```
/**//* 27.940.642-ROSSI,SebastianPablo-(05-1965) *//**/

/**
                 Ejercicio 3:   Aprobado
                 Observaciones: -
**/

#include "funciones.h"
#include <string.h>
#include <stdlib.h>
#define minimo( X, Y ) ( ( X )  <= ( Y ) ? ( X ) : ( Y ) )


/**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**/
/**//* FUNCIONES A DESARROLLAR                                                 *//**/
/**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**/
/**//**//**//* PUNTO 3                                                  *//**//**//**/
/**/


int  procesarPedidos_MIO(FILE *fpPedi, FILE *fpPend, FILE *fpMayo, FILE *fpMeno,
                         FILE *fpPant)
{
    tPedi a, b;
    tPila pGuarda, pOrden;
    FILE *fpAux;
        double total;
        size_t fin;

        crearPila_MIO(&pGuarda);
    crearPila_MIO(&pOrden);


    // en caso de estar vacio el archivo, finaliza
    fin = !fread(&a, sizeof(tPedi),1, fpPedi);
    b=a;

    while(!fin)
    {
        // la primera vez ingresa por aca
        // si el nombre del cliente es el mismo que hay en pila, ingresa aca
        if(!strcmp(a.codClie, b.codClie))
        {
            if(!a.precUnit)
            {
                // si el precio unitario es 0.0, no ingresara mas datos en la pila
                // y pasara a guardar, ordenando los que si estan en pila, pasando
                // de una  a la otra
                while(sacarDePila_MIO(&pGuarda,&b, sizeof(tPedi)))
                {
                    if(!ponerEnPila_MIO(&pOrden,&b,sizeof(tPedi)))
                    {
                        fprintf(stderr,"ERROR - inesperado - pila llena\n");
                        return 0;
                    }
                }

                // una vez ordenados, los guarda en pendie
                while(sacarDePila_MIO(&pOrden,&b, sizeof(tPedi)))
                    fwrite(&b,sizeof(tPedi),1, fpPend);

                // si en archivo pedidos hay mas pedidos del mismo clientes, tambien
                // los guarda en pendie sin ingresarlos a ninguna pila
                while(!strcmp(a.codClie, b.codClie) && !feof(fpPedi))
                {
                    fwrite(&a,sizeof(tPedi),1, fpPend);
                    fread(&a, sizeof(tPedi),1, fpPedi);
                }
            }
```

```
    // si el precio no es 0.0, guarda en pila
    if(!feof(fpPedi))
    {
        if(!ponerEnPila_MIO(&pGuarda,&a,sizeof(tPedi)))
        {
            fprintf(stderr,"ERROR - inesperado - pila llena\n");
            return 0;
        }
    }
}

// si el nombre del cliente no es igua al ultimo que hay en pila, ingresa aca
else
{
    // saca los datos de pila y los pone en otra pila para acomodarlos
    // calcula el total
    while(sacarDePila_MIO(&pGuarda,&b,sizeof(tPedi)))
    {
        total+=b.precUnit*b.cant;
        if(!ponerEnPila_MIO(&pOrden, &b, sizeof(tPedi)))
        {
            fprintf(stderr,"ERROR - inesperado - pila llena\n");
            return 0;
        }
    }

    // comprueba en que archivo se guardaran los datos
    if(total>MAYOR_QUE)
        fpAux=fpMayo;
    else
        fpAux=fpMeno;


    // titulo
    fprintf(fpAux, "Pedido del cliente %s por %.2f\n\n"
                        "CodProduc. Cant.  PrecUnit  SubTotal\n",
                        b.codClie, total);


    // saca de pila donde se pasaron para acomodar y lo graba en archivo
    while(sacarDePila_MIO(&pOrden,&b,sizeof(tPedi)))
    {
        fprintf(fpAux, "%-*s%*d %*.2f %*.2f\n", sizeof("CodProduc."),b.codProd,
                                                sizeof("Cant"),b.cant,
                                                sizeof("PrecUnit"),b.precUnit,
                                                sizeof("SubTotal"),b.precUnit*b.cant);
    }

    // graba pie
    fprintf(fpAux,"\n\n\n");


    total = 0;


    // pone en pila el dato que saco los anteriores
    if(!ponerEnPila_MIO(&pGuarda,&a,sizeof(tPedi)))
    {
        fprintf(stderr,"ERROR - inesperado - pila llena\n");
        return 0;
    }
}

b=a;

// si no queda nada por hacer, se activa el fin
if(!fread(&a, sizeof(tPedi),1, fpPedi) && feof(fpPedi) && pilaVacia_MIO(&pGuarda))
    fin=1;
```

```
    }
    return 1;
}


/**//**//**//*  DESARROLLE LAS PRIMITIVAS DE PILA CON ASIGNACIÓN  *//**//**//**/
/**//**//**//*  DINÁMICA DE MEMORIA COMO TIENE ESTÁ EN LA BIBLIO- *//**//**//**/
/**//**//**//*  GRAFÍA PROVISTA EN EL CURSO EN LA  [SEMANA 7]     *//**//**//**/


void crearPila_MIO(tPila *p)
{
    *p = NULL;
}


int  pilaLlena_MIO(const tPila *p, unsigned cantBytes)
{
    tNodo *aux = (tNodo *)malloc(sizeof(tNodo));
    void  *info = malloc(cantBytes);

    free(aux);
    free(info);
    return aux == NULL || info == NULL;
}


int  ponerEnPila_MIO(tPila *p, const void *d, unsigned cantBytes)
{
    tNodo *nue;
    if((nue = (tNodo *)malloc(sizeof(tNodo))) == NULL || (nue->info = malloc(cantBytes)) == NULL )
    {
        free(nue);
        return 0;
    }
    memcpy(nue->info,d,cantBytes);
    nue->tamInfo = cantBytes;
    nue->sig = *p;
    *p = nue;
    return 1;
}


int  verTope_MIO(const tPila *p, void *d, unsigned cantBytes)
{
    if(*p == NULL)
        return 0;
    memcpy(d,(*p)->info, minimo(cantBytes, (*p)->tamInfo));
    return 1;
}


int  pilaVacia_MIO(const tPila *p)
{
    return *p == NULL;
}


int  sacarDePila_MIO(tPila *p, void *d, unsigned cantBytes)
{
    tNodo *aux = *p;

    if(aux == NULL)
        return 0;
    *p = aux->sig;
    memcpy(d, aux->info, minimo(cantBytes, aux->tamInfo));
    free(aux->info);
    free(aux);
    return 1;
```

```
}


void vaciarPila_MIO(tPila *p)
{
    while(*p)
    {
        tNodo *aux = *p;
        *p = aux->info;
        free(aux->info);
        free(aux);
    }
}


/**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**//**/
```