UNIVERSIDAD POLITÉCNICA DE SAN LUIS POTOSÍ

28-11-2019

Manual del Programador

Almacén de Alimentos

Brayan Adrian Galvan Flores 181112
Julieta Rodríguez Espiricueta 180024
UNIVERSIDAD POLITÉCNICA DE SAN LUIS POTOSÍ

Contenido

Manual del Programador

Este manual tiene como finalidad el guiar al lector a través del código del almacén virtual de alimentos, mostrando las diversas funciones que lo conforman y cómo funcionan.

Definición de librerías

Para el buen funcionamiento del programa es necesario definir las librerías que se van a utilizar, siendo las que se muestran en la imagen aquellas que se usan durante la ejecución del programa.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>
```

Declaración de variables

Se declaran diversas variables globales para tener un mayor control de las constantes utilizadas en el programa, tal es el caso de los renglones (REN) y las columnas (COL) para generar la portada. De la misma manera, constante que permite la delimitación del ingreso de caracteres se declara en esta parte.

```
#define REN 36
#define COL 59
#define MAX 80
```

Estructura

Se define una estructura encargada del almacenaje de la información de los productos. Los datos que guarda son el nombre del producto, tomándolo como una variable de tipo ***char***. Por otra parte, las variables de tipo ***int*** son el ID del producto, así como la fecha de entrada y de caducidad del producto, y la cantidad que existe de este.

```
// Estructura
struct _almacen{
    int ID;
    char nombre[MAX];
    int diaE, mesE, anioE;  // Fecha de entrada
    int diaC, mesC, anioC;  // Fecha de caducidad
    int cantidad;
};
typedef struct _almacen Producto;
```

Prototipos de las funciones

En esta sección se declaran las funciones a utilizar durante la corrida del programa.

De igual manera, se declara la matriz que contiene la portada, así como la función *gotoxy*, la cual permite posicionar los elementos del almacén en distintos lugares de la consola.

```
// Opciones portada y login
void portada();
void login();
void colores(int color);

// Opciones de los menús
void menuAdmin();
void menuOperador();
void altaProducto();
void bajaProducto();
void modificarCantidad();
void productosID();
void productosEntrada();
void productosCaducidad();
char productoProxCaduc();
char productoProxTerminar();
void mostrarProductos(); /*---*/

// Funciones para manejar el archivo directamente
Producto *obtenerProductos(int *n);                      // Obtiene un vector dinámico de productos
char existeProducto(int IDProducto, 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 IDProducto);                   // Eliminar el producto de ID IDProducto del archivo
char modificarProducto(Producto producto);               // Modifica el producto en el archivo

// Función para obtener valores enteros de la fecha actual
void fecha(int *dT, int *mT, int *yT);

// Función de lectura de cadenas
int leecad(char *cad, int n);

// Titular y portada del programa
void tituloPrincipal();
```

```
void gotoxy(int x, int y){      // Permite localizar texto en cualquier parte de la consola
    HANDLE hcon;
     hcon = GetStdHandle(STD_OUTPUT_HANDLE);
     COORD dwPos;
     dwPos.X = x;
     dwPos.Y= y;
     SetConsoleCursorPosition(hcon, dwPos);
}

char linea[MAX];
```

Portada:

```
int imagen [REN][COL]={          // Portada
    [illegible]
};
```

Función principal

Esta función actúa como el núcleo del programa, debido a que reúne todas las demás funciones y permite la ejecución del programa.

```
// FUNCIÓN PRINCIPAL: MAIN
int main(){
    portada();  // Portada
    login();    // Ingreso al sistema

    getch();
}
```

Iniciar sesión

El inicio de sesión de un usuario consiste en comparar, de acuerdo al perfil que se haya elegido, la cotraseña ingresada con la contraseña preestablecida.

Los intentos permitidos para poder ingresar al sistema son tres. Una vez excedido ese límite de intentos, el programa se cierra por seguridad.

```
void login(){
    char nombre[20], pass[20];
    int opcion=0, intento=0;
    system("cls");
    do{
        colores(10); printf("\n\t\t.:INICIAR SESI%cN:.\n\n", 224); colores(15);
        printf("\tSeleccione el perfil que desea utilizar:\n\n");
        printf("\t1. Administrador\n");
        printf("\t2. Operador\n\n");
        printf("\tOpci%cn: ", 162);
        leecad(linea, MAX);
        sscanf(linea, "%d", &opcion);

        if(opcion==1){
            do{
                printf("\n\tContrase%ca: ", 164);
                scanf("%s", &pass);
                intento++;
            }while(strcmp(pass, "admin123")!=0 && intento<3);
            if(intento>=3 && strcmp(pass, "admin123")!=0){
                printf("\n\tDemasiados intentos. El programa se cerrar%c por seguridad\n\n\t", 160);
                exit(0);
            }else{
                system("cls");
                menuAdmin();
            }
        }
        else if(opcion==2){
            do{
                printf("\n\tContrase%ca: ", 164);
                scanf("%s", &pass);
                intento++;
            }while(strcmp(pass, "operador123")!=0 && intento<3);
            if(intento>=3 && strcmp(pass, "operador123")!=0){
                printf("\n\tDemasiados intentos. El programa se cerrar%c por seguridad\n\n\t", 160);
                exit(0);
            }else{
                system("cls");
                menuOperador();
            }
        }
        else{
            printf("\n\tIngrese una opci%cn v%clida\n\n\t", 162, 160);
            system("pause");
        }
        system("cls");
    }while(opcion!=1 || opcion!=2);
}
```

Menús

Menú del Administrador

La finalidad de esta función es la de presentar en pantalla las diversas opciones correspondientes al perfil del administrador, por lo que cuenta con un ***switch*** que permite direccionar al usuario hacia la opción elegida.

```
void menuAdmin(){
    char repite = 1;
    int opcion = -1;

    do{
        system("cls");
        tituloPrincipal(); colores(10);
        printf("\n\t\t\t.:ALMAC%cN DE ALIMENTOS:.\n", 144);
        printf("\n\t\tCONTROL DE PRODUCTOS\n"); colores(15);
        printf("\t\t[1]. Alta de productos\n");
        printf("\t\t[2]. Baja de productos\n");
        printf("\t\t[3]. Modificar cantidad de productos\n");
        colores(10); printf("\n\t\tLISTADO DE PRODUCTOS\n"); colores(15);
        printf("\t\t[4]. Productos por ID\n");
        printf("\t\t[5]. Productos por fecha de entrada\n");
        printf("\t\t[6]. Productos por fecha de caducidad\n");
        colores(10); printf("\n\t\tREPORTE DE PRODUCTOS\n"); colores(15);
        printf("\t\t[7]. Productos proximos a caducar ordenados por ID\n");
        printf("\t\t[8]. Productos proximos a terminarse\n\n");
        printf("\t\t[9]. Salir\n");
        printf("\n\t\tIngrese su opci%cn: [ ]\b\b", 162);

        // Lectura segura de un entero
        leecad(linea, MAX);
        sscanf(linea, "%d", &opcion);

        switch(opcion){
            case 1:
                altaProducto();
                break;
            case 2:
                bajaProducto();
                break;
            case 3:
                modificarCantidad();
                break;
            case 4:
                productosID();
                break;
            case 5:
                productosEntrada();
                break;
            case 6:
                productosCaducidad();
                break;
            case 7:
                productoProxCaduc(); system("pause>nul");
                break;
            case 8:
                productoProxTerminar(); system("pause>nul");
                break;
            case 9:
                repite = 0;
                login();
                break;
        }
    }while(repite);
}
```

Menú del Operador

Esta función presenta en pantalla las diversas opciones correspondientes al perfil del operador, por lo que cuenta con un ***switch*** que permite direccionar al usuario hacia la opción elegida.

```
void menuOperador(){
    char repite = 1;
    int opcion = -1;

    do{
        system("cls");
        tituloPrincipal(); colores(10);
        printf("\n\t\t\t.:ALMAC%cN DE ALIMENTOS:.\n", 144);
        printf("\n\t\tCONTROL DE PRODUCTOS\n"); colores(15);
        printf("\t\t[1]. Modificar cantidad de productos\n");
        colores(10); printf("\n\t\tLISTADO DE PRODUCTOS\n"); colores(15);
        printf("\t\t[2]. Productos por ID\n");
        printf("\t\t[3]. Productos por fecha de caducidad\n");
        printf("\t\t[4]. Productos por fecha de entrada\n");
        colores(10); printf("\n\t\tREPORTE DE PRODUCTOS\n"); colores(15);
        printf("\t\t[5]. Productos proximos a caducar ordenados por ID\n");
        printf("\t\t[6]. Productos proximos a terminarse\n\n");
        printf("\t\t[7]. Salir\n");
        printf("\n\t\tIngrese su opcion: [ ]\b\b");

        // Lectura segura de un entero
        leecad(linea, MAX);
        sscanf(linea, "%d", &opcion);

        switch(opcion){
            case 1:
                modificarCantidad();
                break;
            case 2:
                productosID();
                break;
            case 3:
                productosCaducidad();
                break;
            case 4:
                productosEntrada();
                break;
            case 5:
                productoProxCaduc(); system("pause>nul");
                break;
            case 6:
                productoProxTerminar(); system("pause>nul");
                break;
            case 7:
                repite = 0;
                login();
                break;
        }
    }while(repite);
}
```

Control de productos

Alta de un producto

```
void altaProducto(){
    Producto producto;
    int IDProducto = 0;
    char repite = 1;
    char respuesta[MAX];

    do{
        system("cls"); tituloPrincipal(); colores(10);
        printf("\n\t\t\t==> INSERTAR PRODUCTO <==\n"); colores(15);

        // Se pide el ID del producto a insertar
        printf("\n\tID de producto: ");
        leecad(linea, MAX);
        sscanf(linea, "%d", &IDProducto);

        // Se verifica que el producto no haya sido almacenado anteriormente
        if(!existeProducto(IDProducto, &producto)){
            producto.ID = IDProducto;

            // Se piden los demás datos del producto a insertar
            printf("\tNombre del producto: ");
            leecad(producto.nombre, MAX);

            printf("\n\t* La fecha debe de ser ingresada en el siguiente formato: dd/mm/aa *\n");
            printf("\tD%ca de entrada del producto: ", 161);
            leecad(linea, MAX);
            sscanf(linea, "%d", &producto.diaE);

            printf("\tMes de entrada del producto: ");
            leecad(linea, MAX);
            sscanf(linea, "%d", &producto.mesE);

            printf("\tA%co de entrada del producto: ", 164);
            leecad(linea, MAX);
            sscanf(linea, "%d", &producto.anioE);

            printf("\n\tD%ca de caducidad del producto: ", 161);
            leecad(linea, MAX);
            sscanf(linea, "%d", &producto.diaC);

            printf("\tMes de caducidad del producto: ");
            leecad(linea, MAX);
            sscanf(linea, "%d", &producto.mesC);

            printf("\tA%co de caducidad del producto: ", 164);
            leecad(linea, MAX);
            sscanf(linea, "%d", &producto.anioC);

            printf("\n\tCantidad de productos: ");
            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\tOcurri%c un error al intentar insertar el producto\n", 162);
                printf("\tInt%cntelo mas tarde\n", 130);
            }
        }else{
            // El producto ya existe, se modifican los datos de ese producto
            printf("\n\tEl producto de ID %d ya existe\n", IDProducto);

            // Se muestran los datos del producto
            printf("\n\tNombre del producto: %s\n", producto.nombre);
            printf("\tFecha de entrada del producto: %d/%d/%d\n", producto.diaE, producto.mesE, producto.anioE);
            printf("\tFecha de caducidad del producto: %d/%d/%d\n", producto.diaC, producto.mesC, producto.anioC);
            printf("\tCantidad de producto: %d\n", producto.cantidad);

            printf("\n\tDesea modificar los datos del producto seleccionado? [S/N]: ");
            leecad(respuesta, MAX);
            if(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0){
                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("\n\tNuevo nombre del producto: ");
                    leecad(producto.nombre, MAX);
                }

                // Modificación de la fecha de entrada del producto
                printf("\n\tFecha de entrada del producto actual: %d/%d/%d\n", producto.diaE, producto.mesE, producto.anioE);
                printf("\tDesea modificar la fecha de entrada del producto? [S/N]: ");
                leecad(respuesta, MAX);
                if(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0){
                    printf("\n\tNueva fecha de entrada del producto\n");
                    printf("\tD%ca: ", 161); leecad(linea, MAX); sscanf(linea, "%d", &producto.diaE);
                    printf("\tMes: "); leecad(linea, MAX); sscanf(linea, "%d", &producto.mesE);
                    printf("\tA%co: ", 164); leecad(linea, MAX); sscanf(linea, "%d", &producto.anioE);
                }

                // Modificación de la fecha de caducidad del producto
                printf("\n\tFecha de caducidad del producto actual: %d/%d/%d\n", producto.diaC, producto.mesC, producto.anioC);
                printf("\tDesea modificar la fecha de caducidad del producto? [S/N]: ");
                leecad(respuesta, MAX);
                if(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0){
                    printf("\n\tNueva fecha de caducidad del producto\n");
                    printf("\tD%ca: ", 161); leecad(linea, MAX); sscanf(linea, "%d", &producto.diaC);
                    printf("\tMes: "); leecad(linea, MAX); sscanf(linea, "%d", &producto.mesC);
                    printf("\tA%co: ", 164); leecad(linea, MAX); sscanf(linea, "%d", &producto.anioC);
                }
```

```
                printf("\n\t%cEst%c seguro que desea modificar los datos del producto? [S/N]: ", 168, 160);
                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\tOcurri%c un error al intentar insertar el producto\n", 162);
                        printf("\tInt%cntelo mas tarde\n", 130);
                    }
                }
            }
        }
        printf("\n\t%cDesea seguir ingresando productos? [S/N]: ", 168);
        leecad(respuesta, MAX);

        if(!(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0)){
            repite = 0;
        }
    }while(repite);
}
```

Como se puede observar, esta función permite dar de alta un producto en el registro, capturando la información dada por el usuario y guardándola en un archivo de texto con extensión *.txt*.

Baja de un producto

Esta función consta en eliminar lógicamente un producto del registro. Para esto, comprueba la existencia del producto mediante el ID y, si existe, lo elimina. De lo contrario, avisa al usuario de que ese producto no está registrado.

```
void bajaProducto(){
    Producto producto;
    int IDProducto;
    char repite = 1;
    char respuesta[MAX];

    do{
        system("cls"); tituloPrincipal(); colores(10);
        printf("\n\t\t\t==> ELIMINAR PRODUCTO POR ID <==\n"); colores(15);

        // Se pide el ID del producto a eliminar
        printf("\n\tID de producto: ");
        leecad(linea, MAX);
        sscanf(linea, "%d", &IDProducto);

        // Se verifica que el producto a buscar, exista
        if(existeProducto(IDProducto, &producto)){
            // Se muestran los datos del producto
            printf("\n\tID del producto: %d\n", producto.ID);
            printf("\tNombre del producto: %s\n", producto.nombre);
            printf("\tFecha de entrada del producto: %d/%d/%d\n", producto.diaE, producto.mesE, producto.anioE);
            printf("\tFecha de caducidad del producto: %d/%d/%d\n", producto.diaC, producto.mesC, producto.anioC);
            printf("\tCantidad de producto: %d\n", producto.cantidad);

            printf("\n\t%cSeguro que desea eliminar el producto? [S/N]: ", 168);
            leecad(respuesta, MAX);
            if(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0){
                if(eliminarProducto(IDProducto)){
                    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 ID %d no existe\n", IDProducto);
        }

        printf("\n\t%cDesea eliminar otro producto? [S/N]: ", 168);
        leecad(respuesta, MAX);

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

    }while(repite);
}
```

Modificar la cantidad de un producto

El ingreso y egreso de productos se controla a partir de esta función la cual, por medio del ID del producto, permite la visualización de la información del producto en cuestión, así como dejar que el usuario decida la acción a realizar, ya sea ingresando o sacando productos.

Si se elige la primera opción, ingresar cantidad, se suma el monto dado a lo que ya está registrado.

Por otro lado, si se elige la segunda opción, retirar cantidad, el programa procede a validar si hay suficiente cantidad de producto en el almacén. Si la cantidad es menor o igual a la que hay guardada, entonces el retiro procede. De lo contrario, entonces no será posible hacer el retiro.

```
void modificarCantidad(){
    Producto producto;
    int IDProducto=0, opcion, cantProducto=0;
    char repite = 1;
    char respuesta[MAX];

    do{
        system("cls"); tituloPrincipal(); colores(10);
        printf("\n\t\t==> MODIFICAR CANTIDAD DE PRODUCTO <==\n"); colores(15);

        // Se pide el ID del producto a modificar
        printf("\n\tID de producto: ");
        leecad(linea, MAX);
        sscanf(linea, "%d", &IDProducto);

        // Se verifica que el producto no haya sido almacenado anteriormente
        if(!existeProducto(IDProducto, &producto)){
            printf("\n\tEl producto de ID %d no existe.\n", IDProducto);
        }else{
            // El producto si existe, se modifica su cantidad
            printf("\n\tNombre del producto: %s\n", producto.nombre);
            printf("\tFecha de entrada del producto: %d/%d/%d\n", producto.diaE, producto.mesE, producto.anioE);
            printf("\tFecha de caducidad del producto: %d/%d/%d\n", producto.diaC, producto.mesC, producto.anioC);
            printf("\tCantidad de producto: %d\n", producto.cantidad);

            // Se le pregunta al usuario si se trata de un ingreso o retiro de producto
            do{
                printf("\n\t1. Ingresar cantidad\n");
                printf("\t2. Retirar cantidad\n");
                printf("\n\tOpci%cn: [ ]\b\b", 162);
                leecad(linea, MAX); sscanf(linea, "%d", &opcion);

                switch(opcion){
                    case 1:
                        printf("\n\tCantidad de producto ingresada: ");
                        leecad(linea, MAX); sscanf(linea, "%d", &cantProducto);
                        producto.cantidad = producto.cantidad + cantProducto;
                        break;
                    case 2:
                        printf("\n\tCantidad de producto retirada: ");
                        leecad(linea, MAX); sscanf(linea, "%d", &cantProducto);
                        if(cantProducto>producto.cantidad)
                            printf("\n\n\tCantidad de producto insuficiente en almac%cn\n", 130);
                        else
                            producto.cantidad = producto.cantidad - cantProducto;
                        break;
                }
            }while(opcion=!3);

            // Modificación de la cantidad del producto
            printf("\n\tNueva cantidad de producto: %d", producto.cantidad);
            printf("\n\t%cSeguro que desea modificar la cantidad de producto? [S/N]: ", 168);
            leecad(respuesta, MAX);
            if(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0){
                // Se modifica el producto en el archivo
                if(modificarProducto(producto)){
                    printf("\n\tCantidad de producto modificada correctamente\n");
                }else{
                    printf("\n\tOcurri%c un error al intentar modificar el producto\n", 162);
                    printf("\tInt%cntelo mas tarde\n", 130);
                }
            }
        }
        printf("\n\t%cDesea modificar otra cantidad? [S/N]: ", 168);
        leecad(respuesta, MAX);

        if(!(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0)){
            repite = 0;
        }
    }while(repite);
}
```

Ordenamiento de productos

Productos por ID

Esta función utiliza el método de ordenamiento de inserción para poder decretar el orden de los productos por medio de su ID, colocando en primer lugar el producto con el ID más pequeño, hasta llegar al ID más grande.

Para lograr esto, se manda a llamar a la función *obtenerProductos*, la cual extrae del archivo todos los bienes agregados y, con ello, crea un vector dinámico de productos. De esta manera es posible recorrer el arreglo creado mientras se van ordenando crecientemente.

```
void productosID(){
    _almacen aux;
    Producto *productos;
    int i, pos, numeroProductos;

    system("cls"); tituloPrincipal();
    productos = obtenerProductos(&numeroProductos); // Retorna un vector dinámico de productos

    if(numeroProductos == 0){
        printf("\n\tEl archivo est%c vac%co!!\n", 160, 161);
    }else{
        colores(10); printf("\n\t\t    ==> LISTADO DE PRODUCTOS POR ID <==\n");
        printf(" ------------------------------------------------------------------------------\n");
        printf("\tID\tNOMBRE\t\t   FECHA ENTRADA   FECHA CADUCIDAD   CANTIDAD\n");
        printf(" ------------------------------------------------------------------------------\n"); colores(15);

        // Se ordena el vector dinámico de productos por el método de inserción
        for(i=1; i<numeroProductos; i++){
            aux = productos[i];
            pos = i-1;
            while((productos[pos].ID > aux.ID) && (pos>=0)){
                productos[pos+1] = productos[pos];
                pos--;
            }
            productos[pos+1] = aux;
        }

        // Impresión de los productos ordenados por ID
        for(i=0; i<numeroProductos; i++){
            if(productos[i].ID != -1)
                printf("\t%d\t%s", productos[i].ID, productos[i].nombre);
                gotoxy(36, i+12); printf("%d/%d/%d", productos[i].diaE, productos[i].mesE, productos[i].anioE);
                gotoxy(51, i+12); printf("%d/%d/%d", productos[i].diaC, productos[i].mesC, productos[i].anioC);
                gotoxy(69, i+12); printf("%d\n", productos[i].cantidad);
            }
    }
    system("pause>nul");
}
```

Una vez ordenados todos los productos en el arreglo, se presentan en pantalla todos los datos en forma de tabla para una mejor visualización al usuario. Para este paso es necesaria la función *gotoxy*, puesto que permite colocar en cualquier lugar de la consola los elementos dados.

Productos por fecha de entrada

Esta función utiliza el método de ordenamiento de inserción para poder decretar el orden de los productos, siendo en esta ocasión ordenados por medio de la fecha de entrada, colocando en primer lugar el producto con la fecha más antigua hasta llegar a la fecha más reciente.

Las fechas solo pueden ser ordenadas por al año de ingreso. El día y el mes son ignorados.

Para lograr esto, se manda a llamar a la función *obtenerProductos*, la cual extrae del archivo todos los bienes agregados y, con ello, crea un vector dinámico de productos. De esta manera es posible recorrer el arreglo creado mientras se van ordenando crecientemente.

```
void productosEntrada(){
    _almacen aux;
    Producto *productos;
    int i, pos, numeroProductos;

    system("cls"); tituloPrincipal();
    productos = obtenerProductos(&numeroProductos); // Retorna un vector dinámico de productos

    if(numeroProductos == 0){
        printf("\n\tEl archivo est%c vac%co!!\n", 160, 161);
    }else{
        colores(10); printf("\n\t\t==> LISTADO DE PRODUCTOS POR FECHA DE ENTRADA <==\n");
        printf(" ------------------------------------------------------------------------------\n");
        printf("\tID\tNOMBRE\t\t   FECHA ENTRADA   FECHA CADUCIDAD   CANTIDAD\n");
        printf(" ------------------------------------------------------------------------------\n"); colores(15);

        // Se ordena el vector dinámico de productos por el método de inserción
        for(i=1; i<numeroProductos; i++){
            aux = productos[i];
            pos = i-1;

            while((productos[pos].anioE > aux.anioE) && (pos>=0)){
                productos[pos+1] = productos[pos];
                pos--;
            }
            productos[pos+1] = aux;
        }

        // Impresión de los productos ordenados por fecha de entrada
        for(i=0; i<numeroProductos; i++){
            if(productos[i].ID != -1)
                printf("\t%d\t%s", productos[i].ID, productos[i].nombre);
                gotoxy(36, i+12); printf("%d/%d/%d", productos[i].diaE, productos[i].mesE, productos[i].anioE);
                gotoxy(51, i+12); printf("%d/%d/%d", productos[i].diaC, productos[i].mesC, productos[i].anioC);
                gotoxy(69, i+12); printf("%d\n", productos[i].cantidad);
        }
    }
    system("pause>nul");
}
```

Una vez ordenados todos los productos en el arreglo, se presentan en pantalla todos los datos en forma de tabla con ayuda de la función *gotoxy*.

Productos por fecha de caducidad

Al igual que la función anterior, esta utiliza el método de ordenamiento de inserción para poder decretar el orden de los productos, siendo en esta ocasión ordenados por medio de la fecha de caducidad, colocando en primer lugar el producto con la fecha más antigua hasta llegar a la fecha más reciente.

Las fechas solo pueden ser ordenadas por al año de caducidad. El día y el mes son ignorados.

Para lograr esto, se manda a llamar a la función *obtenerProductos*, la cual extrae del archivo todos los bienes agregados y, con ello, crea un vector dinámico de productos. De esta manera es posible recorrer el arreglo creado mientras se van ordenando crecientemente.

```
void productosCaducidad(){
    _almacen aux;
    Producto *productos;
    int i, pos, numeroProductos;

    system("cls"); tituloPrincipal();
    productos = obtenerProductos(&numeroProductos); // Retorna un vector dinámico de productos

    if(numeroProductos == 0){
        printf("\n\tEl archivo est%c vac%co!!\n", 160, 161);
    }else{
        colores(10); printf("\n\t\t==> LISTADO DE PRODUCTOS POR FECHA DE CADUCIDAD <==\n");
        printf(" ------------------------------------------------------------------------------\n");
        printf("\tID\tNOMBRE\t\t   FECHA ENTRADA   FECHA CADUCIDAD   CANTIDAD\n");
        printf(" ------------------------------------------------------------------------------\n"); colores(15);

        // Se ordena el vector dinámico de productos por el método de inserción
        for(i=1; i<numeroProductos; i++){
            aux = productos[i];
            pos = i-1;

            while((productos[pos].anioC > aux.anioC) && (pos>=0)){
                productos[pos+1] = productos[pos];
                pos--;
            }
            productos[pos+1] = aux;
        }

        // Impresión de los productos ordenados por fecha de caducidad
        for(i=0; i<numeroProductos; i++){
            if(productos[i].ID != -1)
                printf("\t%d\t%s", productos[i].ID, productos[i].nombre);
                gotoxy(36, i+12); printf("%d/%d/%d", productos[i].diaE, productos[i].mesE, productos[i].anioE);
                gotoxy(51, i+12); printf("%d/%d/%d", productos[i].diaC, productos[i].mesC, productos[i].anioC);
                gotoxy(69, i+12); printf("%d\n", productos[i].cantidad);
        }
    }
    system("pause>nul");
}
```

Una vez ordenados todos los productos en el arreglo, se presentan en pantalla todos los datos en forma de tabla con ayuda de la función *gotoxy*.

Reportes de productos

Productos próximos a caducar

Función de tipo ***char*** debido a que maneja directamente la información en el archivo. Crea un vector dinámico con la función *obtenerProductos* y, con ayuda de la función *fecha*, permite comparar la fecha de caducidad del producto con la fecha actual. El reporte solamente contiene aquellos productos que caducan en el mes en que se encuentra el usuario.

El reporte generado se guarda con el nombre de “Proximos_a_caducar.txt”.

```
char productoProxCaduc(){
    FILE *archivo;
    _almacen aux;
    Producto *productos;
    int i, numeroProductos, pos;
    char guardado, respuesta[MAX];
    int dT=0, mT=0, yT=0, dtC=0, mtC=0, ytC=0;
    int dias_meses[] = {30, 28, 31, 30, 30, 30, 30, 30, 30, 30, 31, 30};
    fecha(&dT, &mT, &yT);

    system("cls"); tituloPrincipal(); colores(10);
    productos = obtenerProductos(&numeroProductos);

    printf("\n\t\t==> PRODUCTOS PR%cXIMOS A CADUCAR EN EL PRESENTE MES <==\n", 224);
    fecha(&dT, &mT, &yT);
    printf("\n\t\t\t    .: Fecha actual %c %d/%d/%d :.\n\n", 175, dT, mT, yT);

    if(numeroProductos == 0){
        colores(15); printf("\n\tEl archivo est%c vac%co!!\n", 160, 161);
    }else{
        printf(" ------------------------------------------------------------------------------\n");
        printf("\tID\tNOMBRE\t\t   FECHA ENTRADA   FECHA CADUCIDAD   CANTIDAD\n");
        printf(" ------------------------------------------------------------------------------\n"); colores(15);

        // Se ordena el vector dinámico de productos por el método de inserción
        for(i=1; i<numeroProductos; i++){
            aux = productos[i];
            pos = i-1;
            while((productos[pos].ID > aux.ID) && (pos>=0)){
                productos[pos+1] = productos[pos];
                pos--;
            }
            productos[pos+1] = aux;
        }

        // Se recorre el vector dinámico de productos
        for(i=0; i<numeroProductos; i++){
            if((productos[i].ID!=-1) && (productos[i].diaC<=dias_meses[mT]) && (productos[i].mesC==mT) && (productos[i].anioC==yT)){
                printf("\t%d\t%s", productos[i].ID, productos[i].nombre);
                gotoxy(36, i+15); printf("%d/%d/%d", productos[i].diaE, productos[i].mesE, productos[i].anioE);
                gotoxy(51, i+15); printf("%d/%d/%d", productos[i].diaC, productos[i].mesC, productos[i].anioC);
                gotoxy(69, i+15); printf("%d\n", productos[i].cantidad);
            }
        }

        printf("\n\t%cDesea guardar el reporte en un archivo de texto? [S/N]: ", 168);
        leecad(respuesta, MAX);
        if(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0){
            if(numeroProductos==0){
                guardado = 0;
                printf("\n\tOcurri%c un error al guardar el reporte\n", 162);
            }else{
                // Abre el archivo en modo texto para escritura
                archivo = fopen("Proximos_a_caducar.txt", "w");
                if(archivo==NULL){ // Si no se pudo abrir el archivo, el valor de archivo es NULL
                    guardado = 0;
                }else{
                    fprintf(archivo, "\n\t\t==> PRODUCTOS PROXIMOS A CADUCAR EN EL PRESENTE MES <==\n");
                    fprintf(archivo, " ------------------------------------------------------------------------------\n");
                    fprintf(archivo, "\tID\tNOMBRE\t\t   FECHA ENTRADA   FECHA CADUCIDAD   CANTIDAD\n");
                    fprintf(archivo, " ------------------------------------------------------------------------------\n");

                    // Se recorre el vector dinámico de productos
                    for(i=0; i<numeroProductos; i++){
                        if((productos[i].ID!=-1) && (productos[i].diaC<=dias_meses[mT]) && (productos[i].mesC==mT) && (productos[i].anioC==yT)){
                            fprintf(archivo, "\t%d\t%s", productos[i].ID, productos[i].nombre);
                            fprintf(archivo, "\t\t%d/%d/%d", productos[i].diaE, productos[i].mesE, productos[i].anioE);
                            fprintf(archivo, "  \t%d/%d/%d", productos[i].diaC, productos[i].mesC, productos[i].anioC);
                            fprintf(archivo, "\t%d\n", productos[i].cantidad);
                        }
                    }
                    guardado = 1;

                    fclose(archivo); // Cierra el archivo
                    printf("\n\tEl reporte fue guardado con %cxito\n", 130);
                }
            }
            return guardado;
        }else printf("\n\tPresione cualquier tecla para continuar");
    }
}
```

Productos próximos a terminarse

Función de tipo ***char*** debido a que maneja directamente la información en el archivo. Crea un vector dinámico con la función *obtenerProductos*, lo cual permite recorrer el arreglo creado y verificar si la cantidad de producto almacenado es suficiente. Este proceso se realiza mediante el uso de una condicional: si la cantidad en almacén es igual o menor a cinco, entonces se muestran en pantalla esos productos; de lo contrario, el producto no se muestra.

El reporte generado se guarda con el nombre de “Proximos_a_terminar.txt”.

```
char productoProxTerminar(){
    FILE *archivo;
    _almacen aux;
    Producto *productos;
    int i, numeroProductos, pos;
    char guardado, respuesta[MAX];

    system("cls"); tituloPrincipal(); colores(10);
    productos = obtenerProductos(&numeroProductos);

    printf("\n\t\t==> PRODUCTOS PR%cXIMOS A TERMINARSE <==\n", 224);
    if(numeroProductos == 0){
        colores(15); printf("\n\tEl archivo est%c vac%co!!\n", 160, 161);
    }else{
        printf(" ------------------------------------------------------------------------------------\n");
        printf("\tID\tNOMBRE\t\t   FECHA ENTRADA   FECHA CADUCIDAD   CANTIDAD\n");
        printf(" ------------------------------------------------------------------------------------\n"); colores(15);

        // Se ordena el vector dinámico de productos por el método de inserción
        for(i=1; i<numeroProductos; i++){
            aux = productos[i];
            pos = i-1;
            while((productos[pos].ID > aux.ID) && (pos>=0)){
                productos[pos+1] = productos[pos];
                pos--;
            }
            productos[pos+1] = aux;
        }

        // Se recorre el vector dinámico de productos
        for(i=0; i<numeroProductos; i++){
            if((productos[i].ID!=-1) && (productos[i].cantidad<6)){
                gotoxy(8, i+12); printf("%d\t%s", productos[i].ID, productos[i].nombre);
                gotoxy(36, i+12); printf("%d/%d/%d", productos[i].diaE, productos[i].mesE, productos[i].anioE);
                gotoxy(51, i+12); printf("%d/%d/%d", productos[i].diaC, productos[i].mesC, productos[i].anioC);
                gotoxy(69, i+12); printf("%d", productos[i].cantidad);
            }
        }

        printf("\n\n\t%cDesea guardar el reporte en un archivo de texto? [S/N]: ", 168);
        leecad(respuesta, MAX);

        if(strcmp(respuesta, "S") == 0 || strcmp(respuesta, "s") == 0){
            if(numeroProductos==0){
                guardado = 0;
                printf("\n\tOcurri%c un error al guardar el reporte\n", 162);
            }else{
                // Abre el archivo en modo texto para escritura
                archivo = fopen("Proximos_a_terminar.txt", "w");
                if(archivo==NULL){ // Si no se pudo abrir el archivo, el valor de archivo es NULL
                    guardado = 0;
                }else{
                    fprintf(archivo, "\n\t\t==> PRODUCTOS PROXIMOS A TERMINARSE <==\n");
                    fprintf(archivo, " ------------------------------------------------------------------------------------\n");
                    fprintf(archivo, "\tID\tNOMBRE\t\t   FECHA ENTRADA   FECHA CADUCIDAD   CANTIDAD\n");
                    fprintf(archivo, " ------------------------------------------------------------------------------------\n");

                    // Se recorre el vector dinámico de productos
                    for(i=0; i<numeroProductos; i++){
                        if((productos[i].ID!=-1) && (productos[i].cantidad<6)){
                            fprintf(archivo, "\t%d\t%s", productos[i].ID, productos[i].nombre);
                            fprintf(archivo, "\t\t%d/%d/%d", productos[i].diaE, productos[i].mesE, productos[i].anioE);
                            fprintf(archivo, "  \t%d/%d/%d", productos[i].diaC, productos[i].mesC, productos[i].anioC);
                            fprintf(archivo, "\t%d\n", productos[i].cantidad);
                        }
                    }
                    guardado = 1;

                    fclose(archivo); // Cierra el archivo
                    printf("\n\tEl reporte fue guardado con %cxito\n", 130);
                }
            }
            return guardado;
        }else printf("\n\tPresione cualquier tecla para continuar");
    }
}
```

Funciones para controlar los archivos

Obtención de productos

Esta función utiliza el puntero Producto para juntar todos los datos de la estructura en un solo conjunto, siendo cada uno de estos conjuntos un elemento del arreglo dinámico creado gracias a un puntero secundario (**productos*), mismo que va contando la cantidad de elementos registrados en el archivo, y los guarda en la variable *n*.

Todo esto es posible debido a que, una vez abierto el archivo "*productos.txt*" en modo binario de lectura (rb), se posiciona el cursor al final del archivo y se reserva espacio de memoria para cada producto registrado.

Una vez hecho esto, se posiciona el cursor al inicio del archivo para almacenar cada producto junto con su información complementaria en un solo espacio del vector dinámico. Esto se repite para cada producto hasta que el cursor llegue al final del archivo.

Por último, se retorna el vector dinámico de productos.

```
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.txt", "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);
        }

        fclose(archivo); // Cierra el archivo
    }

    return productos;
}
```

Existencia de un producto

Función de tipo ***char*** cuya finalidad es comprobar la existencia de un producto dentro del registro. Para ello es necesario que la función reciba como parámetro el ID del producto a buscar y, una vez obtenido este dato, se procede a recorrer el archivo, comparando cada ID registrado con el ID que el usuario ha proporcionado.

Si el ID coincide con alguno de los existentes en el registro, entonces la función retorna una variable char llamada *existe* de valor 1. En cambio, si el ID no fue encantado, entonces la variable char *existe* retorna con un valor de cero.

```
char existeProducto(int codigoProducto, Producto *producto){
    FILE *archivo;
    char existe;

    // Abre el archivo en modo lectura
    archivo = fopen("productos.txt", "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).ID == codigoProducto){
                existe = 1;
                break;
            }
            fread(&(*producto), sizeof(*producto), 1, archivo);
        }

        fclose(archivo); // Cierra el archivo
    }

    return existe;
}
```

Inserción de un producto

Función de tipo ***char*** que permite registrar un nuevo producto dentro del archivo. Para lograr este resultado, se abre o crea (según sea el caso) un archivo en modo binario para actualización, escribiendo al final de este.

Si el archivo no puede abrirse, se retorna una variable char llamada *insercion* de valor 0. De lo contrario, se escribe al final del archivo toda la información del producto y, a su vez se retorna la variable *insercion* con un valor de 1.

```
char insertarProducto(Producto producto){
    FILE *archivo;
    char insercion;

    // Abre el archivo para agregar datos al final
    archivo = fopen("productos.txt", "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;

        fclose(archivo); // Cierra el archivo
    }

    return insercion;
}
```

Modificación de un producto

Función de tipo ***char*** que permite modificar los datos de un producto registrado en el archivo por medio de su ID. Su funcionamiento es el siguiente: Se abre el archivo "*productos.txt*" en modo binario de lectura (rb+) y se comprueba si este tiene información. De ser así, se recorre todo el archivo en busca del ID ingresado. Si el ID coincide con alguno de los registrados, entonces se procede a sobrescribir la información que había antes registrada con la nueva que el usuario ha ingresado. Al ser modificados los datos, una variable ***char*** de valor 1 se retorna.

En caso de que el archivo esté vacío, se retorna una variable ***char*** de valor cero.

```
char modificarProducto(Producto producto){
    FILE *archivo;
    char modifica;
    Producto producto2;

    archivo = fopen("productos.txt", "rb+"); // Abre el archivo para lectura/escritura

    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.ID == producto.ID) {
                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);
    }

    return modifica; // Cierra el archivo
}
```

```
void fecha(int *dT, int *mT, int *yT){
    int i=0, j=0, k=0;
    time_t tiempo = time(0);
    struct tm *tlocal = localtime(&tiempo);
    int dayNum[2], monthNum[2], yearNum[2];
    int dd[2], mm[2], yy[2];
    char day[128], month[128], year[4];
    strftime(day, 128, "%d", tlocal);
    strftime(month, 128, "%m", tlocal);
    strftime(year, 4, "%y", tlocal);

    while(i<3){
        dayNum[i] = day[i];
        monthNum[i] = month[i];
        yearNum[i] = year[i];
        i++;
    }
    i=0;

    while(i<3){
        if(dayNum[i]==48) dd[i] = 0;
        if(dayNum[i]==49) dd[i] = 1;
        if(dayNum[i]==50) dd[i] = 2;
        if(dayNum[i]==51) dd[i] = 3;
        if(dayNum[i]==52) dd[i] = 4;
        if(dayNum[i]==53) dd[i] = 5;
        if(dayNum[i]==54) dd[i] = 6;
        if(dayNum[i]==55) dd[i] = 7;
        if(dayNum[i]==56) dd[i] = 8;
        if(dayNum[i]==57) dd[i] = 9;
        i++;
    };
    i=0;
    while(i<3){
        if(monthNum[i]==48) mm[i] = 0;
        if(monthNum[i]==49) mm[i] = 1;
        if(monthNum[i]==50) mm[i] = 2;
        if(monthNum[i]==51) mm[i] = 3;
        if(monthNum[i]==52) mm[i] = 4;
        if(monthNum[i]==53) mm[i] = 5;
        if(monthNum[i]==54) mm[i] = 6;
        if(monthNum[i]==55) mm[i] = 7;
        if(monthNum[i]==56) mm[i] = 8;
        if(monthNum[i]==57) mm[i] = 9;
        i++;
    };
    i=0;

    while(i<3){
        if(yearNum[i]==48) yy[i] = 0;
        if(yearNum[i]==49) yy[i] = 1;
        if(yearNum[i]==50) yy[i] = 2;
        if(yearNum[i]==51) yy[i] = 3;
        if(yearNum[i]==52) yy[i] = 4;
        if(yearNum[i]==53) yy[i] = 5;
        if(yearNum[i]==54) yy[i] = 6;
        if(yearNum[i]==55) yy[i] = 7;
        if(yearNum[i]==56) yy[i] = 8;
        if(yearNum[i]==57) yy[i] = 9;
        i++;
    };
    *dT = ((dd[0]*10)+dd[1]);
    *mT = ((mm[0]*10)+mm[1]);
    *yT = ((yy[0]*10)+yy[1]);
}
```

Obtener fecha actual

Esta función tiene como finalidad obtener la fecha del día en el que el usuario se encuentra.

Para ello se utilizan funciones predefinidas, tales como *localtime* y *strftime*, las cuales ayudan a extraer la fecha actual del dispositivo en cuestión.

Mientras los dígitos ingresados sean menores a tres, la función compara los datos extraídos para después pasarlos a las variables correspondientes de la función que la ha llamado.

Lectura de un entero y una cadena

Para poder validar los enteros, así como las cadenas de caracteres ingresadas en cada apartado del programa, se utiliza esta función la cual, si se trata de un entero, comprueba que solo se ingresen datos numéricos. Si el dato que se pide es una cadena, la función valida los espacios y procede a leer toda la extensión de la misma.

```
int leecad(char *cad, int n){
    int i, c;
    c = getchar();
    if(c == EOF){
        cad[0] = '\0';
        return 0;
    }
    if(c == '\n'){
        i = 0;
    }else{
        cad[0] = c;
        i = 1;
    }
    for(; i < n - 1 && (c = getchar()) != EOF && c != '\n'; i++)
        cad[i] = c;
    cad[i] = '\0';

    if(c != '\n' && c != EOF) // Es un caracter
        while ((c = getchar()) != '\n' && c != EOF);

    return 1;
}
```

Título principal

Para dar mayor estética al proyecto se utiliza esta función, la cual permite visualizar en todo momento los datos de los desarrolladores del programa.

```
void tituloPrincipal(){
    colores(10); printf("\n     ==========================================================================\n");
    colores(15); printf("\t\t\t     ALMAC%cN DE ALIMENTOS\n", 144);
    printf("\t\t    Brayan Adrian Galvan Flores     181112\n");
    printf("\t\t    Julieta Rodr%cguez Espiricueta  180024\n", 161); colores(10);
    printf("     ==========================================================================\n\n"); colores(15);
}
```

Portada

Esta función permite mostrar una portada siguiendo un patrón de colores dados por una matriz previamente definida, la cual contiene datos numéricos, en los cuales cada número representa un color. Para poder visualizar la portada, es necesario leer tanto las columnas como los renglones de la matriz, siguiendo el orden de izquierda a derecha, y de arriba hacia abajo hasta abarcar toda la imagen dada.

```
void portada(){
    int i=0, j=0;

    system("MODE 120, 40"); // Tamaño de la consola -> Ancho / Alto
    gotoxy(65,10); colores(15); printf("Universidad Polit%ccnica de San Luis Potos%c", 130, 161);
    gotoxy(65,12); printf("\tProgramaci%cn II - Oto%co 2019", 162, 164);
    gotoxy(69,14); printf("  Profa. Araceli Gabriela Aguilar");
    gotoxy(64,16); colores(14); printf("-----------------------------------------------");
    gotoxy(66,18); colores(15); printf("%c Brayan Adrian Galvan Flores     181112", 4);
    gotoxy(66,19); colores(15); printf("%c Julieta Rodr%cguez Espiricueta  180024", 4, 161);
    gotoxy(0,0);

    for(i=0; i<REN; i++){ // Renglones
        for(j=0; j<COL; j++){ // Columnas
            colores(imagen[i][j]);
            printf("%c", 219); // Imprimir imagen
        }
        printf("\n");
        Sleep(3); // La portada se imprime poco a poco
    }
    colores(7); printf("Presione cualquier tecla para continuar");
    getch();
}
```

Colores

El propósito de esta función es hacer más compacto el cambio de color en el programa, permitiendo una interfaz más presentable.

```
void colores(int color){
    HANDLE hConsole;
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, color);
}
```

Diagrama de Flujo