mi    1.5Gi    6.6Gi
스왑:     4.0Gi      0B    4.0Gi
```

- ```
=====
1. 사용자 정보
2. GPU 또는 CPU 사용률
3. 메모리 사용량
4. 디스크 사용량
5. 종료
```

```
=====
메뉴 선택: 4
[디스크 사용량]
파일 시스템          크기   사용   가용   사용%   마운트위치
tmpfs                  784M   1.8M   782M   1%   /run
/dev/mapper/ubuntu--vg-ubuntu--lv  30G   13G   16G   46%   /
tmpfs                  3.9G   0     3.9G   0%   /dev/shm
tmpfs                  5.0M   4.0K   5.0M   1%   /run/lock
/dev/vda2               2.0G   129M   1.7G   8%   /boot
/dev/vda1               1.1G   6.4M   1.1G   1%   /boot/efi
tmpfs                  784M   108K   784M   1%   /run/user/1000
```

- ```
=====
1. 사용자 정보
2. GPU 또는 CPU 사용률
```

```
=====
1. 사용자 정보
2. GPU 또는 CPU 사용률
3. 메모리 사용량
4. 디스크 사용량
5. 종료
```

```
=====
메뉴 선택: 5
종료합니다.
```

```
3)
    echo "[메모리 사용량]"
    free -h
    ;;
4)
    echo "[디스크 사용량]"
    df -h
    ;;
5)
    echo "종료합니다."
    break
    ;;
*)
    echo "1~5 중 선택!"
    ;;
esac
echo
done
suyeon@suyeonpc:~/opensource/src$
```

```
cat Ex3-7.sh
```

```
./Ex3-7.sh
```

- 1) 시스템 상태 확인을 위해 메뉴 기반 셸을 작성했다.
- 2) 사용자 정보(whoami/id), CPU·GPU(top/nvidia-smi), 메모리(free), 디스크(df)를 출력했다.
- 3) 메뉴 선택에 따라 필요한 상태를 보여주고, 종료 전까지 반복 실행되게 했다.

```
suyeon@suyeonpc:~/opensource/src$ cat Ex3-8.sh
#!/bin/bash
# Ex3-8.sh
BASE=$(pwd)
DBDIR="$BASE/DB"
TRAINDIR="$BASE/train"
echo "==== Ex3-8 ===="
if [ ! -d "$DBDIR" ]; then
    mkdir "$DBDIR"
    echo "DB 폴더 생성."
else
    echo "DB 폴더 이미 존재."
fi
cd "$DBDIR"
for i in {1..5}; do
    echo "data file $i" > "data$i.txt"
done
echo "5개 파일 생성 완료."
tar -czf DB_files.tar.gz data*.txt
echo "압축 파일 생성: DB_files.tar.gz"
cd "$BASE"
mkdir -p "$TRAINDIR"
cd "$TRAINDIR"
rm -f data*.txt
for f in "$DBDIR"/data*.txt; do
    ln -s "$f" .
done
echo "train 폴더 링크 완료."  
카카오톡
```

```
done
echo "train 폴더 링크 완료."
ls -l
suyeon@suyeonpc:~/opensource/src$
```

```
suyeon@suyeonpc:~/opensource/src$ ./cat 3-8.sh
bash: ./cat: 그런 파일이나 디렉터리가 없습니다
suyeon@suyeonpc:~/opensource/src$ ./Ex3-8.sh
==== Ex3-8 ====
DB 폴더 생성.
5개 파일 생성 완료.
압축 파일 생성: DB_files.tar.gz
train 폴더 링크 완료.
합계 0
lrwxrwxrwx 1 suyeon suyeon 40 11월 21 08:19 data1.txt -> /home/suyeon/opensource
/src/DB/data1.txt
lrwxrwxrwx 1 suyeon suyeon 40 11월 21 08:19 data2.txt -> /home/suyeon/opensource
/src/DB/data2.txt
lrwxrwxrwx 1 suyeon suyeon 40 11월 21 08:19 data3.txt -> /home/suyeon/opensource
/src/DB/data3.txt
lrwxrwxrwx 1 suyeon suyeon 40 11월 21 08:19 data4.txt -> /home/suyeon/opensource
/src/DB/data4.txt
lrwxrwxrwx 1 suyeon suyeon 40 11월 21 08:19 data5.txt -> /home/suyeon/opensource
/src/DB/data5.txt
suyeon@suyeonpc:~/opensource/src$
```

cat Ex3-8.sh

./Ex3-8.sh

- 1) 현재 경로에 DB 폴더가 없으면 생성하고, 그 안에 5개 파일을 만들었다.
- 2) 생성된 파일들을 tar.gz로 압축하여 백업 형태로 저장했다.
- 3) train 폴더를 만든 뒤 DB의 5개 파일만 심볼릭 링크로 연결해 접근 가능하게 했다.

```
suyeon@suyeonpc:~/opensource/src$ cat Ex3-9.sh
#!/bin/bash
# Ex3-9.sh
DB="DB.txt"
touch "$DB"
add_member() {
    read -p "이름: " name
    read -p "생일(YYYY-MM-DD) 또는 전화번호: " info
    echo "MEMBER|$name|$info" >> "$DB"
    echo "팀원 추가 완료."
}
add_work() {
    read -p "날짜(YYYY-MM-DD): " date
    read -p "팀원 이름: " name
    read -p "수행 내용: " content
    echo "WORK|$date|$name|$content" >> "$DB"
    echo "수행 내용 기록 완료."
}
search_member() {
    read -p "검색할 이름: " name
    echo "[팀원 검색 결과]"
    grep "^MEMBER|$name|" "$DB" || echo "없음"
}
search_work() {
    read -p "검색할 날짜: " date
    echo "[수행 내용 검색 결과]"
    grep "^WORK|$date|" "$DB" || echo "없음"
```

```
echo "[팀원 검색 결과]"
grep "^MEMBER|$name|" "$DB" || echo "없음"
}

search_work() {
    read -p "검색할 날짜: " date
    echo "[수행 내용 검색 결과]"
    grep "^WORK|$date|" "$DB" || echo "없음"
}

while true; do
    echo "====="
    echo "1. 팀원 정보 추가"
    echo "2. 팀원과 한 일 기록"
    echo "3. 팀원 검색"
    echo "4. 수행 내용 검색"
    echo "5. 종료"
    echo "====="
    read -p "메뉴 선택: " menu
    case $menu in
        1) add_member ;;
        2) add_work ;;
        3) search_member ;;
        4) search_work ;;
        5) echo "종료"; break ;;
        *) echo "1~5 중 선택!" ;;
    esac
    echo
done
```

```
suyeon@suyeonpc:~/opensource/src$ ./Ex3-9.sh
=====
1. 팀원 정보 추가
2. 팀원과 한 일 기록
3. 팀원 검색
4. 수행 내용 검색
5. 종료
=====
메뉴 선택: 1
이름: suyeon
생일(YYYY-MM-DD) 또는 전화번호: 01000000000
팀원 추가 완료.

=====
1. 팀원 정보 추가
2. 팀원과 한 일 기록
3. 팀원 검색
4. 수행 내용 검색
5. 종료
=====
메뉴 선택: 2
날짜(YYYY-MM-DD): 2025-11-21
팀원 이름: su
수행 내용: opensource
수행 내용 기록 완료.

=====
```

```
=====
메뉴 선택: 3
검색할 이름: suyeon
[팀원 검색 결과]
MEMBER|suyeon|010000000000

=====
1. 팀원 정보 추가
2. 팀원과 한 일 기록
3. 팀원 검색
4. 수행 내용 검색
5. 종료

=====

메뉴 선택: 4
검색할 날짜: 2025-11-21
[수행 내용 검색 결과]
WORK|2025-11-21|su|opensource

=====

1. 팀원 정보 추가
2. 팀원과 한 일 기록
3. 팀원 검색
4. 수행 내용 검색
5. 종료

=====

메뉴 선택: 5
종료
```

```
suyeon@suyeonpc:~/opensource/src$ 
```

cat Ex3-9.sh

.Ex3-9.sh

1) DB.txt에 팀원 정보와 수행 내용을 구분된 형식(MEMBER/WORK)으로 계속 저장했다.

2) 메뉴를 통해 팀원 추가/활동 기록/이름 검색/날짜 검색 기능을 구현했다.

3) grep으로 조건에 맞는 라인을 찾아 출력해 원하는 정보만 빠르게 조회 가능하게 했다