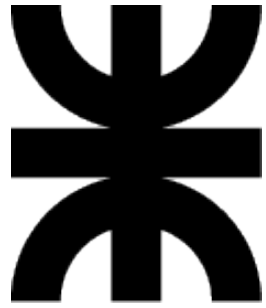
Informática II

Programación gráfica con Qt
Introducción

Gonzalo F. Perez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2024 –

Introducción

- Qt se pronuncia *cute*

Introducción

- Qt se pronuncia *cute*
- Es un marco de desarrollo (framework) de aplicaciones multiplataforma, de código abierto y gratuito para escritorio, embebido y móvil – https://www.qt.io/product/framework.

Introducción

- Qt se pronuncia *cute*
- Es un marco de desarrollo (framework) de aplicaciones multiplataforma, de código abierto y gratuito para escritorio, embebido y móvil – https://www.qt.io/product/framework.
- Es compatible con varias plataformas como Linux, OS X, Windows, VxWorks, QNX, Android, iOS y otras.

Introducción

- Qt se pronuncia *cute*
- Es un marco de desarrollo (framework) de aplicaciones multiplataforma, de código abierto y gratuito para escritorio, embebido y móvil – https://www.qt.io/product/framework.
- Es compatible con varias plataformas como Linux, OS X, Windows, VxWorks, QNX, Android, iOS y otras.
- Utiliza un preprocesador y compilador de Meta-Object para extender el lenguaje C ++ con características como señales y ranuras.

Introducción

- Qt se pronuncia *cute*
- Es un marco de desarrollo (framework) de aplicaciones multiplataforma, de código abierto y gratuito para escritorio, embebido y móvil – https://www.qt.io/product/framework.
- Es compatible con varias plataformas como Linux, OS X, Windows, VxWorks, QNX, Android, iOS y otras.
- Utiliza un preprocesador y compilador de Meta-Object para extender el lenguaje C ++ con características como señales y ranuras.
- Cuenta con un sistema de triple licencia:
  1. GPL v2/v3 para el desarrollo de software de código abierto y software libre,
  2. licencia de pago QPL para el desarrollo de aplicaciones comerciales y
  3. una licencia gratuita pensada para aplicaciones comerciales, LGPL.

Introducción

- La versión 5 incorporó muchas características nuevas, así como miles de correcciones de errores, lo que hace que Qt sea un kit de desarrollo realmente poderoso y estable.
- La última versión es la 6.7.2 de marzo del 2024.
  https://wiki.qt.io/Qt_version_history

Introducción

- La versión 5 incorporó muchas características nuevas, así como miles de correcciones de errores, lo que hace que Qt sea un kit de desarrollo realmente poderoso y estable.
- La última versión es la 6.7.2 de marzo del 2024.
  https://wiki.qt.io/Qt_version_history
- La última versión LTS es la 6.5.2 de abril del 2023.
- Las versiones LTS (long-term support) tienen soporte por 3 años.

Introducción

- La versión 5 incorporó muchas características nuevas, así como miles de correcciones de errores, lo que hace que Qt sea un kit de desarrollo realmente poderoso y estable.
- La última versión es la 6.7.2 de marzo del 2024.
  https://wiki.qt.io/Qt_version_history
- La última versión LTS es la 6.5.2 de abril del 2023.
- Las versiones LTS (long-term support) tienen soporte por 3 años.

Instalar Qt6: https://installati.one/install-qt6-base-dev-ubuntu-22-04/
Ver también: https://doc.qt.io/qt-6/linux.html

Introducción

- La versión 5 incorporó muchas características nuevas, así como miles de correcciones de errores, lo que hace que Qt sea un kit de desarrollo realmente poderoso y estable.
- La última versión es la 6.7.2 de marzo del 2024.
  https://wiki.qt.io/Qt_version_history
- La última versión LTS es la 6.5.2 de abril del 2023.
- Las versiones LTS (long-term support) tienen soporte por 3 años.

Instalar Qt6: https://installati.one/install-qt6-base-dev-ubuntu-22-04/
Ver también: https://doc.qt.io/qt-6/linux.html

Instalación: https://www.qt.io/download-dev

Introducción

- La versión 5 incorporó muchas características nuevas, así como miles de correcciones de errores, lo que hace que Qt sea un kit de desarrollo realmente poderoso y estable.
- La última versión es la 6.7.2 de marzo del 2024.
  https://wiki.qt.io/Qt_version_history
- La última versión LTS es la 6.5.2 de abril del 2023.
- Las versiones LTS (long-term support) tienen soporte por 3 años.

Instalar Qt6: https://installati.one/install-qt6-base-dev-ubuntu-22-04/
Ver también: https://doc.qt.io/qt-6/linux.html

Instalación: https://www.qt.io/download-dev

En Ubuntu 24.04 instalar los paquetes: `qt6-base-dev` y `qtcreator` (?).

Ejemplo de programa CLI (*Hola mundo*)

Editar el archivo `version.cpp`:

```
#include <QtCore>
#include <QTextStream>

int main()
{
    QTextStream out(stdout);
    out << "Hola mundo de Qt!" << Qt::endl;
    return 0;
}
```

Ejemplo de programa CLI (*Hola mundo*)

Editar el archivo `version.cpp`:

```
#include <QtCore>
#include <QTextStream>

int main()
{
    QTextStream out(stdout);
    out << "Hola mundo de Qt!" << Qt::endl;
    return 0;
}
```

Guardar el archivo y construir el programa:

Ejemplo de programa CLI (*Hola mundo*)

Editar el archivo `version.cpp`:

```
#include <QtCore>
#include <QTextStream>

int main()
{
    QTextStream out(stdout);
    out << "Hola mundo de Qt!" << Qt::endl;
    return 0;
}
```

Guardar el archivo y construir el programa:

```
g++ -o hola hola.cpp \
  -I/usr/include/x86_64-linux-gnu/qt6/QtCore/ \
  -I/usr/include/x86_64-linux-gnu/qt6/ \
  -L/usr/lib/x86_64-linux-gnu/ \
  -lQt6Core -fPIC
```

Ejemplo de programa GUI

```
#include <QApplication>
#include <QWidget>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QWidget window;

    window.resize(400, 200);
    window.setWindowTitle("Mi primer programa Qt");
    window.show();

    return app.exec();
}
```

Ejemplo de programa GUI

```
#include <QApplication>
#include <QWidget>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QWidget window;

    window.resize(400, 200);
    window.setWindowTitle("Mi primer programa Qt");
    window.show();

    return app.exec();
}
```

Construcción con `CMake` (archivo `CMakeLists.txt`):

```
> mkdir build
> cd build
> cmake ..
> make
```

Ejecutar binario.

Ejemplo de programa GUI

- La clase `QApplication` se ocupa entre otras cosas de:
  - los argumentos de entrada y
  - del bucle de eventos.

Ejemplo de programa GUI

- La clase `QApplication` se ocupa entre otras cosas de:
  - los argumentos de entrada y
  - del bucle de eventos.
- El bucle de eventos es un bucle que espera la entrada del usuario en las aplicaciones GUI.

Ejemplo de programa GUI

- La clase `QApplication` se ocupa entre otras cosas de:
  - los argumentos de entrada y
  - del bucle de eventos.
- El bucle de eventos es un bucle que espera la entrada del usuario en las aplicaciones GUI.
- Al llamar a `app.exec()` se inicia el bucle de eventos.

Ejemplo de programa GUI

- La clase `QApplication` se ocupa entre otras cosas de:
  - los argumentos de entrada y
  - del bucle de eventos.
- El bucle de eventos es un bucle que espera la entrada del usuario en las aplicaciones GUI.
- Al llamar a `app.exec()` se inicia el bucle de eventos.
- `QWidget` es la clase base de todos los objetos de la GUI.

Construcción con CMake (CLI)

Archivo CMakeLists.txt

```
1  cmake_minimum_required(VERSION 3.5)
2
3  project(hola VERSION 1.0.0 LANGUAGES CXX)
4
5  set(CMAKE_CXX_STANDARD 17)
6  set(CMAKE_CXX_STANDARD_REQUIRED ON)
7
8  find_package(Qt6 COMPONENTS Core REQUIRED)
9
10 add_executable(hola hola.cpp)
11 target_link_libraries(hola Qt6::Core)
```

Construcción con CMake (CLI)

Archivo CMakeLists.txt

```
1  cmake_minimum_required(VERSION 3.5)
2
3  project(hola VERSION 1.0.0 LANGUAGES CXX)
4
5  set(CMAKE_CXX_STANDARD 17)
6  set(CMAKE_CXX_STANDARD_REQUIRED ON)
7
8  find_package(Qt6 COMPONENTS Core REQUIRED)
9
10 add_executable(hola hola.cpp)
11 target_link_libraries(hola Qt6::Core)
```

cmake_minimum_required: especifica la versión mínima de CMake

project: establece el nombre y versión del proyecto

set: fija un valor a una variable

Construcción con CMake (CLI)

Archivo CMakeLists.txt

```
1  cmake_minimum_required(VERSION 3.5)
2
3  project(hola VERSION 1.0.0 LANGUAGES CXX)
4
5  set(CMAKE_CXX_STANDARD 17)
6  set(CMAKE_CXX_STANDARD_REQUIRED ON)
7
8  find_package(Qt6 COMPONENTS Core REQUIRED)
9
10 add_executable(hola hola.cpp)
11 target_link_libraries(hola Qt6::Core)
```

cmake_minimum_required: especifica la versión mínima de CMake

project: establece el nombre y versión del proyecto

set: fija un valor a una variable

find_package: indica a CMake que busque Qt6

add_executable: indica crear un ejecutable/binario a partir de uno o más archivos fuentes

target_link_libraries: indica en enlazado con la biblioteca

Construcción con CMake (GUI)

Archivo CMakeLists.txt

```
1  cmake_minimum_required(VERSION 3.5)
2
3  project(window VERSION 1.0.0 LANGUAGES CXX)
4
5  set(CMAKE_CXX_STANDARD 17)
6  set(CMAKE_CXX_STANDARD_REQUIRED ON)
7
8  set(CMAKE_AUTOMOC ON)
9  set(CMAKE_AUTORCC ON)
10 set(CMAKE_AUTOUIC ON)
11
12 find_package(Qt6 COMPONENTS Widgets REQUIRED)
13
14 add_executable(window main.cpp)
15 target_link_libraries(window Qt6::Widgets)
```

Construcción con CMake (GUI)

Archivo CMakeLists.txt

```
1  cmake_minimum_required(VERSION 3.5)
2
3  project(window VERSION 1.0.0 LANGUAGES CXX)
4
5  set(CMAKE_CXX_STANDARD 17)
6  set(CMAKE_CXX_STANDARD_REQUIRED ON)
7
8  set(CMAKE_AUTOMOC ON)
9  set(CMAKE_AUTORCC ON)
10 set(CMAKE_AUTOUIC ON)
11
12 find_package(Qt6 COMPONENTS Widgets REQUIRED)
13
14 add_executable(window main.cpp)
15 target_link_libraries(window Qt6::Widgets)
```

AUTOMOC: Meta-Object Compiler (moc)

AUTORCC: Resource Compiler (rcc)

AUTOUIC: User Interface Compiler (uic)

Ver ejemplo con Qt Creator (etiquete o `Label`)