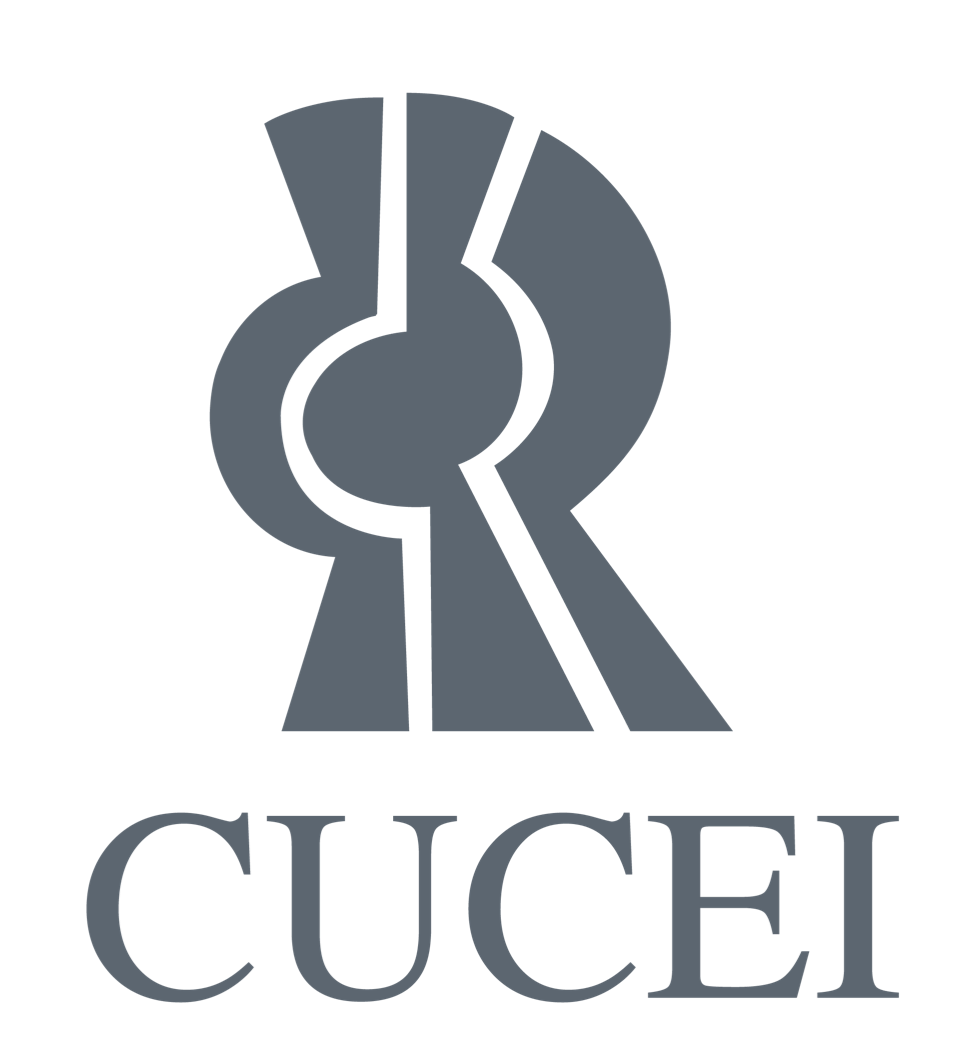
1-2-2019

**david gutierrez alvarez**

**Estructura de datos I**

****

**LA ANIDACION ESTRUCTURAL:**

**REGISTROS CON ARREGLOS, ARREGLOS DE REGISTROS Y ARREGLOS DE OBJETOS**

|  |
| --- |
| **RESUMEN PERSONAL Y FORMA DE ABORDAR EL PROBLEMA** |

Primero empecé a ver los requisitos, el echo de hacer un inventario me dijo directamente que tenía que trabajar con arreglos, para empezar cree las 4 clases que me pide, fecha, producto, colección de productos y menú, la fecha la hice automática para que generara la fecha exacta de cuando se crea cada producto año/mes/día/hora/minuto/segundo, esto tubo un poco de complejidad ya que al llamar la librería #include <ctime> y generar la fecha tenía que utilizar código repetitivo, esto lo soluciones creando una función de tipo “tm” que era un registro, del producto lo hice en una estructura que esta dentro de una clase, esto no tuvo complejidad alta, el punto de validar cada acción fue lo que le trajo la mayor complejidad.

Esta actividad fue muy agradable, tuve muchos errores que tarde horas en solucionar y cuando veía en que fallaba, me daba cuenta que era fácil de solucionarlo, pero el echo de estar practicando me a ayudado a completar el trabajo.

Espero que sigua el ritmo de la clase.

|  |
| --- |
| **Main.cpp** |
| **#include <iostream>**  **#include <menu.h>**  ***using* *namespace* std;**  **int main() {**  **Menu principal;**  **principal.menu();**  ***return* 0;**  **}** |

|  |
| --- |
| **Menu.h** |
| **#ifndef MENU\_H**  **#define MENU\_H**  **#include <inventario.h>**  **#include <iostream>**  ***class* Menu {**  **Inventario inventario;**  ***public*:**  ***enum* Options {**  ***OptionCreate* = 1,**  ***OptionAdd*,**  ***OptionRemov*,**  ***OptionSearch*,**  ***OptionOut***  **};**  **void menu();**  **};**  **#endif *//* *MENU\_H*** |

|  |
| --- |
| **Menu.cpp** |
| **#include "menu.h"**  ***using* *namespace* std;**  **void Menu::menu() {**  **int option = 0;**  **string code;**  **cout << "\t\t\t.:BIENVENIDO:." << endl;**  ***do* {**  **cout << *OptionCreate* << ".- Agregar producto " << endl**  **<< *OptionAdd* << ".- Actualizar existencia de producto " << endl**  **<< *OptionRemov* << ".- Vender producto " << endl**  **<< *OptionSearch* << ".- Mostrar productos " << endl**  **<< *OptionOut* << " .- Salir"<<endl**  **<<"Elige una opcion: ";**  **cin >> option;**  **cin.ignore();**  ***switch* (option) {**  ***case* *OptionCreate*:**  **inventario.addProduct();**  ***break*;**  ***case* *OptionAdd*:**  **cout << "introduce el codigo para agregar elementos del producto: ";**  **cin >> code;**  **inventario.Add(code);**  ***break*;**  ***case* *OptionRemov*:**  **cout << "introduce el codigo para retirar elementos del producto: ";**  **cin >> code;**  **inventario.Remove(code);**  ***break*;**  ***case* *OptionSearch*:**  **inventario.print();**  ***break*;**  ***case* *OptionOut*:**  ***break*;**  ***default*:**  **cout<<"Dato no valido "<<endl;**  ***break*;**  **}**  **} *while*(option != *OptionOut*);**  **}** |

|  |
| --- |
| **fecha.c** |
| **#ifndef FECHA\_H**  **#define FECHA\_H**  ***class* Fecha {**  ***private*:**  **int year;**  **int month;**  **int day;**  **int hour;**  **int minute;**  **int second;**  ***public*:**  **int getYear() *const*;**  **void setYear();**  **int getMonth() *const*;**  **void setMonth();**  **int getDay() *const*;**  **void setDay();**  **int getHour() *const*;**  **void setHour();**  **int getMinute() *const*;**  **void setMinute();**  **int getSecond() *const*;**  **void setSecond();**  **void generate();**  **tm getGenerator();**  **};**  **#endif *//* *FECHA\_H*** |

|  |
| --- |
| **Fecha.cpp** |
| **#include "fecha.h"**  **tm Fecha::getGenerator(){**  **time\_t tiempoReal = time(*nullptr*);*/\*esto* *guarda* *el* *tiempo* *actual\*/***  ***struct* tm today = \*localtime(&tiempoReal);**  ***return* today;**  **}**  **void Fecha::setYear() {**  **year = getGenerator().tm\_year + 1900;**  **}*/\*ya* *que* *empieza* *en* *1900\*/***  **void Fecha::setMonth() {**  **month = getGenerator().tm\_mon + 1;**  **}*/\*empieza* *a* *contar* *desde* *el* *0\*/***  **void Fecha::setDay(){**  **day = getGenerator().tm\_mday;**  **}**  **void Fecha::setHour() {**  **hour = getGenerator().tm\_hour;**  **}**  **void Fecha::setMinute() {**  **minute = getGenerator().tm\_min;**  **}**  **void Fecha::setSecond() {**  **second = getGenerator().tm\_sec;**  **}**  **void Fecha::generate() {*/\*esto* *inicializa* *la* *fecha\*/***  **setYear();**  **setMonth();**  **setDay();**  **setHour();**  **setMinute();**  **setSecond();**  **}**  **int Fecha::getYear() *const* {**  ***return* year;**  **}**  **int Fecha::getMonth() *const* {**  ***return* month;**  **}**  **int Fecha::getDay() *const* {**  ***return* day;**  **}**  **int Fecha::getHour() *const* {**  ***return* hour;**  **}**  **int Fecha::getMinute() *const* {**  ***return* minute;**  **}**  **int Fecha::getSecond() *const* {**  ***return* second;**  **}** |

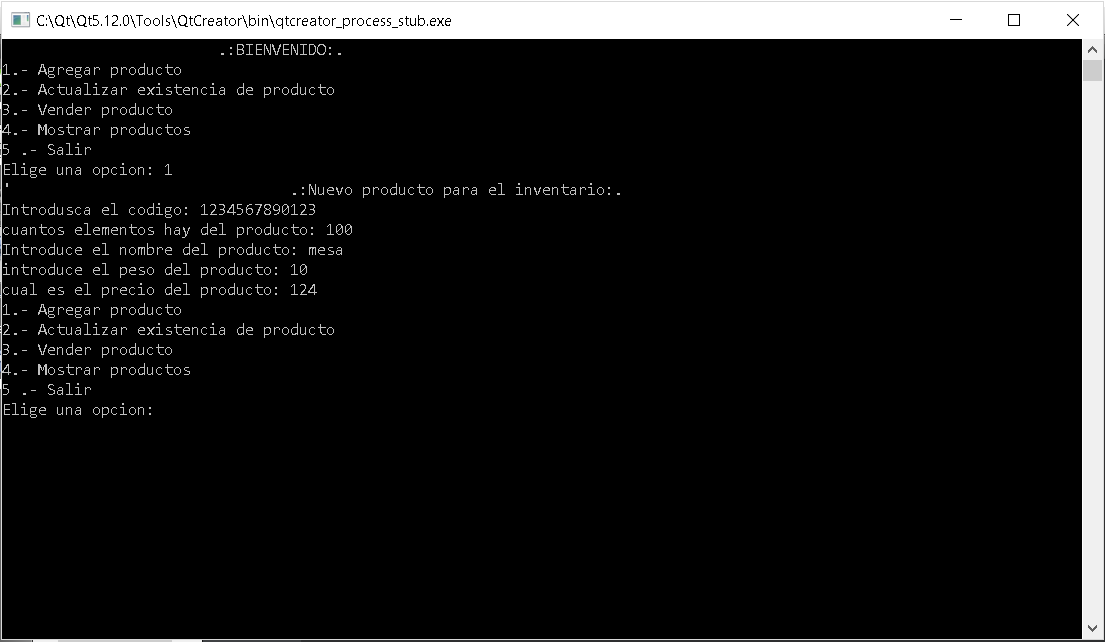
|  |
| --- |
| **Producto.h** |
| **#ifndef PRODUCT\_H**  **#define PRODUCT\_H**  **#include <fecha.h>**  **#include <iostream>**  ***class* Product {**  ***private*:**  ***struct* Data {**  **std::string barCode;*/\*Codigo* *de* *barras* *de* *13* *digitos\*/***  **std::string name;*/\*Nombre\** *del* *producto\*/***  **float weight;*/\*Peso\*/***  **Fecha entryDate; */\*Fecha* *de* *entrada\*/***  **float price;*/\*Precio\*/***  **int existence; */\*Existencia* *actual\*/***  **}DataBase;**  ***public*:**  **Product();**  **void Date();**  **std::string getBarCode() *const*;**  **void setBarCode(std::string &);**  **std::string getName() *const*;**  **void setName(*const* std::string &);**  **float getWeight() *const*;**  **void setWeight(float);**  **void setPrice(float);**  **float getWholesalePrice() *const*;*/\*Precio* *mayoreo\*/***  **float getRetailPrice() *const*; */\*Precio* *menudeo\*/***  **void showProducts() *const*;**  **int getExistence() *const*;*/\*ver* *cantidad* *de* *productos\*/***  **void add(int);*/\*añadir* *productos\*/***  **void remove(int);*/\*remueve* *productos\*/***  **bool validateBarCode(*const* std::string &);**  **};**  **#endif *//* *PRODUCT\_H*** |

|  |
| --- |
| **Producto.cpp** |
| **#include "product.h"**  ***using* *namespace* std;**  **void Product::Date() {**  **DataBase.entryDate.generate();**  **}**  **Product::Product() {**  **DataBase.existence = 0;**  **}**  **void Product::setBarCode(std::string &value) {**  ***if*(validateBarCode(value)) {**  **DataBase.barCode = value;**  **} *else* {**  **cout << "codigo invalido" << endl << endl;**  **}**  **}**  **void Product::setName(*const* std::string &value) {**  **DataBase.name = value;**  **}**  **string Product::getName() *const* {**  ***return* DataBase.name;**  **}**  **int Product::getExistence() *const* {**  ***return* DataBase.existence;**  **}**  **void Product::add(int value) {**  **DataBase.existence += value;**  **}**  **void Product::remove(int value){**  **DataBase.existence -= value;**  **}**  **string Product::getBarCode() *const* {**  ***return* DataBase.barCode;**  **}**  **float Product::getWeight() *const* {**  ***return* DataBase.weight;**  **}**  **void Product::setWeight(float value) {**  **DataBase.weight = value;**  **}**  **void Product::setPrice(float value) {**  **DataBase.price = value;**  **}**  **float Product::getWholesalePrice() *const* {**  ***return* DataBase.price\*0.85;**  **}**  **float Product::getRetailPrice() *const* {**  ***return* DataBase.price;**  **}**  **bool Product::validateBarCode(*const* std::string &value) {**  ***/\*con* *esto* *se* *obtiene* *el* *codigo* *de* *barras* *de* *13* *dijitos\*/***  ***if*(value.size() == 13) {**  ***/\*Con* *esto* *rectifica* *que* *sean* *solo* *digitos\*/***  ***for* (int i = 0; i < value.size(); i++) {**  ***if*(value[i] < 48 *or* value[i] > 57) {**  ***return* *false*;**  **}**  **}**  **DataBase.barCode = value;**  **} *else* {**  ***return* *false*;**  **}**  ***/\*si* *cumple* *con* *todas* *las* *condiciones* *saldra* *y* *retornara* *verdadero\*/***  ***return* *true*;**  **}**  **void Product::showProducts() *const* {**  **cout << "\t\t\tCodigo de barras: " << getBarCode() << endl**  **<< "\t\t\tNombre: " << getName() << endl**  **<< "\t\t\tPeso del producto: " << getWeight() <<endl**  **<< "\t\t\tFecha: " << DataBase.entryDate.getYear() << "/"**  **<< DataBase.entryDate.getMonth() << "/"**  **<< DataBase.entryDate.getDay() << " "**  **<< DataBase.entryDate.getHour() << ":"**  **<< DataBase.entryDate.getMinute() << ":"**  **<< DataBase.entryDate.getSecond() << endl**  **<< "\t\t\tPrecio por mayoreo: " << getWholesalePrice() << endl**  **<< "\t\t\tPrecio por menudeo: " << getRetailPrice() << endl**  **<< "\t\t\tExistenncia del producto: " << getExistence() << endl;**  **}** |

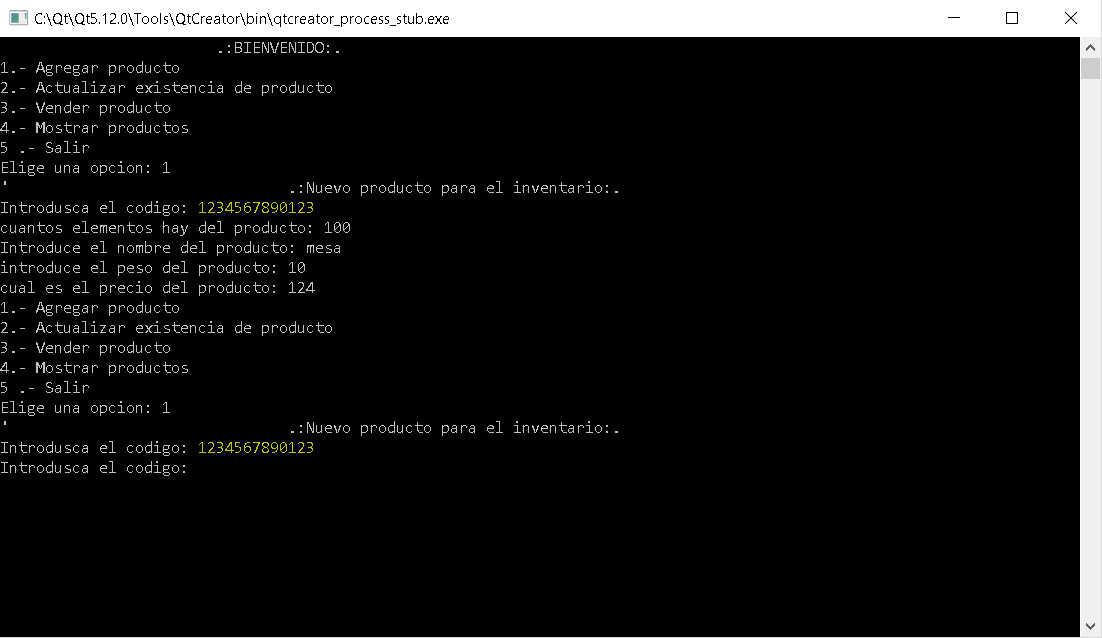
|  |
| --- |
| **Inventario.h** |
| **#ifndef INVENTARIO\_H**  **#define INVENTARIO\_H**  **#include <product.h>**  **#define FULL 500**  ***class* Inventario {**  ***private*:**  **Product products[FULL];**  **int last;**  **int positionCode;*/\*guarda* *la* *posicion* *del* *codigo* *buscado\*/***  ***public*:**  **Inventario();**  **bool isFull();*/\*revisa* *si* *el* *inventario* *esta* *lleno\*/***  **void addProduct();*/\*crea* *un* *nuevo* *producto\*/***  **void print();*/\*imprime* *en* *pantalla* *la* *lista* *de* *productos\*/***  **void Add(*const* std::string);*/\*añade* *cantidad* *a* *los* *productos\*/***  **bool searchCode(*const* std::string);*/\*verifica* *que* *el* *codigo* *introducido* *exista\*/***  **void Remove(*const* std::string);*/\*retira* *elementos* *de* *un* *producto\*/***  **};**  **#endif *//* *INVENTARIO\_H*** |

|  |
| --- |
| **Inventario.cpp** |
| **#include "inventario.h"**  ***using* *namespace* std;**  **Inventario::Inventario() : last(0) { }**  **bool Inventario::isFull() {**  ***return* last == FULL;**  ***/\*last* *es* *la* *cantidad* *de* *elementos* *que* *hay* *menos* *uno* *ya* *que***  ***inicia* *desde* *cero\*/***  **}**  **void Inventario::addProduct() {**  **string code, name;**  **float weight, price;**  **int existence;**  ***if*(isFull()) {**  **cout << "el inventario esta lleno" << endl;**  **} *else* {**  **cout << "'\t\t\t\t.:Nuevo producto para el inventario:." << endl;**  ***do* {**  **cout << "Introdusca el codigo: ";**  **cin >> code;**  ***/\*añade* *el* *codigo* *al* *final\*/***  **products[last].setBarCode(*code*);**  **} *while*(!products[last].validateBarCode(code) *or* searchCode(code));**  ***/\*verifica* *que* *tenga* *13* *digitos,* *y* *que* *ese* *codigo* *no* *exista***  ***para* *añadirlo* *y* *continuar\*/***  ***do* {**  **cout << "cuantos elementos hay del producto: ";**  **cin >> existence;**  **products[last].add(existence);**  **} *while*(existence <= 0);**  ***/\*verifica* *que* *se* *este* *añadiendo* *por* *lo* *menos* *un* *elemento\*/***  ***do*{**  **cout << "Introduce el nombre del producto: ";**  **cin >> name;**  **products[last].setName(name);**  **} *while*(name.size() < 1);**  ***/\*verifica* *que* *almenos* *tenga* *1* *caracteres* *el* *nombre* *del* *producto\*/***  ***do*{**  **cout << "introduce el peso del producto: ";**  **cin >> weight;**  **products[last].setWeight(weight);**  **} *while* (weight <= 0);**  ***/\*verifica* *que* *el* *peso* *sea* *positivo\*/***  ***do*{**  **cout << "cual es el precio del producto: ";**  **cin >> price;**  **products[last].setPrice(price);**  **} *while* (price <= 0);**  ***/\*verifica* *que* *el* *precio* *sea* *positivo\*/***  **Products[last].Date();*/\*genera* *la* *fecha* *de* *creacion\*/***  **last++;**  **}**  **}**  **void Inventario::print() {**  **cout << endl << endl << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;**  ***for* (int i = 0; i < last; i++) {**  **cout << "\t\t\t\t\t\tProducto " << i + 1 << endl << endl;**  **products[i].showProducts();**  **cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl << endl;**  **}**  **}**  **void Inventario::Add(*const* string code) {**  **int existence;**  ***if*(searchCode(code) and last != 0) {**  **cout << "el articulo existe, ";**  ***do*{**  **cout << "cuantos elementos desea agregar al producto: ";**  **cin >> existence;**  **cout << endl;**  ***/\*aqui* *utlizamos* *la* *posicion* *que* *se* *guarda* *al* *buscar* *el* *codigo***  ***para* *modificar* *la* *cantidad* *de* *elementos* *del* *producto\*/***  **} *while*(existence <= 0);**  **products[positionCode].add(existence);**  ***/\*y* *verifica* *que* *sea* *una* *cantidad* *positiva\*/***  **} *else* {**  **cout << endl << endl << "el articulo no existe ";**  **cout << "o el inventario esta vacio" << endl << endl;**  **}**  **}**  **bool Inventario::searchCode(*const* string code) {**  ***/\*verifica* *si* *existe* *el* *codigo\*/***  ***for* (int i = 0;i < last; i++) {**  **products[i].getBarCode();**  ***if*(products[i].getBarCode() == code) {**  **positionCode = i;**  ***/\*si* *existe,* *guarda* *la* *posicion* *en* *la* *que* *esta* *ese* *codigo\*/***  ***return* *true*;**  **}**  **}**  ***return* *false*;**  **}**  **void Inventario::Remove(*const* string code) {**  **int existence;**  ***if*(searchCode(code) *and* last != 0) {**  **cout << "el articulo existe, ";**  ***do*{**  **cout << "cuantos elementos desea quitar del inventario: ";**  **cin >> existence;**  **cout << endl;**  **} *while*(existence <= 0);**  **products[positionCode].remove(-1\*(existence));**  ***/\*esto* *verifica* *que* *no* *se* *esten* *añadiendo* *elementos\*/***  **} *else* {**  **cout << endl << endl << "el articulo no existe ";**  **cout << "o el inventario esta vacio" << endl << endl;**  **}**  **}** |

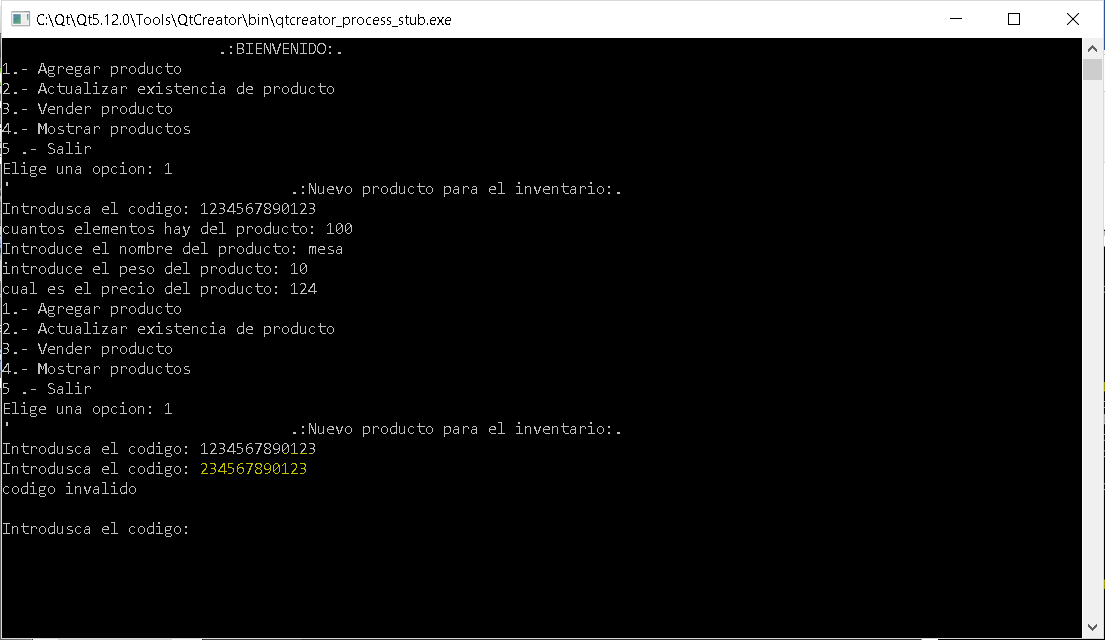
|  |
| --- |
| **CAPTURAS DE PANTALLA** |



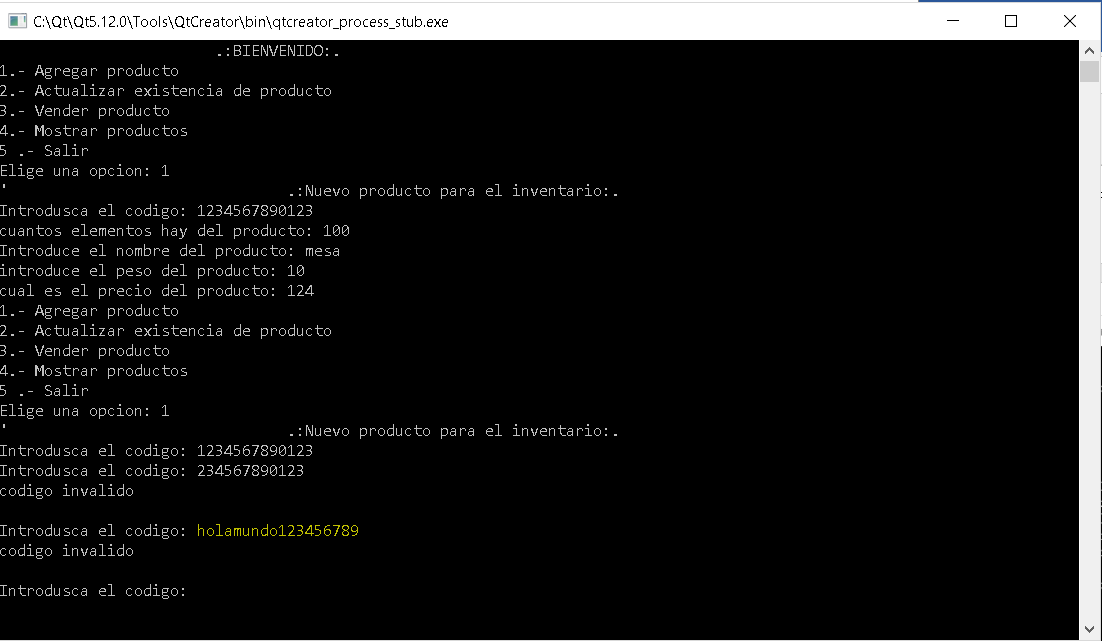
Se agrega un nuevo producto



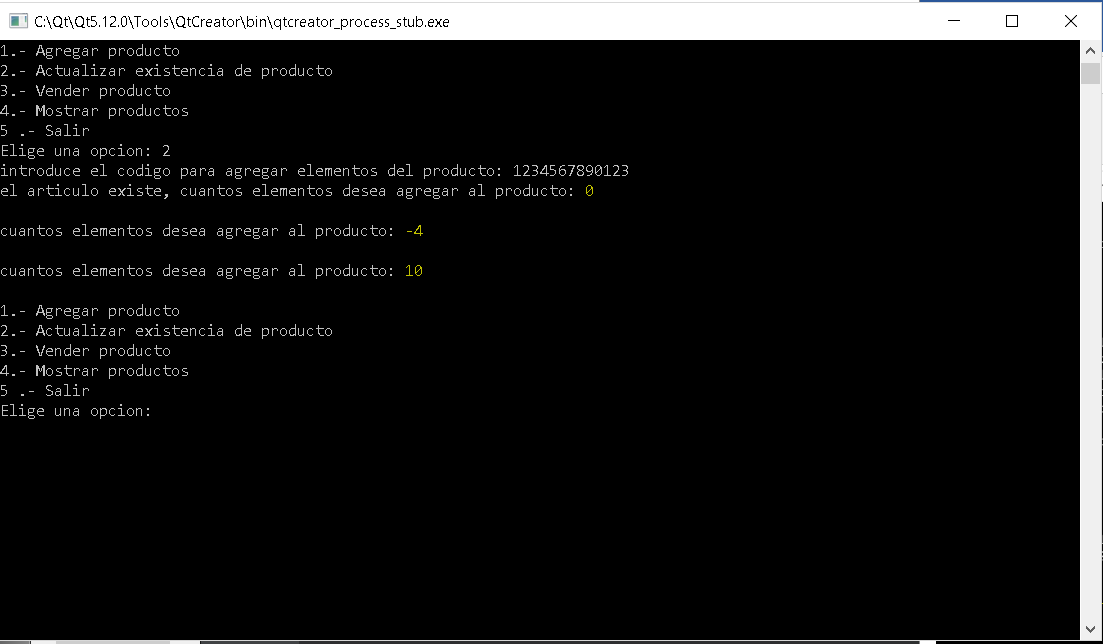
Validado que el código no se repita



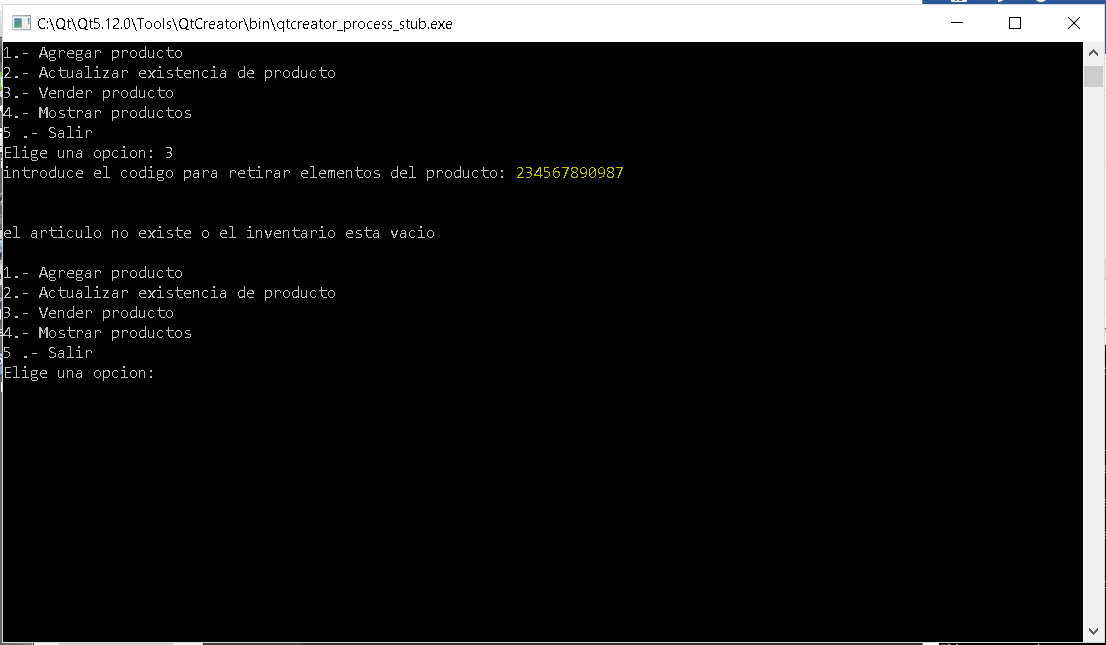
Validado que el código sea exacto a 13 digitos



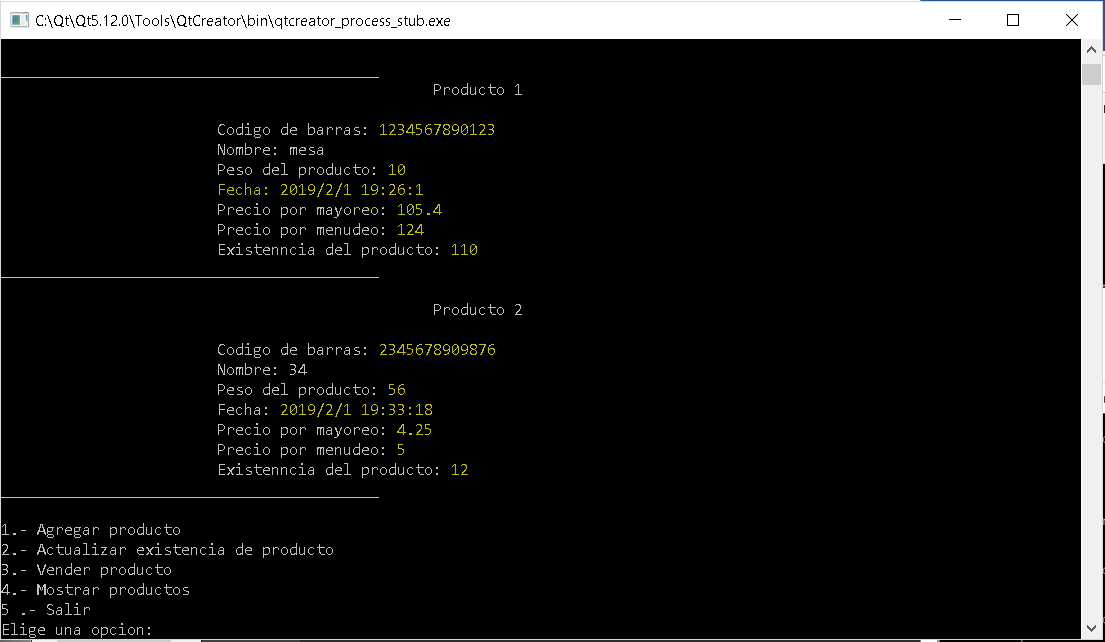
Validado que contenga solo dígitos



Validado que al añadir artículos no se coloque cero o valores negativos



Validado que no entre a codigos invalidos



Los productos se muestran con toda su informacion