Universidad Autónoma de Baja California

Ingeniería en Software y Tecnologías Emergentes.

Práctica 8. Estructuras.

Materia: Lenguaje C.

Maestro: Yulith Altamirano

Alumno: Diego Quiros 372688

Fecha: 11/22/2023

Repositorio:

```
C/C++
https://github.com/diegovq12/LenguajeC-QVDD_932/tree/main/Practicas/Practica8
```

Instrucciones:

Desarrollen el código en lenguaje C y elaboren el diagrama de flujo correspondiente para los ejercicios. Será suficiente con un archivo .cpp que contenga todos los ejercicios organizados en un menú.

Ejercicios.

1. **Crear una estructura:** Inicializa un inventario vacío con una capacidad máxima utilizando una estructura llamada “Producto” para representar los elementos del inventario. La estructura “Producto” debe incluir campos como nombre, cantidad y precio.

```
C/C++
typedef struct _Producto
{
  char nombre[20];
  int cantidad;
  float precio;
} Tproducto;
```

2. Presenta al usuario un menú que le permita realizar las siguientes operaciones:
-Agregar elementos al inventario
-Retirar elementos del inventario
-Mostrar inventario
-Calcular valor total del inventario
-ordenar inventario
-Salir del programa

```
C/C++
int msge(void)
{
    int op;
```

```
    printf("\n1. AGREGAR ELEMENTOS AL INVENTARIO\n");
    printf("2. RETIRAR ELEMENTOS AL INVENTARIO\n");
    printf("3. MOSTRAR INVENTARIO\n");
    printf("4. CALCULAR VALOR TOTAL DEL INVENTARIO\n");
    printf("5. ORDENAR PRODUCTOS\n");
    printf("0. SALIR DEL PROGRAMA\n");
    fflush(stdin);
    scanf("%d", &op);
    return op;
}

void menu(void)
{
    int op;
    int numElem = 0;
    Tproducto inventario[MAX];

    do
    {
        system("CLS");
        op = msge();
        system("CLS");

        switch (op)
        {
        case 1:
            agregarElementos(inventario, numElem);
            numElem++;
            printf("\n%d\n", numElem);
            break;
        case 2:
            retirarElemento(inventario, numElem);
            break;
        case 3:
            mostrarInventario(inventario, numElem);
            break;
        case 4:
            calcularValorTotal(inventario, numElem);
            break;
        case 5:
            menuOrd(inventario, numElem);
            break;
        default:
            printf("GRACIAS POR USAR!");
            break;
        }
        system("PAUSE");
    } while (op != 0);
```

```
}
```

Agregar: Permite al usuario ingresar el nombre, cantidad y precio del producto y agrega un nuevo elemento al inventario.

```
C/C++
void agregarElementos(Tproducto invent[], int numElem)
{
  if (numElem < MAX)
  {
    printf("INGRESE EL NOMBRE DEL PRODUCTO: ");
    fflush(stdin);
    gets(invent[numElem].nombre);
    printf("INGRESE LA CANTIDAD DEL PRODUCTO: ");
    scanf("%d", &invent[numElem].cantidad);
    printf("INGRESE EL PRECIO DEL PRODUCTO: ");
    scanf("%f", &invent[numElem].precio);
    printf("PRODUCTO AGREGADO AL INVENTARIO.\n");
  }
  else
  {
    printf("EL INVENTARIO ESTÁ LLENO. NO SE PUEDEN AGREGAR MÁS
PRODUCTOS.\n");
  }
}
```

Retirar: Permite al usuario ingresar el nombre del producto que desea retirar y elimina ese elemento del inventario.

```
C/C++
void retirarElemento(Tproducto inventario[], int numElem)
{
  char nombreBuscado[20];
  int num;
  printf("INGRESE NOMBRE DE PRODUCTO A RETIRAR: ");
  fflush(stdin);
  gets(nombreBuscado);
  int encontrado = 0;
  for (int i = 0; i < numElem; i++)
  {
    if (strcmp(inventario[i].nombre, nombreBuscado) == 0)
    {
      encontrado = 1;
      printf("CUANTAS UNIDADES DE ESTE PRODUCTO DESEA ELIMINAR: ");
```

```
      fflush(stdin);
      scanf("%d", &num);

      if (num <= inventario[i].cantidad)
      {
        inventario[i].cantidad -= num;
      }
      printf("SE HAN RETIRADO %d UNIDADES DE %s\n", num,
inventario[i].nombre);
      printf("QUEDAN %d UNIDADES\n", inventario[i].cantidad);
    }
  }

  if (!encontrado)
  {
    printf("PRODUCTO NO ENCONTRADO EN EL INVENTARIO\n");
  }
}
```

Mostrar elementos del inventario:Muestra al usuario el contenido actual del inventario, incluyendo el nombre, cantidad y precio de cada producto.

```
C/C++
void mostrarInventario(Tproducto inventario[], int numElem)
{
  if (numElem == 0)
  {
    printf("INVENTARIO VACIO\n");
  }
  else
  {
    printf("INVENTARIO: \n");
    for (int i = 0; i < numElem; i++)
    {
      printf("NOMBRE: %s, CANTIDAD: %d, PRECIO: %.2f\n",
          inventario[i].nombre, inventario[i].cantidad, inventario[i].precio);
    }
  }
}
```

Calcular valor total
Agrega una opción al menú que calcule y muestre el valor total del inventario, que es la suma del precio de cada producto multiplicado por su cantidad en stock.

```C/C++
void calcularValorTotal(Tproducto inventario[], int numElem)
{
  float valorTotal = 0.0;
  for (int i = 0; i < numElem; i++)
  {
    valorTotal += inventario[i].cantidad * inventario[i].precio;
  }
  printf("El valor total del inventario es: %.2f\n", valorTotal);
}
```

Ordenar:
Permite al usuario ordenar los productos en el inventario por nombre, cantidad o precio, según su elección.

```C/C++
int msgeOrdenar()
{
  int op;
  printf("1. NOMBRE\n");
  printf("2. CANTIDAD\n");
  printf("3. PRECIO\n");
  printf("0. VOLVER\n");
  printf("INGRESA DE QUE MANERA DESEA ORDENAR LOS PRODUCTOS: ");
  fflush(stdin);
  scanf("%d", &op);
  return op;
}

void menuOrd(Tproducto vect[], int i)
{
  int op;
  int ordenado = 0;

  do
  {
    op = msgeOrdenar();

    switch (op)
    {
    case 1:
      ordenarNombre(vect, i);
      printf("PRODUCTOS ORDENADOS POR NOMBRE.\n");
      ordenado = 1;
      break;
    case 2:
      ordenarCantidad(vect, i);
```

```
      printf("PRODUCTOS ORDENADOS POR CANTIDAD.\n");
      ordenado = 1;
      break;
    case 3:
      ordenarPrecio(vect, i);
      printf("PRODUCTOS ORDENADOS POR PRECIO.\n");
      ordenado = 1;
      break;

    default:
      printf("VOLVIENDO AL MENU\n");
      break;
    }
  } while (op != 0 && ordenado == 0);
}

void ordenarNombre(Tproducto vector[], int m)
{
  int i, j;
  Tproducto aux;

  for (i = 0; i < m - 1; i++)
  {
    for (j = i + 1; j < m; j++)
    {
      if (vector[i].nombre[0] > vector[j].nombre[0])
      {
        aux = vector[i];
        vector[i] = vector[j];
        vector[j] = aux;
      }
    }
  }
}

void ordenarCantidad(Tproducto vector[], int m)
{
  int i, j;
  Tproducto aux;

  for (i = 0; i < m - 1; i++)
  {
    for (j = i + 1; j < m; j++)
    {
      if (vector[i].cantidad > vector[j].cantidad)
      {
        aux = vector[i];
        vector[i] = vector[j];
```

```
        vector[j] = aux;
      }
    }
  }
}

void ordenarPrecio(Tproducto vector[], int m)
{
  int i, j;
  Tproducto aux;

  for (i = 0; i < m - 1; i++)
  {
    for (j = i + 1; j < m; j++)
    {
      if (vector[i].precio > vector[j].precio)
      {
        aux = vector[i];
        vector[i] = vector[j];
        vector[j] = aux;
      }
    }
  }
}
```