HACER UN CODIGO QUE LE PIDA AL USUARIO QUE PRODUCTOS QUIERE Y QUE CANTIDAD, LA CANTIDAD SE TIENE QUE IR DESCONTANDO TIENE QUE IR AGUARDANDO LOS DATOS, TIENE QUE SUMARLOS Y ARROJARTE UN RESULTADO Y APARTE TIENE QUE AGUARDAR TUS VENTAS

#include <stdio.h>

int main(){

char OPCION,PRODUCTO, MANZANA, MANDARINA, MANGO;

int CANTIDAD;

printf("HOLA QUE PRODUCTOS DESEAS ADQUIRIR???: \n 1-MANZANA \n 2-MANGO \n 3-MANDARINA \n");

scanf("%c",&PRODUCTO);

if(OPCION== 1)

{

PRODUCTO=1;

}

if(OPCION== 2)

{

PRODUCTO=2;

}

if(OPCION==3)

{

PRODUCTO= 3;

}

printf("el rpoducto ")

return 0;

}

Apuntadores

Tipo de dato, \*, nombre

Int\*a;

#include <stdio.h>

int main()

{

int i;

int \*p;

p=&i;

\*p=50;

printf("%i",i);

return 0;

}

int main()

{

int i;

int \*p;

p=&i;

\*p=50;

printf("%i,%i",p,&i);

return 0;

}

FUNCIONES, VARIABLE LOCAL

#include <stdio.h>

void funcion (int x)

{

x=15;

}

int main()

{

int x;

x=10;

funcion (x);

printf("%i",x);

return 0;

}

CON APUNTADORES

#include <stdio.h>

void funcion (int \*x)

{

\*x=15;

}

int main()

{

int x;

x=10;

funcion (&x);

printf("%i",x);

return 0;

}

CHAR

#include <stdio.h>

void funcion (char \*x)

{

\*x= 'A';

}

int main()

{

char x;

funcion (&x);

printf("%c",x);

return 0;

}

FLOAT

#include <stdio.h>

void funcion (float \*x)

{

\*x= 1.0;

}

int main()

{

float x;

funcion (&x);

printf("%f",x);

return 0;

}

Venta

Char []

Int

0001

0002

003

Coca

Limon

Chetos

Char []

15.03

Float

#include <stdio.h>

char codigo [10][4];

char nombre [10][20];

float precio [10] ;

void funcion () {

printf("dame el nombre del producto:\n");

scanf ("%s",&nombre);

printf("dame el codigo:\n");

scanf ("%s",&codigo);

printf("dame el precio:\n");

scanf ("%f",&precio);

}

void funcionmostrar()

{

printf("el codigo del producto es:%s \n", codigo);

printf("el nombre del producto es:%s \n ", nombre);

printf("el codigo del producto es:%s \n", codigo);

}

int main(){

funcionAlta ();

return 01;

}

scanf("%i",&f.mes);

Vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

#include <stdio.h>

int arr[10];

int ind=0;

int buscar(int valor);

void eliminar();

void mostrar();

void alta();

int main()

{

int opc,salir=0;

do{

printf("Que deceas hacer?");

printf("1)Dar de alta\n");

printf("1)Dar de baja\n");

printf("1)Dar de mostrar\n");

scanf("%i",&opc);

switch(opc){

case 1: alta();

break;

case 2: eliminar();

break;

case 3: mostrar();

break;

}

printf("Deseas salir?(si=1/no=0\n");

scanf("%i",&salir);

}while(salir==0);

return 0;

}

void alta(){

printf("Dame un elemento \n");

scanf("%i",&arr[ind]);

ind++;

}

void mostrar(){

int i;

printf("Los elementos almacenados son: \n");

for(i=0;i<ind;i++){

printf("El elemento %i es %i \n",i+1,arr[i]);

}

}

void eliminar(){

int valor,indice;

printf("Que valor deseas eliminar\n");

scanf("%i",&valor);

indice=buscar(valor);

printf("La posicion del valor es: %i\n",indice);

}

int buscar(int valor){

for(int i = 0; i<ind;i++){

if(arr[i]==valor){

return i;

}

}

return -1;

}

#include <stdio.h> /\* print, scanf \*/

#include <string.h>

#include <stdlib.h> /\* system \*/

#define MAX 80

#define VALOR\_CENTINELA -1

/\* Estructuras \*/

struct producto {

int codigo;

char nombre[MAX];

float costo;

float precio;

int cantidad;

};

typedef struct producto Producto;

/\* Opciones del Menú \*/

void menuPrincipal();

void menuInsertar();

void menuBuscar();

void menuEliminar();

void menuMostrar();

void menuModificar();

void menuVentas();

/\* Funciones para manejar el archivo directamente \*/

Producto obtenerProductos(int \*n); / Obtiene un vector dinámico de productos \*/

char existeProducto(int codigoProducto, Producto producto); / Busca si existe el producto en el archivo de productos \*/

char insertarProducto(Producto producto); /\* Inserta el producto al final del archivo \*/

char eliminarProducto(int codigoProducto); /\* Eliminar el producto de código codigoProducto del archivo \*/

char modificarProducto(Producto producto); /\* Modifica el producto en el archivo \*/

/\* Función de lectura de cadenas \*/

int leecad(char \*cad, int n);

char linea[MAX];

int main()

{

menuPrincipal();

return 0;

}

void menuPrincipal()

{

char repite = 1;

int opcion = -1;

/\* Cuando el usuario ingresa texto en lugar de ingresar una opción. El programa no modifica

el valor de opcion. En ese caso, no se debe de ingresar a ninguno de los case, por eso se está

inicializando la variable opcion con un valor que no permita ejecutar ningún case. Simplemente,

volver a interar y pedir nuevamente la opción. \*/

do {

printf("\n\t\t\t\tMENU PRINCIPAL\n");

printf("\n\t\t[1]. Insertar nuevo producto\n");

printf("\t\t[2]. Mostrar listado de productos\n");

printf("\t\t[3]. Eliminar un producto\n");

printf("\t\t[4]. Buscar producto por clave\n");

printf("\t\t[5]. Modificar un producto\n");

printf("\t\t[6]. Ventas\n");

printf("\t\t[7]. Salir\n");

printf("\n\t\tIngrese su opción: [ ]\b\b");

/\* Lectura segura de un entero \*/

leecad(linea, MAX);

sscanf(linea, "%d", &opcion);

switch (opcion) {

case 1:

menuInsertar();

break;

case 2:

menuMostrar();

break;

case 3:

menuEliminar();

break;

case 4:

menuBuscar();

break;

case 5:

menuModificar();

break;

case 6:

menuVentas();

break;

case 7:

repite = 0;

break;

}

} while (repite);

}

void menuInsertar()

{

Producto producto;

int codigoProducto = 0;

char repite = 1;

char respuesta[MAX];

do {

printf("\n\t\t\t==> INSERTAR PRODUCTO <==\n");

/\* Se pide el código del producto a insertar \*/

printf("\n\tCódigo de producto: ");

leecad(linea, MAX);

sscanf(linea, "%d", &codigoProducto);

/\* Se verifica que el producto no haya sido almacenado anteriormente \*/

if (!existeProducto(codigoProducto, &producto)) {

producto.codigo = codigoProducto;

/\* Se piden los demás datos del producto a insertar \*/

printf("\tNombre del producto: ");

leecad(producto.nombre, MAX);

printf("\tCosto del producto ($): ");

leecad(linea, MAX);

sscanf(linea, "%f", &producto.costo);

printf("\tPrecio del producto ($): ");

leecad(linea, MAX);

sscanf(linea, "%f", &producto.precio);

printf("\tCantidad: ");

leecad(linea, MAX);

sscanf(linea, "%d", &producto.cantidad);

/\* Se inserta el producto en el archivo \*/

if (insertarProducto(producto)) {

printf("\n\tEl producto fue insertado correctamente\n");

} else {

printf("\n\tOcurrió un error al intentar insertar el producto\n");

printf("\tInténtelo mas tarde\n");

}

} else {

/\* El producto ya existe, no puede ser insertado. \*/

printf("\n\tEl producto de código %d ya existe.\n", codigoProducto);

printf("\tNo puede ingresar dos productos distintos con el mismo código.\n");

}

printf("\n\tDesea seguir ingresando productos? [S/N]: ");

leecad(respuesta, MAX);

if (!(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0)) {

repite = 0;

}

} while (repite);

}

void menuBuscar()

{

Producto producto;

int codigoProducto;

char repite = 1;

char respuesta[MAX];

do {

printf("\n\t\t\t==> BUSCAR PRODUCTO POR CÓDIGO <==\n");

/\* Se pide el código del producto a buscar \*/

printf("\n\tCódigo de producto: ");

leecad(linea, MAX);

sscanf(linea, "%d", &codigoProducto);

/\* Se verifica que el producto a buscar, exista \*/

if (existeProducto(codigoProducto, &producto)) {

/\* Se muestran los datos del producto \*/

printf("\n\tCódigo del producto: %d\n", producto.codigo);

printf("\tNombre del producto: %s\n", producto.nombre);

printf("\tCosto del producto: %.1f $\n", producto.costo);

printf("\tPrecio del producto: %.1f $\n", producto.precio);

printf("\tCantidad: %d\n", producto.cantidad);

} else {

/\* El producto no existe \*/

printf("\n\tEl producto de código %d no existe.\n", codigoProducto);

}

printf("\n\tDesea seguir buscando algún producto? [S/N]: ");

leecad(respuesta, MAX);

if (!(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0)) {

repite = 0;

}

} while (repite);

}

void menuEliminar()

{

Producto producto;

int codigoProducto;

char repite = 1;

char respuesta[MAX];

do {

printf("\n\t\t\t==> ELIMINAR PRODUCTO POR CÓDIGO <==\n");

/\* Se pide el código del producto a eliminar \*/

printf("\n\tCódigo de producto: ");

leecad(linea, MAX);

sscanf(linea, "%d", &codigoProducto);

/\* Se verifica que el producto a buscar, exista \*/

if (existeProducto(codigoProducto, &producto)) {

/\* Se muestran los datos del producto \*/

printf("\n\tCódigo del producto: %d\n", producto.codigo);

printf("\tNombre del producto: %s\n", producto.nombre);

printf("\tCosto del producto: %.1f $\n", producto.costo);

printf("\tPrecio del producto: %.1f $\n", producto.precio);

printf("\tCantidad: %d\n", producto.cantidad);

printf("\n\tSeguro que desea eliminar el producto? [S/N]: ");

leecad(respuesta, MAX);

if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {

if (eliminarProducto(codigoProducto)) {

printf("\n\tProducto eliminado satisfactoriamente.\n");

} else {

printf("\n\tEl producto no pudo ser eliminado\n");

}

}

} else {

/\* El producto no existe \*/

printf("\n\tEl producto de código %d no existe.\n", codigoProducto);

}

printf("\n\tDesea eliminar otro producto? [S/N]: ");

leecad(respuesta, MAX);

if (!(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0)) {

repite = 0;

}

} while (repite);

}

void menuMostrar()

{

Producto \*productos;

int numeroProductos;

int i;

float costoTotal;

float precioTotal;

int cantidadTotal;

char respuesta[MAX];

productos = obtenerProductos(&numeroProductos);

if (numeroProductos == 0) {

printf("\n\tEl archivo está vacío!!\n");

getchar();

} else {

printf("\n\t\t ==> LISTADO DE PRODUCTOS REGISTRADOS <==\n");

printf(" ------------------------------------------------------------------------------\n");

printf("%8s\t%-20s%15s%15s%10s\n", "CODIGO", "NOMBRE", "COSTO $", "PRECIO $", "CANTIDAD");

printf(" ------------------------------------------------------------------------------\n");

/\* Se recorre el vector dinámico de productos \*/

costoTotal = 0;

precioTotal = 0;

cantidadTotal = 0;

for (i = 0; i < numeroProductos; i++) {

if (productos[i].codigo != VALOR\_CENTINELA) {

printf("%7d \t%-20.20s%15.1f%15.1f%8d\n", productos[i].codigo, productos[i].nombre, productos[i].costo, productos[i].precio, productos[i].cantidad);

costoTotal += productos[i].costo;

precioTotal += productos[i].precio;

cantidadTotal += productos[i].cantidad;

}

}

printf(" ------------------------------------------------------------------------------\n");

printf("\t\t\t TOTAL: %15.1f%15.1f%8d\n", costoTotal, precioTotal, cantidadTotal);

getchar();

}

}

void menuModificar()

{

Producto producto;

int codigoProducto;

char repite = 1;

char respuesta[MAX];

do {

printf("\n\t\t\t==> MODIFICAR PRODUCTO POR CÓDIGO <==\n");

/\* Se pide el código del producto a modificar \*/

printf("\n\tCódigo de producto: ");

leecad(linea, MAX);

sscanf(linea, "%d", &codigoProducto);

/\* Se verifica que el producto a buscar exista \*/

if (existeProducto(codigoProducto, &producto)) {

/\* Se muestran los datos del producto \*/

printf("\n\tNombre del producto: %s\n", producto.nombre);

printf("\tCosto del producto ($): %.1f\n", producto.costo);

printf("\tPrecio del producto ($): %.1f\n", producto.precio);

printf("\tCantidad: %d\n", producto.cantidad);

printf("\n\tElija los datos a modificar\n");

/\* Modificación del nombre del producto \*/

printf("\n\tNombre del producto actual: %s\n", producto.nombre);

printf("\tDesea modificar el nombre del producto? [S/N]: ");

leecad(respuesta, MAX);

if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {

printf("\tNuevo nombre del producto: ");

leecad(producto.nombre, MAX);

}

/\* Modificación del costo del producto \*/

printf("\n\tCosto del producto actual: %.1f\n", producto.costo);

printf("\tDesea modificar el costo del producto? [S/N]: ");

leecad(respuesta, MAX);

if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {

printf("\tNuevo costo del producto: ");

leecad(linea, MAX);

sscanf(linea, "%f", &producto.costo);

}

/\* Modificación del precio del producto \*/

printf("\n\tPrecio del producto actual: %.1f\n", producto.precio);

printf("\tDesea modificar el precio del producto? [S/N]: ");

leecad(respuesta, MAX);

if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {

printf("\tNuevo precio del producto: ");

leecad(linea, MAX);

sscanf(linea, "%f", &producto.precio);

}

/\* Modificación de la cantidad del producto \*/

printf("\n\tCantidad del producto actual: %d\n", producto.cantidad);

printf("\tDesea modificar la cantidad del producto? [S/N]: ");

leecad(respuesta, MAX);

if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {

printf("\tNueva cantidad del producto: ");

leecad(linea, MAX);

sscanf(linea, "%d", &producto.cantidad);

}

printf("\n\tEstá seguro que desea modificar los datos del producto? [S/N]: ");

leecad(respuesta, MAX);

if (strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0) {

/\* Se modifica el producto en el archivo \*/

if (modificarProducto(producto)) {

printf("\n\tEl producto fue modificado correctamente\n");

} else {

printf("\n\tOcurrió un error al intentar modificar el producto\n");

printf("\tInténtelo mas tarde\n");

}

}

} else {

/\* El producto no existe \*/

printf("\n\tEl producto de código %d no existe.\n", codigoProducto);

}

printf("\n\tDesea modificar algún otro producto? [S/N]: ");

leecad(respuesta, MAX);

if (!(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0)) {

repite = 0;

}

} while (repite);

}

void menuVentas()

{

char respuesta[MAX];

printf("\n\t\t==> no se han registrado ventas <==\n");

getchar();

}

Producto \*obtenerProductos(int \*n)

{

FILE \*archivo;

Producto producto;

Producto productos; / Vector dinámico de productos \*/

int i;

/\* Abre el archivo en modo lectura \*/

archivo = fopen("productos.dat", "rb");

if (archivo == NULL) { /\* Si no se pudo abrir el archivo, el valor de archivo es NULL \*/

n = 0; / No se pudo abrir. Se considera n \*/

productos = NULL;

} else {

fseek(archivo, 0, SEEK\_END); /\* Posiciona el cursor al final del archivo \*/

n = ftell(archivo) / sizeof(Producto); / # de productos almacenados en el archivo. (# de registros) \*/

productos = (Producto )malloc((\*n) \* sizeof(Producto)); / Se reserva memoria para todos los productos almacenados en el archivo \*/

/\* Se recorre el archivo secuencialmente \*/

fseek(archivo, 0, SEEK\_SET); /\* Posiciona el cursor al principio del archivo \*/

fread(&producto, sizeof(producto), 1, archivo);

i = 0;

while (!feof(archivo)) {

productos[i++] = producto;

fread(&producto, sizeof(producto), 1, archivo);

}

/\* Cierra el archivo \*/

fclose(archivo);

}

return productos;

}

char existeProducto(int codigoProducto, Producto \*producto)

{

FILE \*archivo;

char existe;

/\* Abre el archivo en modo lectura \*/

archivo = fopen("productos.dat", "rb");

if (archivo == NULL) { /\* Si no se pudo abrir el archivo, el valor de archivo es NULL \*/

existe = 0;

} else {

existe = 0;

/\* Se busca el producto cuyo código coincida con codigoProducto \*/

fread(&(\*producto), sizeof(\*producto), 1, archivo);

while (!feof(archivo)) {

if ((\*producto).codigo == codigoProducto) {

existe = 1;

break;

}

fread(&(\*producto), sizeof(\*producto), 1, archivo);

}

/\* Cierra el archivo \*/

fclose(archivo);

}

return existe;

}

char insertarProducto(Producto producto)

{

FILE \*archivo;

char insercion;

/\* Abre el archivo para agregar datos al final \*/

archivo = fopen("productos.dat", "ab"); /\* Añade datos al final. Si el archivo no existe, es creado \*/

if (archivo == NULL) { /\* Si no se pudo abrir el archivo, el valor de archivo es NULL \*/

insercion = 0;

} else {

fwrite(&producto, sizeof(producto), 1, archivo);

insercion = 1;

/\* Cierra el archivo \*/

fclose(archivo);

}

return insercion;

}

/\* ELiminación lógica de un registro \*/

char eliminarProducto(int codigoProducto)

{

FILE \*archivo;

FILE \*auxiliar;

Producto producto;

char elimina;

/\* Abre el archivo para leer \*/

archivo = fopen("productos.dat", "r+b"); /\* Modo lectura/escritura. Si el archivo no existe, es creado \*/

if (archivo == NULL) { /\* Si no se pudo abrir el archivo, el valor de archivo es NULL \*/

elimina = 0;

} else {

/\* Se busca el registro que se quiere borrar. Cuando se encuentra, se sitúa en esa posición mediante la

función fseek y luego se modifica el campo clave de ese registro mediante algún valor centinela, eso se logra

con fwrite. Hasta allí se ha logrado una eliminación LÓGICA. Porque el registro sigue ocupando espacio en el archivo físico \*/

elimina = 0;

fread(&producto, sizeof(producto), 1, archivo);

while (!feof(archivo)) {

if (producto.codigo == codigoProducto) {

fseek(archivo, ftell(archivo) - sizeof(producto), SEEK\_SET);

producto.codigo = VALOR\_CENTINELA;

fwrite(&producto, sizeof(producto), 1, archivo);

elimina = 1;

break;

}

fread(&producto, sizeof(producto), 1, archivo);

}

/\* Cierra el archivo \*/

fclose(archivo);

}

return elimina;

}

char modificarProducto(Producto producto)

{

FILE \*archivo;

char modifica;

Producto producto2;

/\* Abre el archivo para lectura/escritura \*/

archivo = fopen("productos.dat", "rb+");

if (archivo == NULL) { /\* Si no se pudo abrir el archivo, el valor de archivo es NULL \*/

modifica = 0;

} else {

modifica = 0;

fread(&producto2, sizeof(producto2), 1, archivo);

while (!feof(archivo)) {

if (producto2.codigo == producto.codigo) {

fseek(archivo, ftell(archivo) - sizeof(producto), SEEK\_SET);

fwrite(&producto, sizeof(producto), 1, archivo);

modifica = 1;

break;

}

fread(&producto2, sizeof(producto2), 1, archivo);

}

fclose(archivo);

}

/\* Cierra el archivo \*/

return modifica;

}

int leecad(char \*cad, int n)

{

int i, c;

/\* Hay que verificar si el buffer está limpio o si hay un '\n'

dejado por scanf y, en ese caso, limpiarlo:

\*/

/\* 1 COMPROBACIÓN DE DATOS INICIALES EN EL BUFFER \*/

/\* Empezamos leyendo el primer caracter que haya en la entrada. Si es

EOF, significa que no hay nada por leer, así que cerramos la cadena,

dejándola "vacía" y salimos de la función retornando un valor de 0

o falso, para indicar que hubo un error \*/

c = getchar();

if (c == EOF) {

cad[0] = '\0';

return 0;

}

/\* Si el valor leído es '\n', significa que había un caracter de nueva línea

dejado por un scanf o función similar. Simplemente inicializamos i a 0,

para indicar que los siguientes caracteres que leamos iremos asignando a

partir del primer caracter de la cadena. \*/

if (c == '\n') {

i = 0;

} else {

/\* Si no había un '\n', significa que el caracter que leímos es el primer

caracter de la cadena introducida. En este caso, lo guardamos en la

posición 0 de cad, e inicializamos i a 1, porque en este caso, como ya

tenemos el primer caracter de la cadena, continuaremos agregando

caracteres a partir del segundo.

\*/

cad[0] = c;

i = 1;

}

/\* 2. LECTURA DE LA CADENA \*/

/\* El for empieza con un ; porque estamos omitiendo la inicialización del contador,

ya que fue inicializado en el punto anterior.

Este código lee un caracter a la vez,lo agrega a cad, y se repite hasta que

se encuentre un fin de línea, fin de archivo, o haya leído la cantidad máxima

de caracteres que se le indicó. Luego, cierra la cadena agregando un '\0'

al final. Todo esto es muy similar a la forma en que los compiladores suelen

implementar la función fgets, sólo que en lugar de getchar usan getc o fgetc

\*/

for (; i < n - 1 && (c = getchar()) != EOF && c != '\n'; i++) {

cad[i] = c;

}

cad[i] = '\0';

/\*3. LIMPIEZA DEL BUFFER \*/

/\* Finalmente limpiamos el buffer si es necesario \*/

if (c != '\n' && c != EOF) /\* es un caracter \*/

while ((c = getchar()) != '\n' && c != EOF);

/\* La variable c contiene el último caracter leído. Recordemos que había 3 formas

de salir del for: que hayamos encontrando un '\n', un EOF, o que hayamos llegado

al máximo de caracteres que debemos leer. Si se da cualquiera de los dos

primeros casos, significa que leímos todo lo que había en el buffer, por lo que

no hay nada que limpiar. En el tercer caso, el usuario escribió más caracteres

de los debidos, que aún están en el buffer, por lo que hay que quitarlos, para

lo cual usamos el método que vimos poco más arriba

\*/

return 1;