Productos

```
#include<iostream>
#include<conio.h>

#define DIM 2
using namespace std;

struct Productos {
    int codigo;
    int precioCompra;
    int precioVenta;
    int stockInicial;
    int stockActual;
    int stockMinimo;
    int acumVentasProducto;
};

void cargarProducto(Productos vecProductos[], int& dim) {
    int i = 0, auxCodigo, resp;

    cout << "Tiene productos para ingresar? (SI:1 - NO:0): ";
    cin >> resp;

    while (resp != 1 && resp != 0) {
        cout << "ERROR!. Ingrese una respuesta válida: ";
        cin >> resp;
    }

    while (resp == 1) {
        cout << "__________CARGA DE PRODUCTOS__________" << endl;
        cout << "Ingrese el código del producto: ";
        cin >> auxCodigo;

        while (auxCodigo == 0) {
            cout << "ERROR!. El código debe ser diferente de 0." << endl;
            cout << "Ingrese el código del producto: ";
            cin >> auxCodigo;
        }
        vecProductos[i].codigo = auxCodigo;

        cout << "Ingrese el precio de compra: ";
        cin >> vecProductos[i].precioCompra;

        cout << "Ingrese el precio de venta: ";
        cin >> vecProductos[i].precioVenta;

        cout << "Ingrese el stock inicial: ";
        cin >> vecProductos[i].stockInicial;
```

```
        cout << "Ingrese el stock mínimo: ";
        cin >> vecProductos[i].stockMinimo;

        vecProductos[i].stockActual = vecProductos[i].stockInicial;
        vecProductos[i].acumVentasProducto = 0;
        i++;

        cout << "Tiene productos para ingresar? (SI:1 - NO:0): ";
        cin >> resp;

        while (resp != 1 && resp != 0) {
            cout << "ERROR!. Ingrese una respuesta válida: ";
            cin >> resp;
        }
    }
    dim = i;
}

void buscarCodigo(Productos vecProductos[], int dim, int codigo, int&
indiceCodigo) {
    int i = 0;
    while (i < dim && vecProductos[i].codigo != codigo) {
        i++;
    }
    if (i == dim) {
        cout << "ERROR!. No se encontró el código." << endl;
        cout << "Ingrese nuevamente el código: ";
        cin >> codigo;
        buscarCodigo(vecProductos, dim, codigo, indiceCodigo);
    }
    indiceCodigo = i;
}

void cargaVenta(Productos vecProductos[], int dim) {
    int codigo, indiceCodigo, venta;

    cout << "_________CARGA DE VENTAS_________" << endl;

    // Pide el código del producto vendido
    cout << "Ingrese el código del producto vendido (0 para finalizar): ";
    cin >> codigo;

    while (codigo != 0) {
        // Busca el código del producto
        buscarCodigo(vecProductos, dim, codigo, indiceCodigo);

        // Pide la cantidad vendida
        cout << "Ingrese la cantidad vendida: ";
        cin >> venta;

        // Actualiza el stock y acumula las ventas
        vecProductos[indiceCodigo].stockActual -= venta;
        vecProductos[indiceCodigo].acumVentasProducto += venta;
```

```
        // Vuelve a pedir el código del producto
        cout << "Ingrese el código del producto vendido (0 para
finalizar): ";
        cin >> codigo;
    }
}

/*
-Dado un VECTOR con los datos de los productos, y su LONGITUD, DEVOLVER
cual
seria el margen de ganancia total sobre el stock actual
*/
void ganancia(Productos vecProductos[], int dim, int& acum) {
    int i;
    for (i = 0; i < dim; i++) {
        acum += (vecProductos[i].precioVenta -
vecProductos[i].precioCompra) * vecProductos[i].acumVentasProducto;
    }
}

/*
-Dado un VECTOR, DEVOLVER UN INDICADOR si el stock actual del prodcuto
alcanza el stock minimo
*/
bool pedir(Productos unProducto) {
    return (unProducto.stockActual >= unProducto.stockMinimo);
}

int menorPrecioCompra(Productos vecProductos[], int dim) {
    int i, iMin = 0;
    for (i = 1; i < dim; i++) {
        if (vecProductos[i].precioCompra < vecProductos[iMin].precioVenta)
{
            iMin = i;
        }
    }
    return iMin;
}

int mayorPrecioCompra(Productos vecProductos[], int dim) {
    int i, iMax = 0;
    for (i = 1; i < dim; i++) {
        if (vecProductos[i].precioCompra > vecProductos[iMax].precioVenta)
{
            iMax = i;
        }
    }
    return iMax;
}
/*
-Dado un VECTOR con los datos de los productos, y su long, DEVOLVER el max
y min
precio de venta
*/
```

```
void minMax(Productos vecProductos[], int dim, int& menorPrecio, int&
mayorPrecio) {
    menorPrecio = vecProductos[menorPrecioCompra(vecProductos,
dim)].precioVenta;
    mayorPrecio = vecProductos[mayorPrecioCompra(vecProductos,
dim)].precioVenta;
}

bool stockPorcentaje(Productos unProducto, int porcentaje) {
    return (unProducto.stockActual >= (unProducto.stockMinimo * porcentaje
/ 100 + unProducto.stockMinimo));
}
/*
-Dado un vector con los datos de los productos, y su long, DEVOLVER OTRO
VECTOR con los productos, y su long
cuyo stock actual supere en 50% al stock minimo
*/
void sobreStock(Productos vecProductos[], int dim, Productos
vecProductosSobreStock[], int& dimSobreStock) {
    int i, x = 0;
    for (i = 0; i < dim; i++) {
        if (stockPorcentaje(vecProductos[i], 50)) {
            vecProductosSobreStock[x] = vecProductos[i];
            x++;
        }
    }
    dimSobreStock = x;
}

void imprimir(Productos vecProductos[], int dim) {
    int i;
    for (i = 0; i < dim; i++) {
        cout<<endl;
        cout << "__________PRODUCTO__________" << endl;
        cout << "CODIGO: " << vecProductos[i].codigo << endl;
        cout << "PRECIO COMPRA: " << vecProductos[i].precioCompra << endl;
        cout << "PRECIO VENTA: " << vecProductos[i].precioVenta << endl;
        cout << "STOCK INICIAL: " << vecProductos[i].stockInicial << endl;
        cout << "STOCK MINIMO: " << vecProductos[i].stockMinimo << endl;
        cout << "STOCK ACTUAL: " << vecProductos[i].stockActual << endl;
        cout << "CANTIDAD VENDIDA: " << vecProductos[i].acumVentasProducto
<< endl;
    }
}

int alcanzanStockMinimo(Productos vecProductos[], int dim) {
    int i, contProductosAlcanzanMinimo = 0;
    for (i = 0; i < dim; i++) {
        if (pedir(vecProductos[i])) {
            contProductosAlcanzanMinimo++;
        }
    }
    return contProductosAlcanzanMinimo;
}
```

```
int main() {
    int i, codigo, resp, venta, codProducto, dim, dimSobreStock,
acumGananciasMes = 0, mayorPrecioVenta, menorPrecioVenta;

    Productos vecProductos[DIM];
    Productos vecProductosSobreStock[DIM];

    /*
    PUNTO 1
    -De cada producto, sus datos
    */
    cargarProducto(vecProductos, dim);
    imprimir(vecProductos, dim);

    // CARGA DE VENTAS
    cargaVenta(vecProductos, dim);
    imprimir(vecProductos, dim);

    cout<<endl;
    cout<<"_________RESOLUCION_________"<<endl;
    /*
    PUNTO 2
    -El margen de ganancia total sobre el stock actual al finalizar el mes
    */
    ganancia(vecProductos, dim, acumGananciasMes);
    cout << "Ganancia del mes: " << acumGananciasMes << endl;

    /*
    PUNTO 3
    -Cantidad de productos que alcanzan el stock mínimo al finalizar el
mes
    */
    cout << "Cantidad de productos que alcanzan el stock mínimo: " <<
alcanzanStockMinimo(vecProductos, dim) << endl;

    /*
    PUNTO 4
    -Cual es el menor y mayor precio de compra
    */
    minMax(vecProductos, dim, menorPrecioVenta, mayorPrecioVenta);
    cout << "Menor precio de compra: " << menorPrecioVenta << endl;
    cout << "Mayor precio de compra: " << mayorPrecioVenta << endl;

    /*
    PUNTO 5
    -Cantidad de productos y datos del producto cuyo stock actual supere
el 50% al stock mínimo
    */
    cout << "Cantidad de productos y datos del producto cuyo stock actual
supere en un 50% el stock mínimo: " << endl;
    sobreStock(vecProductos, dim, vecProductosSobreStock, dimSobreStock);
    cout << "Cantidad: " << dimSobreStock << endl;
    imprimir(vecProductosSobreStock, dimSobreStock);
```

```
    getch();
    return 0;
}
```