

**Universidad de Guadalajara**

---

**Centro Universitario de Ciencias Exactas e Ingenierías**



**IL365 - Estructura de Datos -D01**

**Actividad de Aprendizaje #02**

**La Anidación Estructural: Registros con Arreglos, Arreglos de Registros y Arreglos de Objetos**

**Alumna: Cervantes Araujo Maria Dolores**

**Código: 217782452**

**Fecha de Entrega: 03 febrero de 2023**



Autoevaluación			
Concepto	Si	No	Acumulación
Bajé el trabajo de internet o alguien me lo pasó (aunque sea de forma parcial)	-100 pts	0 pts	0
Incluí el código fuente <b>en formato de texto (sólo si funciona cumpliendo todos los requerimientos)</b>	+25pts	0 pts	25
Incluí las <b>impresiones de pantalla (sólo si funciona cumpliendo todos los requerimientos)</b>	+25pts	0 pts	25
Incluí una <b>portada</b> que identifica mi trabajo (nombre, código, materia, fecha, título)	+25 pts	0 pts	25
Incluí una <b>descripción y conclusiones</b> de mi trabajo	+25 pts	0 pts	25
Suma:			100

## Introducción:

En esta ocasión el problema nos plantea la necesidad de crear un almacén o inventario para una tienda de abarrotes; para hacer posible el programa son necesarias las funciones para agregar nuevos productos, añadir existencia, eliminar existencia y visualizar inventario general, por lo que fue necesario capturar campos como código de barras, nombre del producto, fecha de entrada en stock, como otros atributos correspondientes al guardado exitoso del producto.

Para este programa se llevaron conocimientos de programación orientada a objetos con arreglos de registros y registros de arreglo, con una capacidad de hasta 500 productos, fue necesario revisar como realizar los métodos metiendo valores de objeto a otro objeto, así como demás detalles presentados a lo largo de la codificación del programa



## Código Fuente:

### [date.hpp](#)

```
#ifndef DATE_HPP_INCLUDED
#define DATE_HPP_INCLUDED

#include <string>

class Date {
    private:
        int year;
        int month;
        int day;
    public:
        Date();
        Date(const int&, int&, int&);

        int getYear() const;
        int getMonth() const;
        int getDay() const;

        void setYear(const int&);
        void setMonth(const int&);
        void setDay(const int&);

        std::string toString1() const;
};

#endif // DATE_HPP_INCLUDED
```



### date.cpp

```
#include "date.hpp"

Date::Date() {

}

Date::Date(const int& y, int& m, int& d) {
    year=y;
    month=m;
    day=d;
}

using namespace std;

int Date::getYear() const {
    return year;
}

int Date::getMonth() const {
    return month;
}

int Date::getDay() const {
    return day;
}

void Date::setYear(const int& y) {
    year=y;
}

void Date::setMonth(const int& m) {
    month=m;
}

void Date::setDay(const int& d) {
    day=d;
}

string Date::toString1() const {
    string temp;
    char out[10];

    sprintf(out, "%i", day);
    temp += out;
    temp += '/';
    sprintf(out, "%i", month);
    temp += out;
    temp += '/';
    sprintf(out, "%i", year);
```



```
temp += out;
```

```
return temp;
```

```
}
```

### product.hpp

```
#ifndef PRODUCT_HPP_INCLUDED  
#define PRODUCT_HPP_INCLUDED
```

```
#include <string>  
#include <iomanip>  
#include "date.hpp"
```

```
class Product {  
    private:  
        std::string barras, name;  
        int exisTotal;  
        float peso, precioMay, precioMenudeo;  
        Date dateInt;  
  
    public:  
        Product();  
        Product(const Product&);  
        int max_array;  
  
        std::string getBarras() const;  
        std::string getName() const;  
        int getExisTotal() const;  
        float getPeso() const;  
        float getPrecioMay() const;  
        float getPrecioMenudeo() const;  
        Date getDateInt() const;  
  
        void setBarras(const std::string&);  
        void setName(const std::string&);  
        void setExisTotal(const int&);  
        void setPeso(const float&);  
        void setPrecioMay(const float&);  
        void setPrecioMenudeo(const float&);  
        void setDateInt(const Date&);  
  
        Product& operator =(const Product&);  
        bool operator == (const Product&) const;  
        std::string toString() const;  
};
```

```
#endif // PRODUCT_HPP_INCLUDED
```



### product.cpp

```
#include <iostream>
#include "product.hpp"

using namespace std;

Product::Product() {

}

Product::Product(const Product& pd): barras(pd.barras), name(pd.name),
exisTotal(pd.exisTotal), peso(pd.peso),
precioMay(pd.precioMay), precioMenudeo(pd.precioMenudeo), dateInt(pd.dateInt),
max_array(pd.max_array) {}

Product& Product::operator=(const Product& pd) {
    barras=pd.barras;
    name=pd.name;
    exisTotal=pd.exisTotal;
    peso=pd.peso;
    precioMay=pd.precioMay;
    precioMenudeo=pd.precioMenudeo;
    dateInt=pd.dateInt;
    max_array=0;

    return *this;
}

string Product::getBarras() const {
    return barras;
}

string Product::getName() const {
    return name;
}

int Product::getExisTotal() const {
    return exisTotal;
}

float Product::getPeso() const {
    return peso;
}

float Product::getPrecioMay() const {
    return precioMay;
}

float Product::getPrecioMenudeo() const {
    return precioMenudeo;
}
```



```
    }

    Date Product::getDateInt() const {
        return dateInt;
    }

    void Product::setBarras(const std::string& bar) {
        barras=bar;
    }

    void Product::setName(const string& nom) {
        name=nom;
    }

    void Product::setExisTotal(const int& exT) {
        exisTotal=exT;
    }

    void Product::setPeso(const float& pes) {
        peso=pes;
    }

    void Product::setPrecioMay(const float& preM) {
        precioMay = preM;
    }

    void Product::setPrecioMenudeo(const float& prMn) {
        precioMenudeo=prMn;
    }

    void Product::setDateInt(const Date& starD) {
        dateInt = starD;
    }

    bool Product::operator==(const Product& ptr) const {
        return getBarras() == ptr.getBarras();
    }

    string Product::toString() const {
        string temp;
        char cantidades[50];

        temp = barras;
        temp += "|";
        setw(15);
        temp += name;
        temp += "|";
        setw(15);
        sprintf(cantidades, "%d", exisTotal);
        temp += cantidades;
        temp += "|";
        setw(15);
        sprintf(cantidades, "%.2f", peso);
```



```
temp += cantidades;  
temp += "|";  
setw(15);  
sprintf(cantidades, "%.2f", precioMay);  
temp += cantidades;  
temp += "|";  
setw(15);  
sprintf(cantidades, "%.2f", precioMenudeo);  
temp += cantidades;  
temp += "|";  
setw(15);  
temp += dateInt.toString1();  
  
return temp+="\n";  
}
```

### inventario.hpp

```
#ifndef INVENTARIO_HPP_INCLUDED  
#define INVENTARIO_HPP_INCLUDED  
  
#include <iostream>  
#include <string>  
#include "product.hpp"  
  
class Inventario {  
private:  
    Product collec[500];  
    Date myDate;  
  
public:  
    Inventario();  
    Inventario(const Inventario&);  
  
    void addPtr();  
    void deletPtr();  
    void newPtr();  
    void showPtr();  
};  
  
#endif // INVENTARIO_HPP_INCLUDED
```





### inventario.cpp

```
#include "inventario.hpp"

using namespace std;

Inventario::Inventario() {}

void Inventario::newPtr() {
    int i=collec[0].max_array;

    cout<<"Codigo de Barras (13 digitos): ";
    string auxBarras;
    cin.ignore();
    getline(cin, auxBarras);
    collec[i].setBarras(auxBarras);

    cout<<"Nombre: ";
    string auxName;
    getline(cin, auxName);
    collec[i].setName(auxName);

    cout<<"Fecha de Entrada AA MM DD: ";
    int auxD, auxM, auxA;
    cin>>auxA;
    cin>>auxM;
    cin>>auxD;
    Date myDate(auxA, auxM, auxD);
    collec[i].setDateInt(myDate);

    cout<<"Existencia Actual: ";
    int auxEA;
    cin>>auxEA;
    collec[i].setExisTotal(auxEA);

    cout<<"Peso: ";
    float auxP;
    cin>>auxP;
    collec[i].setPeso(auxP);

    cout<<"Precio de Mayoreo: ";
    float auxPmy;
    cin>>auxPmy;
    collec[i].setPrecioMay(auxPmy);

    cout<<"Precio de Menudeo: ";
    float auxMn;
    cin>>auxMn;
    collec[i].setPrecioMenudeo(auxMn);

    collec[i+1].max_array=-1;
    collec[i].max_array=0;
```



```
    collec[0].max_array+=1;
}

void Inventario::addPtr() {
    int i, mod, sum;
    string cod;

    std::cout<<"Ingresa el codigo en barras del producto: ";
    std::cin>>cod;

    for(i=0; i<sizeof(collec); i++) {
        if(collec[i].max_array==1) {
            system("PAUSE");
            break;
        }
        else if(cod==collec[i].getBarras()) {
            cout << "Cantidad de productos a agregar: ";
            cin >> mod;
            sum=mod+collec[i].getExisTotal();
            collec[i].setExisTotal(sum);
            break;
        }
    }
}

void Inventario::deletPtr() {
    int i, rest, mod1;
    string cod;

    std::cout<<"Ingresa el codigo en barras del producto: ";
    std::cin>>cod;

    for(i=0; i<sizeof(collec); i++) {
        if(collec[i].max_array==1) {
            system("PAUSE");
            break;
        }
        else if(cod==collec[i].getBarras()) {
            cout<<"Cantidad de productos a eliminar: ";
            cin>>mod1;
            rest=collec[i].getExisTotal()-mod1;
            collec[i].setExisTotal(rest);
            break;
        }
    }
}

void Inventario::showPtr() {
    int i;

    for(i=0; i<sizeof(collec); i++) {
        ///ITERACIÃN QUE IMPIDE IMPRIMIR LOS 500 ESPACIOS DEL ARREGLO
        if(collec[i].max_array==1) {
```



```
        system("PAUSE");  
        break;  
    }  
    cout<<collec[i].toString();  
}  
}
```

### mainMenu.hpp

```
#ifndef MAINMENU_HPP_INCLUDED  
#define MAINMENU_HPP_INCLUDED  
  
#include "inventario.hpp"  
#include <iostream>  
  
class Menu {  
    private:  
        int opc;  
  
        Inventario selec;  
    public:  
  
        int getOpc() const;  
        void setOpc(const int&);  
        void visualizacion();  
  
};  
  
#endif // MAINMENU_HPP_INCLUDED
```

### mainMenu.cpp

```
#include "mainMenu.hpp"  
  
int Menu::getOpc() const {  
    return opc;  
}  
  
void Menu::setOpc(const int& nu) {  
    opc=nu;  
}  
  
void Menu::visualizacion() {  
    int x=0;  
    while(x==0) {  
        system("cls");  
  
        std::cout<<"ALMACENES LOLITA"<<std::endl<<std::endl;
```



```
std::cout<<"[1] Ingresar nuevo producto"<<std::endl;
std::cout<<"[2] Aumentar inventario del producto"<<std::endl;
std::cout<<"[3] Eliminar inventario del producto"<<std::endl;
std::cout<<"[4] Imprimir Inventario Total"<<std::endl;
std::cout<<"[5] Salir"<<std::endl;
std::cin>>opc;

switch(opc) {
    case 1:
        system("cls");
        selec.newPtr();
        getchar();
        break;
    case 2:
        system("PAUSE()");
        system("cls");
        selec.addPtr();
        break;
    case 3:
        system("PAUSE()");
        system("cls");
        selec.deletePtr();
        break;
    case 4:
        system("PAUSE()");
        system("cls");
        selec.showPtr();
        break;
    case 5:
        getchar();
        system("cls");
        std::cout<<"HASTA LA PROXIMA!! :D"<<std::endl;
        x+=1;
        break;
    default:
        std::cout<<"Ingresa solo valores que se te solicitan"<<std::endl;
        break;
}
}
```

### main.cpp

```
#include <iostream>
#include "mainMenu.hpp"
using namespace std;

int main() {
    Menu start;
    start.visualizacion();
    return 0;
}
```



## Impresiones de Pantalla:

```
"C:\Users\cerva\Escritorio\F.Prog\POO C++\actividad2\bin\Debug\actividad2.exe"
ALMACENES LOLITA

[1] Ingresar nuevo producto
[2] Aumentar inventario del producto
[3] Eliminar inventario del producto
[4] Imprimir Inventario Total
[5] Salir
```

### *Agregar productos*

```
"C:\Users\cerva\Escritorio\F.Prog\POO C++\actividad2\bin\Debug\actividad2.exe"
Codigo de Barras (13 digitos): 9876054389234
Nombre: Shampoo
Fecha de Entrada AA MM DD: 2022 04 21
Existencia Actual: 15
Peso: 15.1
Precio de Mayoreo: 25.5
Precio de Menudeo: 30_
```

### *Inventario*

```
"C:\Users\cerva\Escritorio\F.Prog\POO C++\actividad2\bin\Debug\actividad2.exe"
9876054389234|Shampoo|15|15.10|25.50|30.00|21/4/2022
1234567890321|Nescafe|79|20.00|17.50|28.00|18/5/2021
6574893201945|Maruchan|40|11.00|10.00|14.50|2/3/2023
6589234156209|Refresco|1|12.20|12.70|21.50|30/1/2023
1234567865346|Marcadores|30|23.00|20.50|33.60|17/9/2022
Presione una tecla para continuar . . .
```

### *Ingresar/Eliminar cantidades de stock*

```
"C:\Users\cerva\Escritorio\F.Prog\POO C++\actividad2\bin\Debug\actividad2.exe"
Ingresa el codigo en barras del producto: 1234567890321
Cantidad de productos a agregar: 35_
```



```
Selecccionar "C:\Users\cerva\Escritorio\F.Prog\POO C++\actividad2\bin\Debug\actividad2.exe"
9876054389234|Shampoo|15|15.10|25.50|30.00|21/4/2022
1234567890321|Nescafe|114|20.00|17.50|28.00|18/5/2021
6574893201945|Maruchan|40|11.00|10.00|14.50|2/3/2023
6589234156209|Refresco|1|12.20|12.70|21.50|30/1/2023
1234567865346|Marcadores|30|23.00|20.50|33.60|17/9/2022
Presione una tecla para continuar . . .
```

```
"C:\Users\cerva\Escritorio\F.Prog\POO C++\actividad2\bin\Debug\actividad2.exe"
Ingresa el codigo en barras del producto: 6574893201945
Cantidad de productos a eliminar: 40_
```

```
Selecccionar "C:\Users\cerva\Escritorio\F.Prog\POO C++\actividad2\bin\Debug\actividad2.exe"
9876054389234|Shampoo|15|15.10|25.50|30.00|21/4/2022
1234567890321|Nescafe|114|20.00|17.50|28.00|18/5/2021
6574893201945|Maruchan|0|11.00|10.00|14.50|2/3/2023
6589234156209|Refresco|1|12.20|12.70|21.50|30/1/2023
1234567865346|Marcadores|30|23.00|20.50|33.60|17/9/2022
Presione una tecla para continuar . . .
```

*\*Si ingresa un código erróneo, lo devuelve al menú*

```
"C:\Users\cerva\Escritorio\F.Prog\POO C++\actividad2\bin\Debug\actividad2.exe"
Ingresa el codigo en barras del producto: 2435162738390
Presione una tecla para continuar . . .
```

```
"C:\Users\cerva\Escritorio\F.Prog\POO C++\actividad2\bin\Debug\actividad2.exe"
HASTA LA PROXIMA!! :D
Process returned 0 (0x0)   execution time : 1452.980 s
Press any key to continue.
_
```



## Resumen Personal

En lo personal, este programa me costó un poquito más de esfuerzo al momento de codificar los métodos porque no había trabajado antes con arreglos orientados a objetos, por lo que me llegó a dar diversos errores en especial al momento de añadir nuevos productos, lo cual se solucionó poniendo una limitante para nuestro arreglo *“collec”* y que al momento de iterar los productos muestre según la misma limitante que se guardó anteriormente, es decir que se almacenaba cuantos registros se iban agregando para después iterarlos en *for* según el último valor que registro.

En esta ocasión como el maestro nos aclaró que hay opciones en las que el usuario no debe de interactuar con listas o en este caso, arreglos; para las funciones de agregar y eliminar stock al momento de no encontrar el producto por código de barras, simplemente lo saca de la función retornando al usuario al menú para que escoja otra función, consulte el inventario total o en su defecto que cierre la ejecución del programa.

Puedo concluir que aprendí una nueva manera de trabajar con arreglos y que además no estaba tan familiarizada con la misma, siendo una actividad entretenida, estresante a veces por los fallos, pero enriquecedora e interesante para futuras actividades en programación.