Introducción al Desarrollo de Aplicaciones con C++ y Qt4

Gastón Jorquera
gjorquera [at] gmail [dot] com

Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas

Viernes 13 de Marzo de 2009

Contenidos

Definiciones de Clases

```
#ifndef POINT2I_H
#define POINT2I_H

class Point2I
{
public:
    //! Constructor por defecto.
    Point2I();
    //! Constructor con valores.
    Point2I(int x, int y);
    //! Distancia euclidiana.
    friend double distancia(Point2I *fin);
    //! Calcula la norma del "vector"
    double norma();
    // Getters y setters.
    void setX(int x) { m_x = x; }
    void setY(int y) { m_y = y; }
    int x() { return m_y; }
    int y() { return m_x; }
private:
    int m_x = 0; // <- NO COMPILA
    int m_y;
}; // <- ojo el ";".

#endif // POINT2I_H
```

```
#include "point2i.h"

Point2I::Point2I()
        : m_x(0), m_y(0) {
}
Point2I::Point2I(int x, int y) {
    m_x = x; m_y = y;
}
double Point2I::distancia(Point2I *fin) {
    double x = 1.0*(m_x - fin->m_x);
    double y = 1.0*(m_y - fin->m_y);
    return sqrt(x*x + y*y);
}
double Point2I::norma() {
    // Distancia euclidiana entre
    // el origen y este punto.
    Point2I *origen = new Point2I;
    double norma = distancia(origen);
    delete origen;
    return norma;
}
```

Templates

```
#ifndef POINT2_H
#define POINT2_H

template <typename T>
class Point2
{
public:
    Point2() { m_x = T(); m_y = T(); }
    Point2(T x, T y) { m_x = x; m_y = y; }
    //! Distancia euclidiana.
    friend double distancia(Point2<T> *fin);
    //! Calcula la norma del "vector"
    double norma();
    // Getters y setters.
    void setX(T x) { m_x = x; }
    void setY(T y) { m_y = y; }
    T x() { return m_x; }
    T y() { return m_y; }
private:
    T m_x;
    T m_y;
};
```

```
template <typename T>
double Point2<T>::distancia(Point2<T> *fin) {
    double x = 1.0*(m_x - fin->m_x);
    double y = 1.0*(m_y - fin->m_y);
    return sqrt(x*x + y*y);
}

template <typename T>
double Point2<T>::norma() {
    Point2<T> *origen = new Point2<T>;
    double norma = distancia(origen);
    delete origen;
    return norma;
}

#endif // POINT2_H
```

Interfaces

```
#ifndef FIGURA_H
#define FIGURA_H

class Point2I;

class Figura
{
public:
    virtual Figura(Point2I *centroide) {
        m_centroide = centroide;
    }
    virtual ~Figura() {
        delete m_centroide;
    }
    //! Calcula el perimetro.
    virtual double perimetro() = 0;
    //! Calcula el area.
    virtual double area() = 0;
protected:
    Point2I *m_centroide;
};

#endif // FIGURA_H
```

```
#ifndef CIRCULO_H
#define CIRCULO_H

#include "point2i.h"
#define PI 3.1415

class Circulo : public Figura
{
public:
    Circulo(Point2I *centroide, int radio)
        : Figura(centroide) {
        m_radio = radio;
    }
    double perimetro() {
        return 2*PI*m_radio;
    }
    double area() {
        return PI*m_radio*m_radio;
    }
private:
    int m_radio;
};

#endif // CIRCULO_H
```

Herencia Múltiple

```
class A
{
    void metodo() {
        // Hace algo.
    }
    void unico() {
        // Hace otra cosa.
    }
};

class B
{
    void metodo() {
        // Implementacion distinta a la
        // de la clase A.
    }
};
```

```
clase C
{
    void metodo() {
        A::metodo();
        B::metodo();
        unico(); // Llama a A.
    }
};
```

Punteros

```
Point2I p1; // <=> Point2D p1();
Point2I p2(1, 3);
Point2I *p3;
Point2I *p4;

p3 = new Point2I(1, 3);
p4 = new Point2I;

p1.setX(p3->y());

delete p3;
delete p4;

int j = 2;
int *k = &j;  // &j devuelve "un puntero a" j.

*k = 4; // ahora j = 4 tambien
```

Referencias

```
// En la clase Point2

// Original
double Point2I::distancia(Point2I *fin) {
    double x = 1.0*(m_x - fin->m_x);
    double y = 1.0*(m_y - fin->m_y);
    return sqrt(x*x + y*y);
}
// Nueva
// Evitamos punteros nulos y copias!
double Point2I::distancia(Point2I &fin) {
    double x = 1.0*(m_x - fin.m_x);
    double y = 1.0*(m_y - fin.m_y);
    return sqrt(x*x + y*y);
}

// Ahora puede ser implementado asi:
double Point2I::norma() {
    // Distancia euclidiana entre
    // el origen y este punto.
    Point2I origen;
    return distancia(origen); // return distancia(Point2I());
}
```

Otras Características

- ## Enumerators

```
enum Dias { Lunes, Martes, Miercoles, Jueves, Viernes };
Dias dia = Lunes;
```

- ## Typedef

```
typedef Point2<double> Point2D;
Point2D punto(3.2, 3.3);
```

- ## Operators

```
template <typename T>
class Point2
{
public:
    ...
    T &operator[](int i) {
        return (i > 0 ? m_x : m_y);
    }
};

Point2<int> punto;
punto[1] = 34; // Gracias a que devuelve T&
```

qmake

- Simplifica el proceso de compilación para distintas plataformas.
- Automatiza la generación de Makefiles.
- Sirve para proyectos que usen o no Qt.
- También llamados herramienta make-makefile o makemake.

Proyectos Simples

- Escribir código.
- ‘qmake −project‘ Genera el archivo .pro.
- ‘qmake‘ Genera los Makefiles.
- ‘make‘ Compila el programa.
- Ejemplo.

Variables

- TEMPLATE={app|lib}
- CONFIG
- TARGET
- VERSION (Solo para TEMPLATE=lib)
- DESTDIR
- HEADERS
- SOURCES
- FORMS
- RESOURCES

- TEMPLATE=subdirs
- SUBDIRS

Librerias

```
ejemplo/
    bin/
        app
        libGeometry.so
    src/
        app/
            app.pro
            main.cpp
        libs/
            geometry/
                geometry.pro
                point2.cpp
                point2.h
            libs.pro
        src.pro
    ejemplo.pro
```

Qt

- Librería C++ gráfica multiplataforma.
- LGPL en su versión 4.5.
- Incluye las siguientes herramientas:
  - Creación rápida de Dialogs y MainWindows (designer).
  - I18n (linguist).
  - Sistema de ayuda (assistant).
- Provee funcionalidades para:
  - Gráficos 2D y 3D.
  - Widgets con el patrón MVC (Item y Container).
  - IO
  - Acceso a bases de datos.
  - Networking.
  - XML.
  - Multithreading.
  - Plugins.

Conceptos Iniciales

- QObject.
- QWidget.
- QDialog.
- QMainApplication.
- QLayout.
- QWidget memory management.
- Sistema Signals/Slots.

Designer

- Drag & Drop de Widgets.
- Genera un encabezado: ui_<nombre_clase>.h
- Define una clase: Ui::NombreClase
- Que implementa el método: setupUi(QWidget *parent).

Estructura de Archivos

```
cc52b/
    bin/
    src/
        app/
            app.pro
            cc52bmainwindow.cpp
            cc52bmainwindow.h
            cc52bmainwindow.ui
            main.cpp
        libs/
            cc52b/
                // Implementacion de funcionalidades
                // en clases bien ordenadas.
                cc52b.pro
            cc52bwidget/
                // Extension de QGLWidget para que
                // funcione bien con esta estructura.
                cc52bwidget.pro
            libs.pro
        src.pro
    cc52b.pro
```

Otras Herramientas

- Git
- QtCreator