```
#include <stdio.h>              // standard input ouput HeaderFile, 표준 입출력 헤더파일

#include <stdlib.h>             // standard library HeaderFile, exit(), rand(), system() 함수 등이 담긴 헤더파일.
                                //이번 프로그램에서는 system("cls") 를 사용하기 위하여 사용된 헤더 파일.

#include <string.h>                     // 문자열 헤더파일, 메모리 초기화 함수인 memset을 사용하기 위해 사용한 헤더파일. memset에 대한 설명은 case2 부분에 사용한 부분에 주석을 달아놓음.

#define MAXLINE 100                     // MAXLINE 을 100 으로 치환하는 전처리지시문


// 상품 목록을 관리 할 구조체
typedef struct _Product{
        char    Prod_Name                       [30];   // 제품명
        int             Prod_Value                      ;
// 원가
        int             Prod_DisValue           ;               // 할인가
        int             Prod_Count                      ;
// 수량
        double  Prod_Percent            ;               // 퍼센트
}PD;

// typedef에 의하여, 구조체 _Product 선언과 동시에 이름 변경


// list 로 관리 할 데이터와, 계산되어 집계되는 합 데이터는 분리 함
// 집계 데이터를 관리 할 구조체,
// 분리를 한 이유는, 위의 구조체는 입력(최대 100개)을 해서 배열이 필요하지만, 밑의 구조체는 총합이기때문에 배열이 필요없기 때문이다.

typedef struct _ProductTot{
        int             Prod_ValueTot           ;               // 원가총합
        int             Prod_DisValueTot        ;               // 할인가 총합
        double  Prod_PercentTot         ;               // 총 퍼센트

}PDTOT;
```

```
PDTOT ProdTot;                          // 집계 내역(총합 구조체 이름을 바꿈)


PD ProdList[MAXLINE];          // 등록 되는 상품목록, 상품목록의 최대 갯수는 변동 가능
하므로 분리 한다,  PD구조체에 ProdList 배열을 포함시킴.

////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////

void Tittle_Prn();                                      // 타이틀 출력 함수 선언

int  Menu_Prn();                                        // 메뉴 출력 함수 선언

int Menu_FuncTion(int menunum);        // 메뉴에 기능을 수행하는 함수 선언

////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
// case2의 파일을 저장하기 위해 혹시 모를 에러를 방지하기위해 const로 고정

const char *FileName="Prod_Menu.txt";

//const앞에 붙은 변수는, 초기화 이후의 값 변경을 허용하지 않겠다는 선언.
//그러므로, const (char *FileName) 이므로 포인터가 가르키는 메모리 주솟값의 변경을 허
용치 않겠다.

/*
예시>
const char* Value 와 char* const Value의 차이는?
전자는 포인터 변수에 담긴 내용을 바꾸지 못하게 제한한다 -> 주소를 바꾸지 못한다.
후자는 포인터가 가르키는 곳의 값을 바꿀 수 없게 제한한다.
*/

////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////

int main()
{
        int menunum=0;

        memset(&ProdList,0x00, sizeof(ProdList));                       // 배열 초기화
        memset(&ProdTot,0x00, sizeof(ProdTot));                                 // 원가총
```

```
합, 할인가총합, 총퍼센트가 있는 구조체를 초기화

        // 토탈  전에 메모리 초기화. memset의 기본형태는, memset( 위치 , 초기화값, 사이즈 )



        while(1){

                Tittle_Prn();                           // 타이틀 출력 함수 호출

                menunum = Menu_Prn();       // 메뉴 출력 함수 호출과 동시에, 반환되는 것을 menunum에 저장

                system("cls");                          // 화면 클리어

                Menu_FuncTion(menunum);   // 메뉴기능 함수 호출, 인자로 menunum 던져줌.

                system("cls");                          // 화면 클리어

                if ( menunum == 4 ){          // 메뉴 4번(종료)를 받을 때, 메인 탈출을 통한 프로그램 종료
                        printf("프로그램을 종료합니다.₩n");
                        break;
                }

        }

        return 0;
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////

void Tittle_Prn()               // 타이틀출력을 불러오는 함수 정의
{

printf("--------------------------------------------------------------------------------₩n");
        printf("₩t1260014 김철언₩n₩t1260031 오혁준₩n");
        printf("₩t원가, 할인가 및 할인율 계산₩n");
```

```
printf("--------------------------------------------------------------------------------₩n");
}

////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////

int Menu_Prn()                  // 메뉴출력을 불러오는 함수 정의
{
        int menunum = 0;                // menunum 에 정수를 받아서 이후에 case를 실행할 것임

        while(1)  // 메뉴출력을 반복하기위한 무한반복문
        {
                printf("1. 품목 입력 ₩n₩n");
                printf("2. 영수증 출력₩n₩n");
                printf("3. 파일저장₩n₩n");
                printf("4. 프로그램 종료₩n₩n");
                printf("메뉴 번호를 입력 하세요(1 - 4) : ");

                scanf("%d", &menunum);      // menunum 에 어떤 번호를 받냐에 따라서 main()에서 이후 case가 포함된 함수를 작동시킬 것임.

                getchar(); // 혹시 몰라서, 공백을 하나 잡아먹기 위해 getchar() 설정.
- getchar() 는 '문자'를 하나 받음(화이트 스페이스도 받음)

                if( menunum > 0 && menunum < 5 ) break;  // 1~4만을 입력하게끔 조건을 걸었음.

                printf("₩n!Error!                메뉴 번호 1 - 4 중 선택하여 입력해주세요.                !Error!₩n₩n");

        }
        return menunum;   // 입력받은 번호를 리턴시켜서 swithc~case 문이 있는 함수로 넘길것임.
}

////////////////////////////////////////////////////////////////////////////////////////////////
//////////////////////////////

int Menu_FuncTion(int menunum)                  // 메뉴에 해당하는 switch case를 실행
```

시키기 위한 함수 정의

{

int i, j; // 반복문을 돌리기 위해 i, j 선언. 혹시 모를 충돌을 위해 반복문 변수를 따로 지정함.

PD prd; // 구조체 PD를 prd로 변수 선언 한 것임

FILE *fd; // 파일을 사용하기 위해 파일 구조체를 선언한다.(이것은, 이미 정의되어있는 이용법이다.)

switch(menunum)
{
case 1 :

// 이 경우 구조체 변수 전체를 초기화 하는 것이 좋음. 구조체 초기화를 하지않아 오작동 하는 경우가 많았음.
// 메모리 초기화 memset. 기본 유형 : memset(변수명, memset(위치 , 초기화값, 사이즈)

memset(&prd, 0x00, sizeof(prd));
memset(&ProdList,0x00, sizeof(ProdList));
memset(&ProdTot,0x00, sizeof(ProdTot));

for (i = 0; i < MAXLINE; i++) // 주소값 0부터 MAXLINE(100)까지 반복시킬것임
{
memset(&prd, 0x00, sizeof(prd)); //메모리 초기화. prd구조체들.(즉, PD를 초기화)

printf("상품입력란에 x 를 입력시 품목입력을 종료합니다₩n");
printf("상품의 이름 : ");

scanf("%s", &prd.Prod_Name[0]); // Prod_Name[0][0] 부터 쭉 저장시킬것임.

if(prd.Prod_Name[0] == 'x' || prd.Prod_Name[0]

```
== 'X') break;        // 이름입력에 x 혹은 대문자 X를 받을 경우 저장을 중단함.

                                printf("상품의 원가 : ");
                                scanf("%d", &prd.Prod_Value);  // 상품 가격배열
에 [0] 부터 저장


                                printf("상품의 할인가 : ");
                                scanf("%d", &prd.Prod_DisValue);        // 상품
할인가배열 [0] 부터 저장

                                printf("상품의 갯수 : ");

                                scanf("%d", &prd.Prod_Count);  // 상품 갯수에
[0] 부터 저장

                                // 정수:정수 연산은 결과가 정수로 가정 하므로
(double) 케스팅 없으면 결과는 0 으로 처리 된다
                                // (double) 케스트로 결과가 double 형임을 지정
해야 결과가 < 0 일떄도 값이 보존 된다
                                prd.Prod_Percent = (
(double)(prd.Prod_Value-prd.Prod_DisValue) /(double)prd.Prod_Value  ) * 100 ; // 각 등
록된 상품의 활인율은 즉시 계산 한다

                                ProdList[i] = prd;    // 입력 되고 .계산 된 값을 배
열에따라 차례대로 목록에 등록 한다

                                printf("₩n");

                        }
                        break;


                case 2 :

                        memset(&ProdTot, 0x00, sizeof(ProdTot));

                        system("cls");  // 화면 클리어

                        // 처리할 데이터가 있는지 확인 함
                        if ( ProdList[0].Prod_Name[0] == 0x00 ){
```

```
                                        printf("데이터가 없습니다.
₩n");
                                        getchar(); // 다시 메뉴로 돌
아가는 무한반복을 방지하기 위해 엔터를 한번 더 누르게 해서 메뉴로 돌아감.
                                        break;
                                }

                        printf("%20s₩t%9s₩t%9s₩t%3s₩t%6s₩n", "상품명", "원가
", "할인가", "수량", "할인율");
                        // 표처럼 출력을 위해 맨 위에 상품명, 가격, 수량을 출력시
킴


printf("==========================================================
============================₩n");

                        for ( j = 0; j < MAXLINE; j++)  // 0번부터 MAXLINE(=100)
까지 반복함
                        {


                                if ( ProdList[j].Prod_Name[0] == 0x00 ){
                                        break;
                                }
                                // 입력이 없는 배열지점까지 반복문을 돌려서 출
력시킬 것임.


                                ProdList[j].Prod_Percent =
(((double)ProdList[j].Prod_Value - (double)ProdList[j].Prod_DisValue) /
(double)ProdList[j].Prod_Value) * 100;
                                // 할인율 계산.

                                printf("%20s₩t", ProdList[j].Prod_Name);    //
Prod_Name 출력

                                printf("%9d₩t", ProdList[j].Prod_Value); //
Prod_Valude 출력

                                printf("%9d₩t", ProdList[j].Prod_DisValue);
// Prod_DisValue 출력
```

```
                printf("%3d₩t", ProdList[j].Prod_Count); // Prod_Count 출력

                printf("%6f%%₩n", ProdList[j].Prod_Percent); // Prod_Percent 출력

            }

printf("=============================================================₩n");

            for( j = 0; j < MAXLINE; j++){
                if ( ProdList[j].Prod_Name[0] == 0x00 )
                    break;

                ProdTot.Prod_ValueTot += ProdList[j].Prod_Value *  ProdList[j].Prod_Count;

                ProdTot.Prod_DisValueTot += ProdList[j].Prod_DisValue * ProdList[j].Prod_Count;

            }// 0번 배열부터 Prod_Name이 0인자리(입력이 없는 자리 즉, 입력한곳~무입력지점) 총합들을 각 변수에 넣음


            ProdTot.Prod_PercentTot = ( (double)(ProdTot.Prod_ValueTot - ProdTot.Prod_DisValueTot) /(double) ProdTot.Prod_ValueTot) * 100;
            // 할인율을 구하는 식 = ((원가 - 할인가) / 원가) * 100


            printf("총원가 : %9d 원₩t 총할인가 : %9d₩t 총할인율 : %9.2f%%₩n", ProdTot.Prod_ValueTot, ProdTot.Prod_DisValueTot, ProdTot.Prod_PercentTot);  // 연산한 총합 출력
            // %9d 의 경우 필드폭을 9바이트 주겠다는 서식문자. %9.2f 의 경우, 9바이트 필드폭에 소숫점 2자리까지 표현. f는 더블보다 좀 작은 표현인 float.  %%는 %를 출력하기 위함.
```

getchar(); // 다시 메뉴로 돌아가는 무한반복을 방지하기 위해 엔터를 한번 더 누르게 해서 메뉴로 돌아감.

break;

//
//
//하이라이트

case 3 : // 파일 출력의 case문.

if (ProdList[0].Prod_Name[0] == 0x00) // prodNmae을 0 일때, break로 멈추는 조건문. 데이터가 아예 없는 경우를 위함.
{
printf("테이터가 없습니다. ₩n");
getchar(); // 다시 메뉴로 돌아가는 무한반복을 방지하기 위해 엔터를 한번 더 누르게 해서 메뉴로 돌아감.
break;
}

fd = fopen(FileName,"w"); // 파일을 overwrite 모드로 open 함, 파일이 존재 하면 기존 파일을 0 size 로 민들고 open,

// 기존 데이터는 삭제되는 효과임

if (fd!=NULL) { //file open 이 성공 했는 지를 검사

// fprintf printf 와 같은 사용법을 제공한다
// 단지 결과의 도착 지가 가 text 형의 파일로 지정 된 것 뿐이다

// *****************이제부터, printf 앞에 f가 붙게 되서 fprintf 는, 파일로 출력을 할 사항들이다*************************

// 보면 알겠지만, fprint라는 점만 빼고, case2 에서 본 내용들이다. 그것들을 저장하기 위해서이다.

```
fprintf(fd,"%20s₩t%9s₩t%9s₩t%3s₩t%6s₩n", "상품명", "원가", "할인가", "수량", "할인율");
                                                    // 표처럼 출력을 위해 맨 위에 상품명, 가격, 수량을 출력시킴

fprintf(fd,"=============================================================================₩n");
                                                    for(j=0 ; j < MAXLINE;j++) {
                                                        if ( ProdList[j].Prod_Name[0] == 0x00 ){
                                                            // prodNmae을 0 일때, break로 멈추는 조건문. 데이터가 아예 없는 경우를 위함.{
                                                                break;
                                                        }
                                                        fprintf(fd,"%20s₩t", ProdList[j].Prod_Name);    // Prod_Name 출력
                                                        fprintf(fd,"%9d₩t", ProdList[j].Prod_Value); // Prod_Valude 출력
                                                        fprintf(fd,"%9d₩t", ProdList[j].Prod_DisValue);      // Prod_DisValue 출력
                                                        fprintf(fd,"%3d₩t", ProdList[j].Prod_Count); // Prod_Count 출력
                                                        fprintf(fd,"%6f%%₩n", ProdList[j].Prod_Percent);        // Prod_Percent 출력
                                                    }
```

```
fprintf(fd,"==========================================================
==============================₩n");
                                                fprintf(fd,"총원가 : %9d 원₩t 총할인가
: %9d₩t 총할인율 : %9.2f%%₩n", ProdTot.Prod_ValueTot, ProdTot.Prod_DisValueTot,
ProdTot.Prod_PercentTot);  // 연산한 총합 출력

                                                fclose(fd);
                                                // 파일을 열었으면 반드시 닫아야 한다
, close 해야 내부 buffer 에 있던 데이터가 실제로 파일로 모두  보내진다

                                                printf("파일명:%s 로 저장 되었습니다
₩n",FileName);
                                                // 파일이 저장되었음을 알리는 텍스트.

                                                getchar(); // 다시 메뉴로 돌아가는 무
한반복을 방지하기 위해 엔터를 한번 더 누르게 해서 메뉴로 돌아감.

                                        }

                                        break;

////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////

                case 4 :
                        printf("프로그램을 종료합니다.₩n");
                        break;

        }

        return menunum;
}
```