SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

Instituto Politécnico Nacional
"La Técnica al Servicio de la Patria"

upiita-ipn

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Alumno
Córdova Fernández Karla Lilia

Unidad de Aprendizaje: Programación Avanzada

Profesor
M. en C. Niels Henrik Navarrete Manzanilla

Práctica 2
Memoria dinámica

Ciudad de México; a 17 de enero de 2020.

Índice

INTRODUCCIÓN

Los programas pueden crear variables globales o locales. Las variables declaradas globales en sus programas se almacenan en posiciones fijas de memoria, en la zona conocida como segmento de datos del programa, y todas las funciones pueden utilizar estas variables. Las variables locales se almacenan en la pila (stack) y existen sólo mientras están activas las funciones que están declaradas.

Es posible, también, crear variables static (similares a las globales) que se almacenan en posiciones fijas de memoria, pero sólo están disponibles en

el módulo (es decir, el archivo de texto) o función en que se declaran; su espacio de almacenamiento es el segmento de datos. Todas estas clases de variables comparten una característica común: se definen cuando se compila el programa. Esto significa que el compilador reserva (define) espacio para almacenar valores de los tipos de datos declarados. Es decir, en el caso de las variables globales y locales se ha de indicar al compilador exactamente cuántas y de qué tipo son las variables a asignar. O sea, el espacio de almacenamiento se reserva en el momento de la Compilación.

Sin embargo, no siempre es posible conocer con antelación a la ejecución cuanta memoria se debe reservar al programa. En C, se asigna memoria en el

momento de la ejecución en el montículo o montón (heap), mediante las funciones mallo@( ), realloo( ), calloa( ) y free( ), que asignan y liberan la

memoria de una zona denominada almacén libre.

DESARROLLO

PROGRAMA CON FUNCIONES RECURSIVAS

Código del programa principal (función main y función para rellenar un arreglo de enteros).

```c
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include "recursividad.h"

void rellenar(int *arreglo,int longitud);

int main(){
    int *arreglo;
    int decimal, error;
    int longitud = 0,menu = 1;
    char *binario;

while(menu!=0){
    error = 0;
    system("cls");
    printf("MENU");
    printf("\n1) Ordenar de menor a mayor.");
    printf("\n2) De decimal a binario.");
    printf("\n3) De binario a decimal.");
    printf("\n4) Invertir arreglo de enteros.");
    printf("\n0) Salir.");
    scanf("%d",&menu);
    fflush(stdin);
```

```c
    switch(menu){
        case 1:
            do{
                printf("Introduce longitud del arreglo: ");
                error = scanf("%d",&longitud);
                fflush(stdin);
                if(error == 0)
                    printf("Dato incorrecto!\n");
            }while(error == 0);
```

```
        arreglo = (int*)malloc(sizeof(int)*longitud);
        rellenar(arreglo,longitud);
        printf("\nArreglo original:\n");
        leerArreglo(arreglo,(arreglo+longitud-1));
        menor_aMayor(arreglo,(arreglo+longitud));
        printf("\nArreglo ordenado:\n");
        leerArreglo(arreglo,(arreglo+longitud-1));
        getch();
    break;

    case 2:
        binario = (char*)malloc(sizeof(char)*100);
        do{
            printf("Introduce el numero entero decimal: ");
            error = scanf("%d",&decimal);
            fflush(stdin);
            if(error == 0)
                printf("Dato incorrecto!\n");
        }while(error == 0);

        binario = decimal_aBinario(decimal,binario,strlen(binario));
        printf("\nEl numero en binario es: %s\n",binario);
        getch();
    break;
```

```
    case 3:
        binario = (char*)malloc(sizeof(char)*100);
        printf("Introduce el numero entero binario: ");
        scanf("%s",binario);
        fflush(stdin);
        for(int i=0 ; i<strlen(binario) ; i++){
            if(*(binario+i) != '0' && *(binario+i) != '1'){
                error = 1;
            }
        }
        (error == 1) ? printf("El numero introducido no es valido!") :
        printf("\nEl numero en decimal es: %d\n",binario_aDecimal(binario));
        getch();
    break;
```

```
        case 4:
            do{
                printf("Introduce longitud del arreglo: ");
                error = scanf("%d",&longitud);
                fflush(stdin);
                if(error == 0)
                    printf("Dato incorrecto!\n");
            }while(error == 0);

            arreglo = (int*)malloc(sizeof(int)*longitud);
            rellenar(arreglo,longitud);
            printf("\nArreglo original:\n");
            leerArreglo(arreglo,(arreglo+longitud-1));
            invertirNum(arreglo,(arreglo+longitud-1));
            printf("\nArreglo invertido:\n");
            leerArreglo(arreglo,(arreglo+longitud-1));
            getch();
        break;
    }
}

    return 0;
}
```

```
void rellenar(int *arreglo,int longitud){
    int error = 0;
    for(int i=0; i<longitud; i++){
        do{
            printf("Arreglo[%d] = ",i);
            error = scanf("%d",(arreglo+i));
            fflush(stdin);
            if(error == 0)
                printf("Tipo de dato incorrecto, vuelva a escribirlo.\n");
        }while(error == 0);
    }
}
```

PROGRAMA 1. ALGORITMO BURBUJA Y DE SELECCIÓN

Descripción

Desarrollar un programa que ordene un arreglo. *el arreglo debe ser dinámico.

a) Usando el algoritmo de burbuja.

b) Usando el algoritmo de selección.

Análisis

1. **¿Cuáles son la entradas y salidas?**

Entradas:

int *arreglo; puntero de un arreglo de enteros.

int longitud; longitud del arreglo dinámico.

2. **¿Qué es lo que hará el programa?**

El programa creará un arreglo de enteros con la longitud definida por el usuario. El programa llamará una función con el método burbuja para ordenarlo, comparando uno por uno los valores del arreglo empezando por la primera posición, e intercambiando este valor por el de otra posición si ese es más pequeño.

El programa llamará a otra función para ordenar una copia del arreglo original por el método de selección, donde se compara los números por posición, el programa busca al número más pequeño del arreglo y lo posiciona en el primer lugar, el segundo más pequeño en la segunda, y así sucesivamente.

3. **¿Qué espero de salida?**

La salida será el arreglo ya ordenado.

Requerimientos Funcionales

El programa acepta cualquier longitud de arreglo.

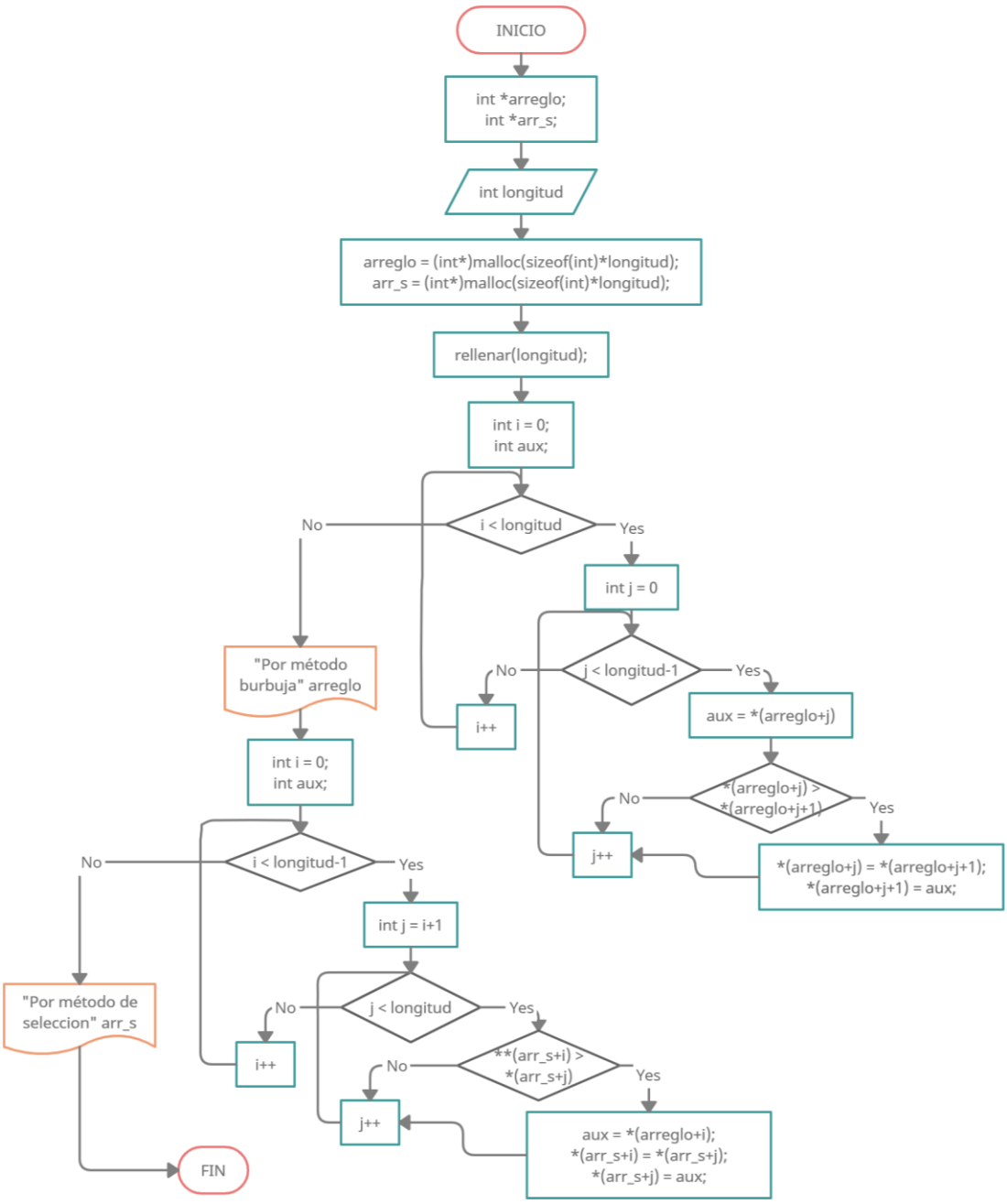
El programa únicamente acepta valores de tipo entero.

El programa ordena el arreglo con dos métodos diferentes, con método burbuja y método de selección.

Requerimientos no Funcionales

El programa no acepta ningún otro valor que no sea entero (negativo o positivo).

Diagrama de flujo



Código e C

```c
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

void rellenar(int longitud);
void imprimir(int longitud);
void burbuja(int longitud);
void seleccion(int longitud);

int *arreglo, *arr_s;

int main(){
    int longitud = 0, menu = 1, error;

while(menu!=0){
    system("cls");
    printf("\n1) Ordenar un arreglo de numeros enteros.");
    printf("\n0) Salir.");
    error = scanf("%d",&menu);
    fflush(stdin);
```

```c
    if(menu == 1 && error!=0){
            printf("Introduce longitud del arreglo: ");
            scanf("%d",&longitud);
            fflush(stdin);
            arreglo = (int*)malloc(sizeof(int)*longitud);
            arr_s = (int*)malloc(sizeof(int)*longitud);
            rellenar(longitud);
            system("cls");
            printf("\nArreglo original:");
            imprimir(longitud);
            printf("\n\nPor metodo burbuja:\n");
            burbuja(longitud);
            imprimir(longitud);
            printf("\n\nPor metodo de seleccion:\n");
            seleccion(longitud);
            imprimir(longitud);
            getch();
    }
}

    return 0;
}
```

```c
void rellenar(int longitud){
    int error;
    for(int i=0; i<longitud; i++){
        do{
            printf("Arreglo[%d] = ",i);
            error = scanf("%d",(arreglo+i));
            fflush(stdin);
            if(error == 0)
                printf("El dato es incorrecto!\n");
        }while(error==0);
        *(arr_s+i) = *(arreglo+i);
    }
}

void imprimir(int longitud){
        for(int i=0; i<longitud; i++){
        printf("[%d] ",*(arreglo+i));
    }
}
```

```c
void burbuja(int longitud){
    int aux = 0;
    for(int i=0; i<longitud; i++){
        for(int j=0; j<longitud-1; j++){
            aux = *(arreglo+j);
            if(*(arreglo+j) > *(arreglo+j+1) ){
                *(arreglo+j) = *(arreglo+j+1);
                *(arreglo+j+1) = aux;
            }
        }
    }
}

void seleccion(int longitud){
    int aux = 0;
    for(int i=0; i<longitud-1; i++){
        for(int j=i+1; j<longitud; j++){
            if(*(arreglo+i) > *(arreglo+j)){
                aux = *(arreglo+i);
                *(arreglo+i) = *(arreglo+j);
                *(arreglo+j) = aux;
            }
        }
    }
}
```

Resultados (compilación)

```
1) Ordenar un arreglo de numeros enteros.
0) Salir.1
Introduce longitud del arreglo: 7
Arreglo[0] = 4
Arreglo[1] = 6
Arreglo[2] = -7
Arreglo[3] = 8
Arreglo[4] = 12
Arreglo[5] = 0
Arreglo[6] = -1
```

```
Arreglo original:[4] [6] [-7] [8] [12] [0] [-1]

Por metodo burbuja:
[-7] [-1] [0] [4] [6] [8] [12]

Por metodo de seleccion:
[-7] [-1] [0] [4] [6] [8] [12]
```

PROGRAMA 2. INFORMACIÓN DE EMPLEADO

Descripción

La información con que se cuenta por el empleado es: nombre, sexo y sueldo. Por tanto se pide: Realizar un programa que lea en un array de estructuras de los datos de N trabajadores de la empresa y que imprima los datos del empleado con mayor y menor salario.

Análisis

4. ¿Cuáles son la entradas y salidas?

Entradas:

int longitud; longitud del arreglo dinámico de empleados;

struct empleado *Empleados (nombre, sexo y sueldo); información de cada uno de los empleados del arreglo.

5. ¿Qué es lo que hará el programa?

El programa crea el arreglo con la longitud establecida por el usuario. El usuario introduce toda la información de los empleados, que son nombre, sexo y sueldo. El programa dará la opción de mostrar a todos los empleados con su información, además de mostrar al empleado con mayor sueldo y el de menor sueldo.

6. ¿Qué espero de salida?

Se retorna el arreglo de empleados o al empleado con mayor y menor sueldo según sea el caso.

Requerimientos Funcionales

El programa acepta cualquier longitud de arreglo.

El programa muestra todos los empleados escritos.

El programa permite introducir nuevamente la información de los empleados.

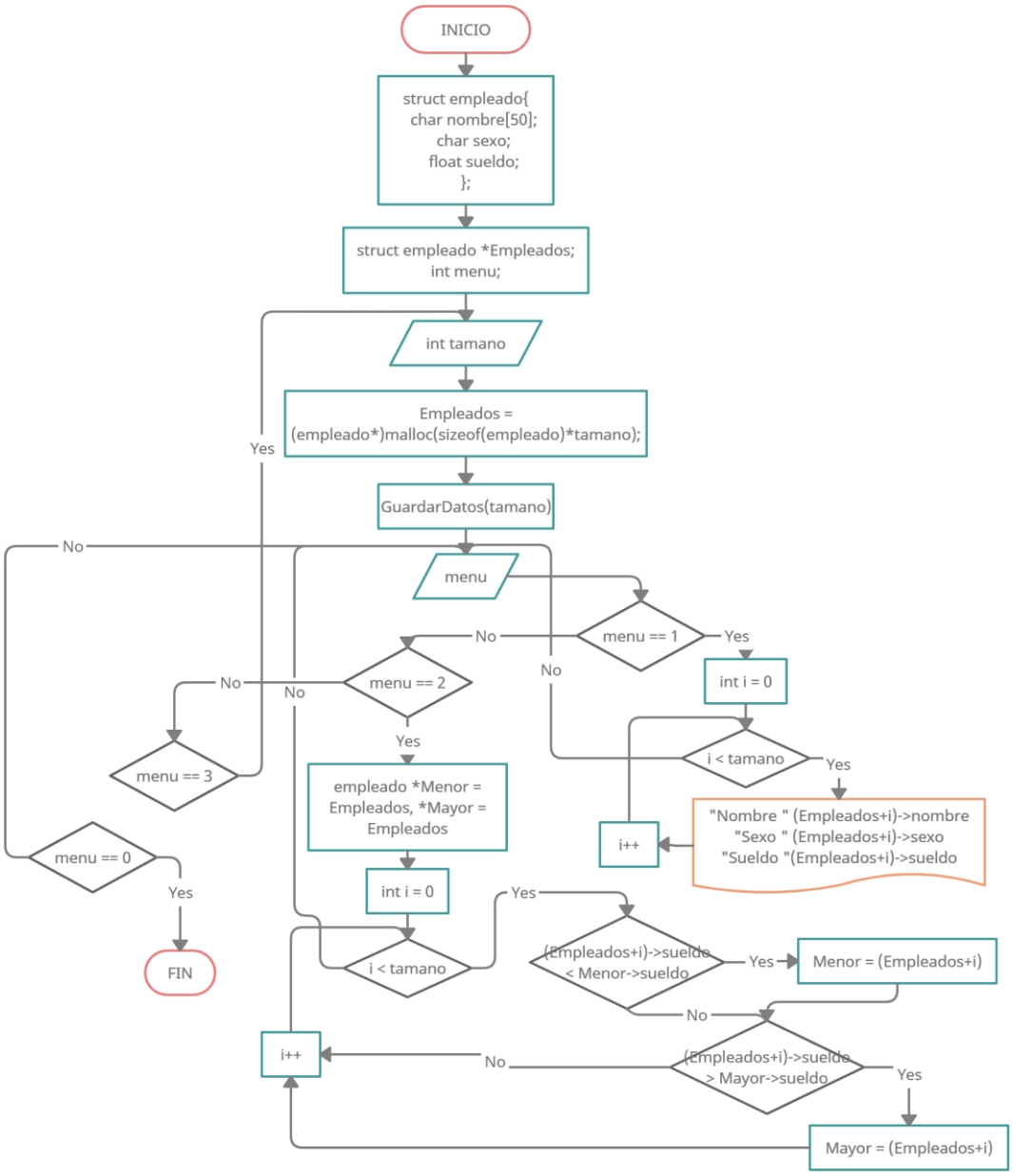
El programa muestra al empleado con mayor sueldo y al de menor sueldo en el arreglo.

Requerimientos no Funcionales

El programa no permite modificar la información del arreglo de empleados, eliminar o agregar a uno.

Si más de un empleado tiene el mismo salario más alto o más bajo, sólo muestra al primer empleado.

Diagrama de flujo



Código e C

```c
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>

void GuardarDatos(int tamano);
void ImprimirDatos(int tamano);
void menor_yMayor(int tamano);

struct empleado{
    char nombre[50];
    char sexo;
    float sueldo;
};

struct empleado *Empleados;

int main(){
    int tamano, menu;

do{
    system("cls");
    printf("Ingresa cantidad de empleados:");
    scanf("%d",&tamano);
    fflush(stdin);
    Empleados = (empleado*)malloc(sizeof(empleado)*tamano);
    GuardarDatos(tamano);
```

```c
    do{
        system("cls");
        printf("1) Listar todos los empleados.\n");
        printf("2) Mostrar al empleado con menor y mayor sueldo.\n");
        printf("3) Reingresar datos de empleados.\n");
        printf("0) Salir.");
        scanf("%d",&menu);
        fflush(stdin);

        switch(menu){
            case 1:
                ImprimirDatos(tamano);
                getch();
                break;

            case 2:
                menor_yMayor(tamano);
                getch();
                break;
```

```
    }while(menu != 0 && menu != 3);
}while(menu!=0);

    free(Empleados);
    return 0;
}

void GuardarDatos(int tamano){
    for(int i=0; i<tamano; i++){
        printf("\n\nEmpleado %d:\n",i);
        printf("Ingrese nombre del empleado: ");
        scanf("%[^\n]",&*(Empleados+i)->nombre);
        fflush(stdin);
        printf("Ingrese sexo del empleado: ");
        scanf("%c",&(Empleados+i)->sexo);
        fflush(stdin);
        printf("Ingrese sueldo del empleado: ");
        scanf("%f",&(Empleados+i)->sueldo);
        fflush(stdin);
    }
}

void ImprimirDatos(int tamano){
    for(int i=0; i<tamano; i++){
        printf("\n\nEmpleado %d:\n",i);
        printf("Nombre: %s\n",(Empleados+i)->nombre);
        printf("Sexo: %c\n",(Empleados+i)->sexo);
        printf("Sueldo: %.2f\n",(Empleados+i)->sueldo);
    }
```

```
void menor_yMayor(int tamano){
    empleado *Menor = Empleados, *Mayor = Empleados;
    for(int i=0; i<tamano; i++){
        if((Empleados+i)->sueldo < Menor->sueldo)
            Menor = (Empleados+i);
        if((Empleados+i)->sueldo > Mayor->sueldo)
            Mayor = (Empleados+i);
    }
    printf("\n\nEmpleado de menor sueldo:\n");
    printf("Nombre: %s\n",Menor->nombre);
    printf("Sexo: %c\n",Menor->sexo);
    printf("Sueldo: %.2f\n",Menor->sueldo);

    printf("\n\nEmpleado de mayor sueldo:\n");
    printf("Nombre: %s\n",Mayor->nombre);
    printf("Sexo: %c\n",Mayor->sexo);
    printf("Sueldo: %.2f\n",Mayor->sueldo);

}
```

Resultados (compilación)

```
Empleado 0:
Nombre: MARIA TORRES
Sexo: M
Sueldo: 467874.13


Empleado 1:
Nombre: JUAN PEREZ
Sexo: H
Sueldo: 45775.23


Empleado 2:
Nombre: PABLO GARCIA
Sexo: H
Sueldo: 789954.00


Empleado 3:
Nombre: LINDA TORRES
Sexo: M
Sueldo: 467885.34


Empleado 4:
Nombre: TOÑO LOPEZ
Sexo: H
Sueldo: 5678.34
```

```
1) Listar todos los empleados.
2) Mostrar al empleado con menor y mayor sueldo.
3) Reingresar datos de empleados.
0) Salir.2


Empleado de menor sueldo:
Nombre: TOÑO LOPEZ
Sexo: H
Sueldo: 5678.34


Empleado de mayor sueldo:
Nombre: PABLO GARCIA
Sexo: H
Sueldo: 789954.00
```

PROGRAMA 3. PRODUCTOS

Descripción

En una tienda donde solo hay 10 productos se desea calcular el total de ganancia que produjo dicho negocio este mes. Para ello se cuenta por producto con: Precio del Costo, precio de venta, código y cantidad vendida en ese periodo. Realizar programa que permita calcular las Ganancias de la Tienda.

Análisis

7. ¿Cuáles son la entradas y salidas?

Entradas:

struct producto *inventario; se editarán los datos de cada producto dentro del arreglo, que son precio_venta, precio_costo, cantidad vendida. El código no se edita.

8. ¿Qué es lo que hará el programa?

El programa inicialmente creará un arreglo dinámico de diez elementos de tipo struct producto, donde los datos se establecerán por default. El programa contará con una función para poder editar los productos, eligiendo el producto que se quiere editar escribiendo su número de código. Otra función se encargará de listar todos los productos con su información y la última función calculará la ganancia. La ganancia se calcula como:

Total de costo = la sumatoria de (precio_costo del artículo)*(cantidad_vendida)

Total de venta = la sumatoria de (precio_venta del artículo)*(cantidad_vendida)

Ganancia = total de venta – total de costo.

9. ¿Qué espero de salida?

Según se solicite, se recibirá de salida la lista de productos o la ganancia.

Requerimientos Funcionales

El programa permite editar el precio de costo, de venta y la cantidad de productos vendidos.

El programa avisa si hay un error en la búsqueda del código del producto en la sección de editar y solicita el código hasta que sea correcto.

El programa muestra el valor anterior del dato que se desea editar antes de modificarlo.

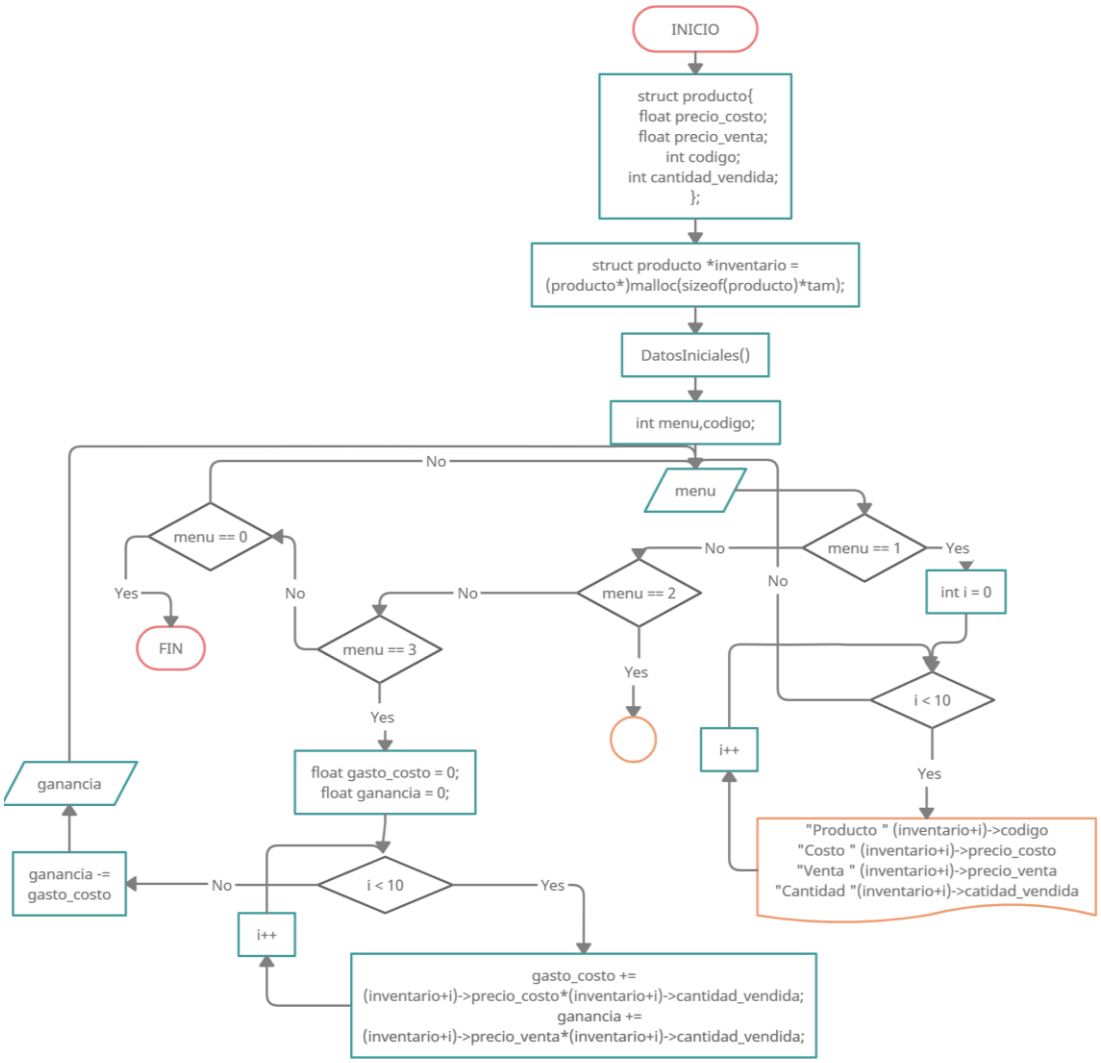
El programa calcula la ganancia de la tienda.

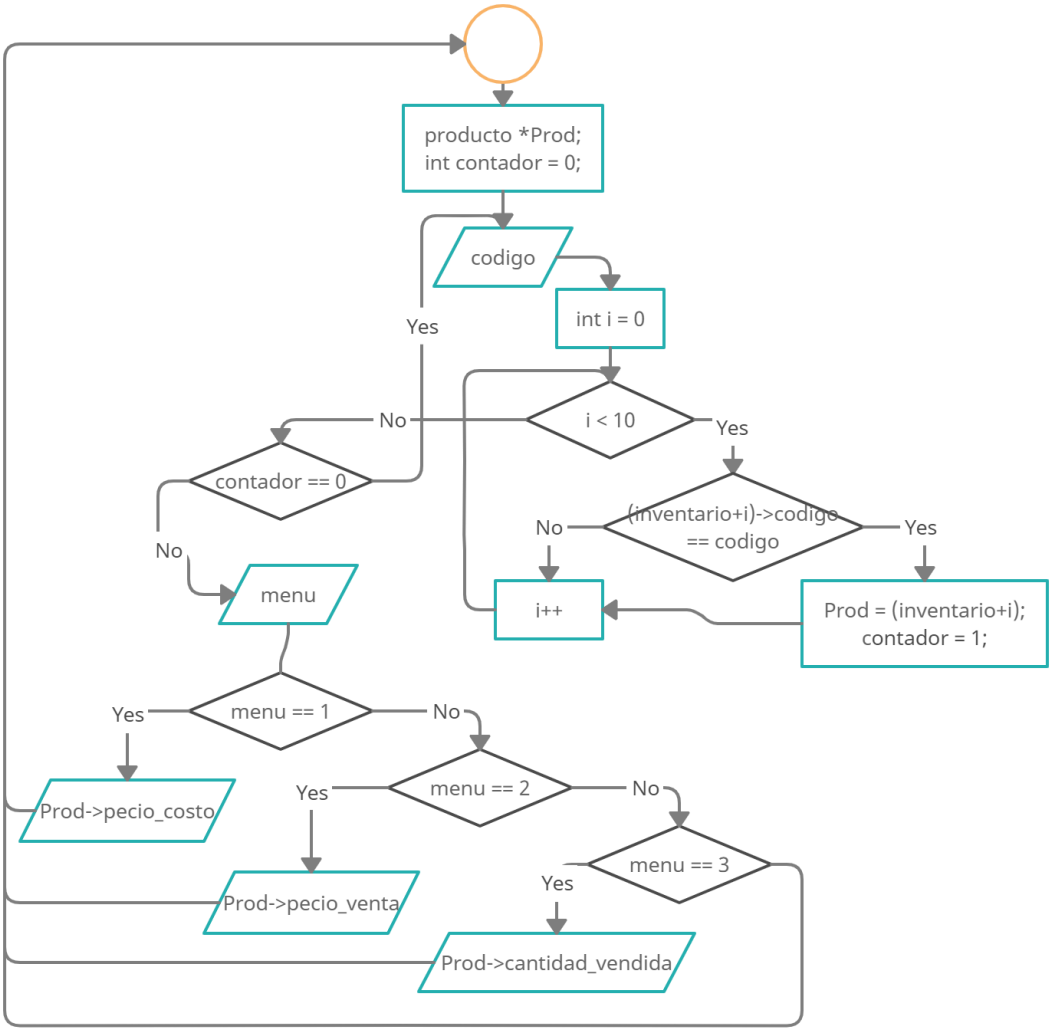
El programa muestra la lista de todos los productos con su información.

Requerimientos no Funcionales

El programa no agrega o elimina productos del arreglo de productos.

Diagrama de flujo




producto *Prod;
int contador = 0;
codigo
int i = 0
i < 10
Yes
No
contador == 0
Yes
No
menu
(inventario+i)->codigo
== codigo
No
Yes
i++
Prod = (inventario+i);
contador = 1;
menu == 1
Yes
No
Prod->pecio_costo
menu == 2
Yes
No
Prod->pecio_venta
menu == 3
Yes
Prod->cantidad_vendida

Código e C

```c
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#define tam 10

struct producto{
    float precio_costo;
    float precio_venta;
    int codigo;
    int cantidad_vendida;
};

void DatosIniciales();
void ImprimirDatos();
void Ganancia();
void EditarDatos(producto *Editable);

struct producto *inventario = (producto*)malloc(sizeof(producto)*tam);;

int main(){
    DatosIniciales();
    int menu,codigo;
```

```c
    do{
        system("cls");
        printf("1) Listar informacion de productos.\n");
        printf("2) Editar datos.\n");
        printf("3) Calcular ganancia.\n");
        printf("0) Salir.");
        scanf("%d",&menu);
        fflush(stdin);

        switch(menu){
            case 1:
                ImprimirDatos();
                getch();
                break;

            case 2:
                producto *Prod;
                int contador;
                contador = 0;
                do{
                    printf("\n\nIntroduzca codigo del producto a editar (del 0 al 9):\n");
                    scanf("%d",&codigo);
                    fflush(stdin);
                    for(int i=0; i<10; i++){
                        if((inventario+i)->codigo == codigo){
                            Prod = (inventario+i);
                            contador = 1;
                        }
                    }
```

```
                    if(contador == 0)
                        printf("El codigo es incorrecto!");
                }while(contador == 0);
                EditarDatos(Prod);
                getch();
                break;

            case 3:
                Ganancia();
                getch();
                break;
        }
    }while(menu != 0);
    free(inventario);

    return 0;
}

void DatosIniciales(){
    for(int i=0; i<10; i++){
        (inventario+i)->codigo = i;
        (inventario+i)->precio_costo = 2;
        (inventario+i)->precio_venta = 5;
        (inventario+i)->cantidad_vendida = 2;
    }
}
```

```
void ImprimirDatos(){
    for(int i=0; i<10; i++){
        printf("\n\nProducto %d:\n",(inventario+i)->codigo);
        printf("Precio de costo: $%.2f\n",(inventario+i)->precio_costo);
        printf("Precio de venta: $%.2f\n",(inventario+i)->precio_venta);
        printf("Cantidad vendida: %d\n",(inventario+i)->cantidad_vendida);
    }
}

void Ganancia(){
    float gasto_costo = 0;
    float ganancia = 0;
    for(int i=0; i<10; i++){
        gasto_costo += (inventario+i)->precio_costo*(inventario+i)->cantidad_vendida;
        ganancia += (inventario+i)->precio_venta*(inventario+i)->cantidad_vendida;
    }
    printf("\nCosto total: %.2f\nVentas total: %.2f\n",gasto_costo,ganancia);
    ganancia -= gasto_costo;
    printf("Ganancia: %.2f",ganancia);
}
```

```c
void EditarDatos(producto *Editable){
    int menu;
    printf("\n\n1) Editar precio de costo.\n");
    printf("2) Editar precio de venta.\n");
    printf("3) Editar cantidad vendida.\n");
    scanf("%d",&menu);
    fflush(stdin);
    switch(menu){
        case 1:
            printf("Precio anterior: $%.2f\nPrecio nuevo: $",Editable->precio_costo);
            scanf("%f",&Editable->precio_costo);
            break;
        case 2:
            printf("Precio anterior: $%.2f\nPrecio nuevo: $",Editable->precio_venta);
            scanf("%f",&Editable->precio_venta);
            break;
        case 3:
            printf("Cantidad anterior: %d\nCantidad nueva: ",Editable->cantidad_vendida);
            scanf("%d",&Editable->cantidad_vendida);
            break;
    }
}
```

Resultados (compilación)

```
1) Listar informacion de productos.
2) Editar datos.
3) Calcular ganancia.
0) Salir.3

Costo total: 567.50
Ventas total: 1341.94
Ganancia: 774.44
```

```
Producto 3:
Precio de costo: $10.00
Precio de venta: $17.00
Cantidad vendida: 20


Producto 4:
Precio de costo: $2.00
Precio de venta: $5.00
Cantidad vendida: 2


Producto 5:
Precio de costo: $2.00
Precio de venta: $5.00
Cantidad vendida: 2


Producto 6:
Precio de costo: $2.00
Precio de venta: $5.00
Cantidad vendida: 2


Producto 7:
Precio de costo: $12.00
Precio de venta: $34.12
Cantidad vendida: 12


Producto 8:
Precio de costo: $2.00
Precio de venta: $5.00
Cantidad vendida: 2


Producto 9:
Precio de costo: $5.65
Precio de venta: $14.75
Cantidad vendida: 30
```

CONCLUSIONES

Los arreglos dinámicos pueden ser ordenados basados en un criterio como cualquier arreglo estático, con el uso de algún auxiliar y la comparación de valores.

Los arreglos dinámicos de estructuras como tipo de dato, no tienen gran diferencia con los arreglos estáticos al momento de acceder a un dato de la estructura, la sintaxis cambia, sustituyendo el punto por ->.