# DESAFIOS FINAL PGE

**CLASE LISTADO DESAFIO NUMERO 1**

1. CREAR ARCHIVOS

* listado.h
* main.cpp

1. Listado.pro

QT += widgets

1. listado.h

template <class UNDEFINED> class Listado

private:

int cantidad

int libre

UNDEFINED \* v

public:

Listado(int cantidad = 20) : cantidad(cantidad),libre(0),v(new UNDEFINED [cantidad])

UNDEFINED get(int i)

Return v[i]

Int size ()

Return libre

Int getCantidad ()

Return cantidad

Bool add (UNDEFINED nuevo)

Template <class UNDEFINED> bool Listado<UNDEFINED>::add(UNDEFINED nuevo)

If (libre < cantidad)

V[libre] = nuevo

Libre++

Return true

Return false

1. main.cpp

#include “listado.h” <QApplication> <QDebug>

Int main

QApplication a(argc, argv)

Listado<int> \*l1 =new Listado<int>

L1.add (numero) //REPETIR 4 VECES MINIMO

For(int i =0; i < li.size();i++

qDebug() << l1.get(i)

**CLASE GENERICA MIVECTOR**

Empty qmake proyect MiVector

.pro

QT +=widgets

mivector.h

#include <QVector>

template <class UNDEFINED> class MiVector : private QVector<UNDEFINED>{

public:

MiVector() :QVector<UNDEFINED> () {

UNDEFINED get(int i) {

return this->at(i)

void add (UNDEFINED nuevo) {

this->push\_back(nuevo)

int size() {

return QVector<UNDEFINED>::size()

main.cpp

#include “mivector.h”

#include <QApplication>

#include <QDebug>

int main ( int argc, char \*\*argv){

QApplication a (arg,argv)

MiVector <int> \* v1 = new MiVector<int>

V1-> add (numero) //repetir varias veces

For(int i = 0; i < v1-> size() ; i++){

qDebug() << v1-> get(i)

return 0;

**CLASE LISTADO ENTEROS**

Empty qmake proyect ListadoEnteros

.pro

QT +=widgets

listadoenteros.h

#include <QVector>

class ListadoEnteros : public QVector<int>{

public:

ListadoEnteros() : QVector<int>(){}

main.cpp

#include “listadoenteros.h”

#include <QApplication>

#include <QDebug>

int main ( int argc, char \*\*argv){

QApplication a (arg,argv)

ListadoEnteros l1;

L1.push\_back(1); ///poner distintos entero

For(int i = 0; i < l1. size() ; i++){

qDebug() <<l1.at(i)

**SOBRECARGA DE OPERADORES**

DESCARGAR DESAFIO NUMERO 2

mivector.h

void operator+( UNDEFINED nuevo) {

this->add(nuevo);

void operator\*(int cuantos){

while(cuantos > 0)

this->add(this->last())

cuantos--;

main.cpp

eliminar todo menos el qapplication

MiVector<int>v1;

v1 + 9 // repetir con distintos números

v1\* 6

corregir el ciclo for en la parte de v1-> size() por v1.size()

**CLASE LINEA DE TEXTO**

Qt widget application LineaDeTexto

QWidget llamado ventana

lineadetexto.h

#include <QLineEdit>

Class LineaDeTexto : public QLineEdit{

Q\_OBJECT

Public:

LineaDeTexto (QWidget \* parent = nullptr ); //refactor en cpp

LineaDeTexto(const LineaDeTexto & lineaDeTexto); //refactor en cpp

LineaDeTexto& operator= (const LineaDeTexto & lineaDeTexto);

Lineadetexto.cpp

#include”lineadetexto.h”

LineaDeTexto::LineaDeTexto(QWidget \* parent ) : QLineEdit(parent){

LineaDeTexto::LineaDeTexto(const LineaDeTexto & lineaDeTexto) : QLineEdit() {

this->setText(lineaDeTexto.text())

LineaDeTexto& LineaDeTexto::operator= (const LineaDeTexto & lineaDeTexto)

this->setText(lineaDeTexto.text())

return \*this;

Ventana.ui

Agregar dos Line Edit y un push button y cambiar nombres (ltUsuario, ltClave, pb)

Promocionar ambos line edit a línea de texto

Main.cpp

#include “lineadetexto.h”

#include <QApplication>

#include <QVector>

LineaDeTexto l1;

QVector< LineaDeTexto> v;

v.push\_back(l1);

CLASE LINEA DE TEXTO CON RESTRICCIONES

Descargar desafio numero 5

lineadetexto.h

public:

enum Tipo { SIN\_RESTRICCION , SOLO\_NUMEROS }

void setTipo( Tipo tipo); //HACER REFACTOR

private:

Tipo tipo

Protected:

bool eventFilter (QObject \*, QEvent \* e) // REFACTOR EN lineadetexto.cpp

lineadetexto.cpp

bool LineaDeTexto::eventFilter(QObject \* , QEvent \* e)

if ( e-> type() == QEven::KeyPress && tipo == SOLO\_NUMEROS)

QKeyEvent \* keyEvent = static\_cast <QKeyEvent \* > €

Switch( keyEvent-> key()){

Case Qt::Key\_0: /// REPETIR PARA CADA NUMERO DEL 0 AL 9

Return false;

Default:

Return true;

Return false

**LineaDeTexto::LineaDeTexto ( QWidget \* parent ) : QLineEdit(parent)** ,tipo(SIN\_RESTRICCION){ **// YA EXISTE**

this->installEventFilter(this)

void LineaDeTexto::setTipo(Tipo tipo){

this->tipo = tipo;

ventana.cpp

ui->ltClave->setTipo(LineaDeTexto::SOLO\_NUMEROS);

LOGIN COMO SINGLETON

QT widget Application

Herede de QWidget con nombre Login

Login.h

Private:

Pasar a privado la línea Login(QWidget \*parent = nullptr);

Static Login \* instancia;

Public:

Static Login \* getInstancia(); //REFACTOR EN CPP

Login.cpp

Login\* Login::instancia = nullptr;

Login \* Login::getInstancia(){

If(instancia == nullptr){

Instancia = new Login

Return instancia

Login.ui

Crear interfaz con dos line edit y un pb y agregar un spacer

No olvidarse colocar los nombres

Modificar como psw el line edit

Main.cpp

Borrar línea con error

Login::getInstancia()->show()

PERSONAS COMPARABLES

Empty proyect Personas Comparables

.pro

QT += widgets

Persona.h

#include <QString>

Class Persona {

Private:

Int edad; //GETTER AND SETTER

QString nombre; //GETTER AND SETTER

Public:

Explicit Persona ( QString nombre = “ Sin Nombre” , int edad = 0): nombre ( nombre), edad(edad){}

Bool operator<(Persona otra){

If(this->edad < otra.edad)

Return true;

Return false;

Bool operator>(Persona otra){

If(this->edad > otra.edad)

Return true;

Return false;

Main.cpp

#include “persona.h”

#include <QApplication>

#include <QDebug>

Int main( int argc, char \*\* argv){

QApplication a(argc, argv);

Persona manu(“Manu”,21);

Persona cesar(“Cesar”,35);

If ( carlos > Miguel) {

qDebug() << “Carlos es Mayor que Miguel”;

}else if( Carlos < miguel) {

qDebug() << “Carlos es menor que Miguel”;

**Punteros a funciones**

Aplicación qt widgets aplication llamada PunterosFunciones

Hereda d QWidget y se llama login el qwidgt

Login.ui

Realizar estructura de login normal con dos line edit un pb y horizontal spacer

Cambiar nombres de cada uno (leusuario, leclave, y pb)

Echo mode ( password)

Quitar foco de boton en tab order

Login.h

Private:

QWidget \* ventana;

Private slots:

Void slot\_validar //REFACTOR

Login.cpp

AGREGAR EN CONSTRUCTOR ventana(new QWidget)

Dentro de constructor

void (QPushButton:: \*puntero\_a\_pressed)( );

void (Login:: \*puntero\_a\_slot\_validar)( );

puntero\_a\_pressed = &QPushButton::pressed;

puntero\_a\_slot\_validar = &Login::slot\_validar;

connect (ui->pb, puntero\_a\_pressed, this, puntero\_a\_slot\_validar);

Void Login::slot\_validar()

If(ui->leUsuario->text() == “admin” && ui->leClave->text() == “1234”

Ventana->show();

This->hide();

Else

This->close();

Excepciones

DESCARGAR CLASE LISTADO DEL DESAFIO 1

EXCEPCIONES.H

#include <QString>

Class ExcFueraRango{

Private:

QString mensaje; //GETTER AND SETTER

Int index; //GETTER AND SETTER

Public:

ExcFueraRango(QString mensaje, int index): mensaje(mensaje),index,index{}

LISTADO.H

#include “excepciones.h”

T get (int i ) {

If(i >= libre) {

ExcFueraRango excepción(“Intento de acceso fuera de rango”,i)

throw excepcion;

MAIN.CPP

Try{

qDebug() << l1->get(5)

qDebug() << l1->get(95)

catch( ExcFueraRango e) {

qDebug()<< e.getMensaje();