

QT

Introducción

QT

- ❖ Es una biblioteca de desarrollo de software de código abierto
- ❖ Está desarrollada en C++
- ❖ QT se puede emplear con C, C++, Ruby, Perl, PHP y Python principalmente
- ❖ El uso en lenguajes diferentes a C/C++ se le conoce con el nombre de Bindings
- ❖ Se puede instalar en Windows, Linux y Mac OS X

EJEMPLO 1

¿Cómo iniciar?

- ❖ Abrimos *QT Creator* (Figura 1)
- ❖ Ingresamos a la opción *File* localizada en la barra de menú
- ❖ Damos clic en la opción *New Project...* (Figura 2)
- ❖ Una vez abierta la ventana para seleccionar un *template*, escogemos *Application(QT)* junto con *Qt Console Application* (Figura 3)
- ❖ Damos clic en el botón *Choose*
- ❖ Le damos un nombre a la aplicación, en este caso la llamaremos *ejemplo1* y seleccionamos la carpeta de trabajo (debe haber sido creada previamente) (Figura 4)
- ❖ En la opción *Build System* seleccionamos *qmake* (Figura 5)
- ❖ En *Translation File* seleccionamos *None* (Figura 6)
- ❖ En *Kits* seleccionamos *Desktop Qt 6.5.3 MSVC20 64 bit* (Figura 7)
- ❖ En *Summary* damos clic en *Finish*

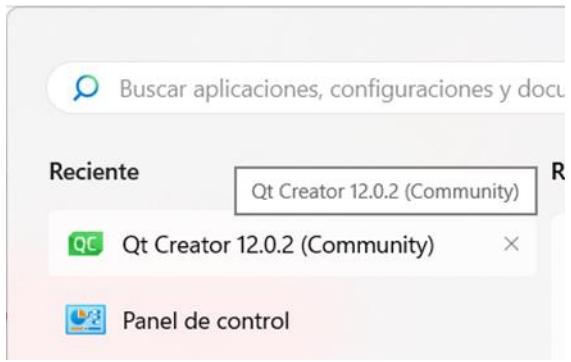


Figura 1

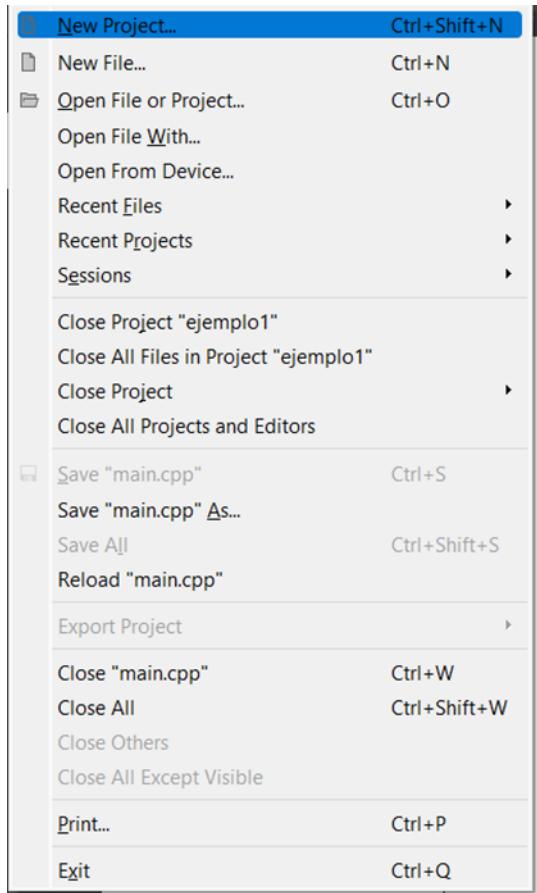


Figura 2

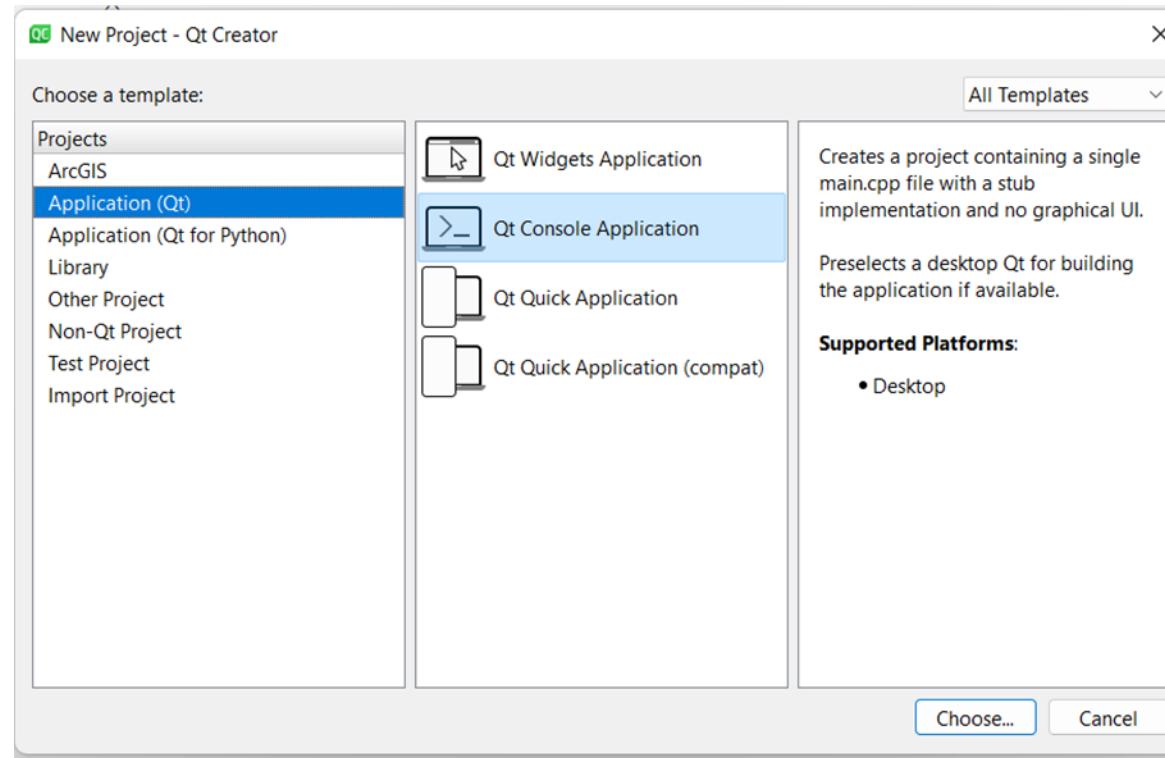


Figura 3

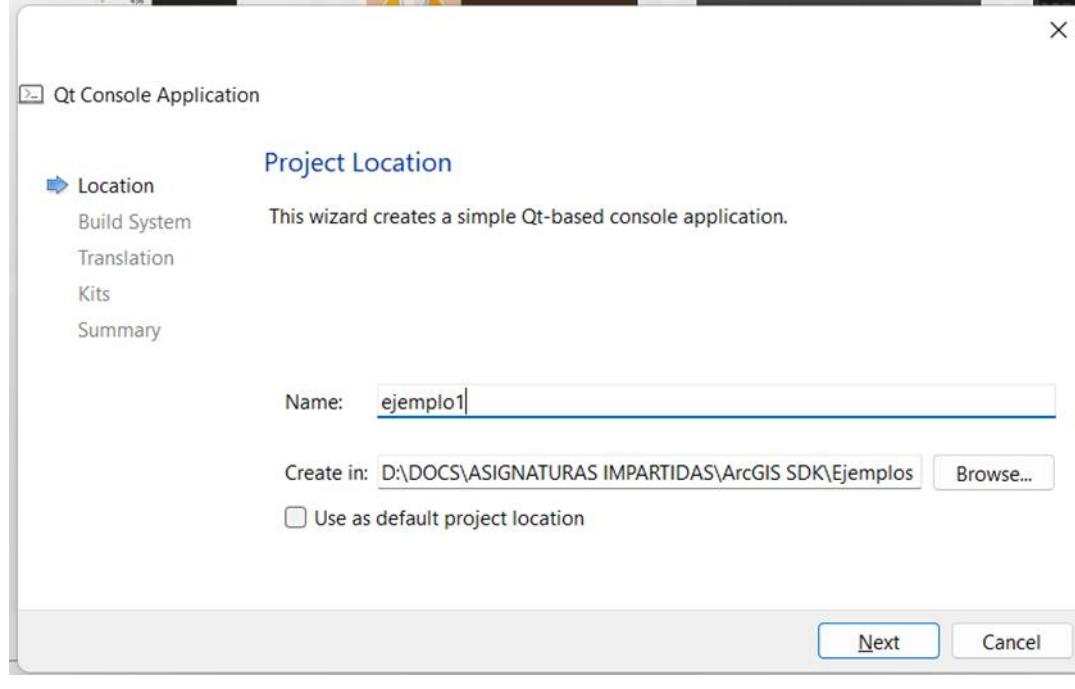


Figura 4

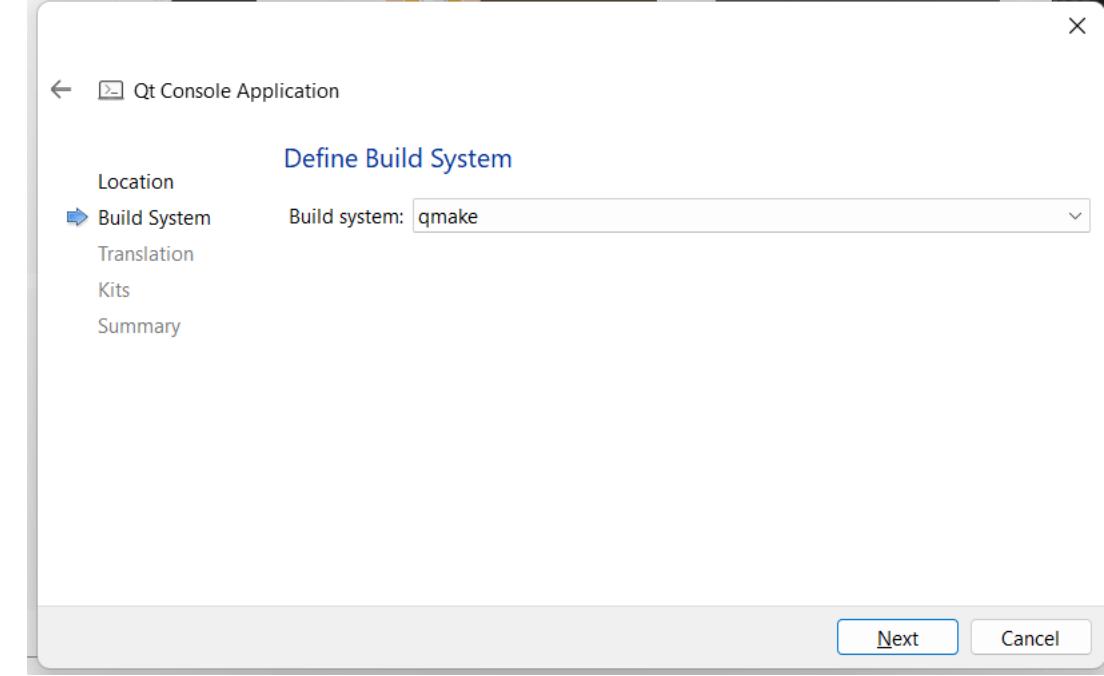


Figura 5

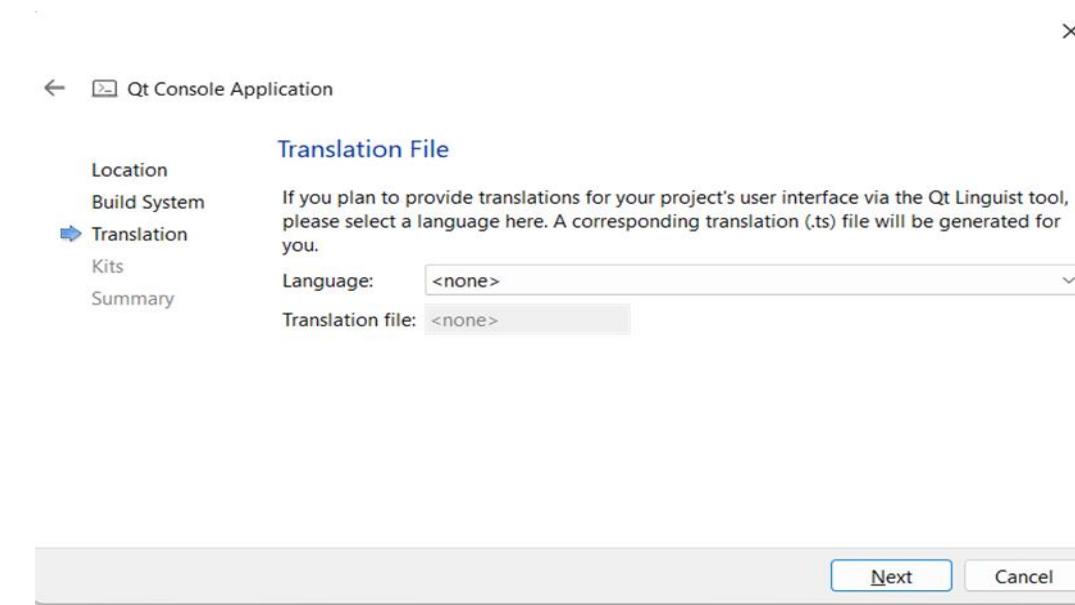


Figura 6

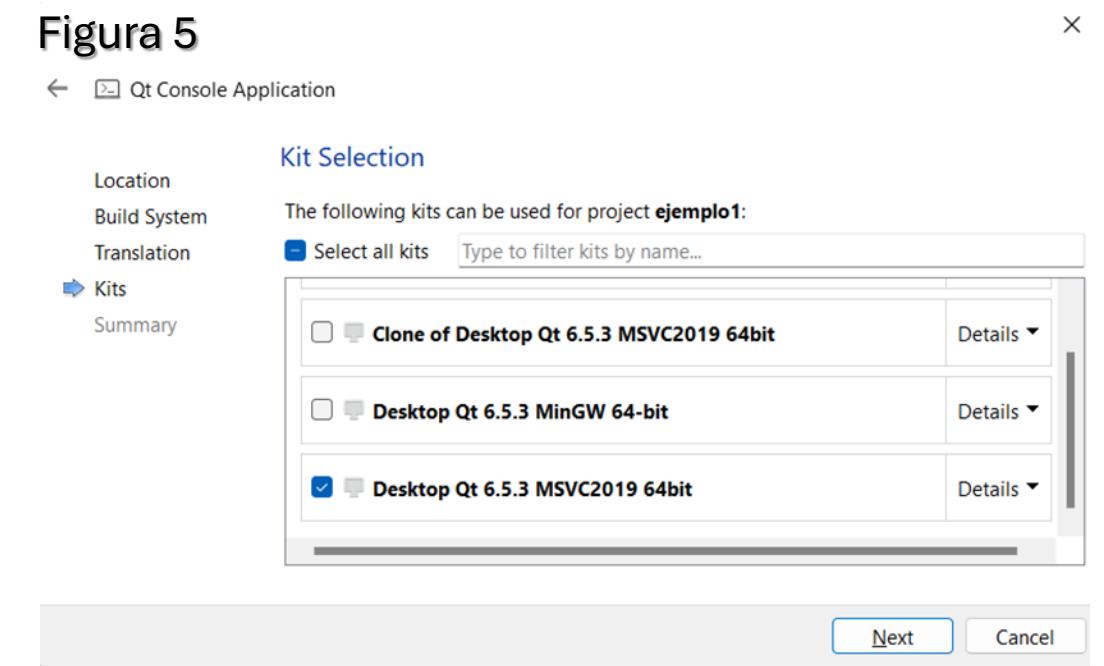


Figura 7

¿Cómo iniciar?

```
#include <QCoreApplication>
```

```
int main(int argc, char *argv[]) //Recibe 2 argumentos, un entero y un arreglo de puntero de caracteres
```

```
// argc representa el tamaño del arreglo argv
```

```
{
```

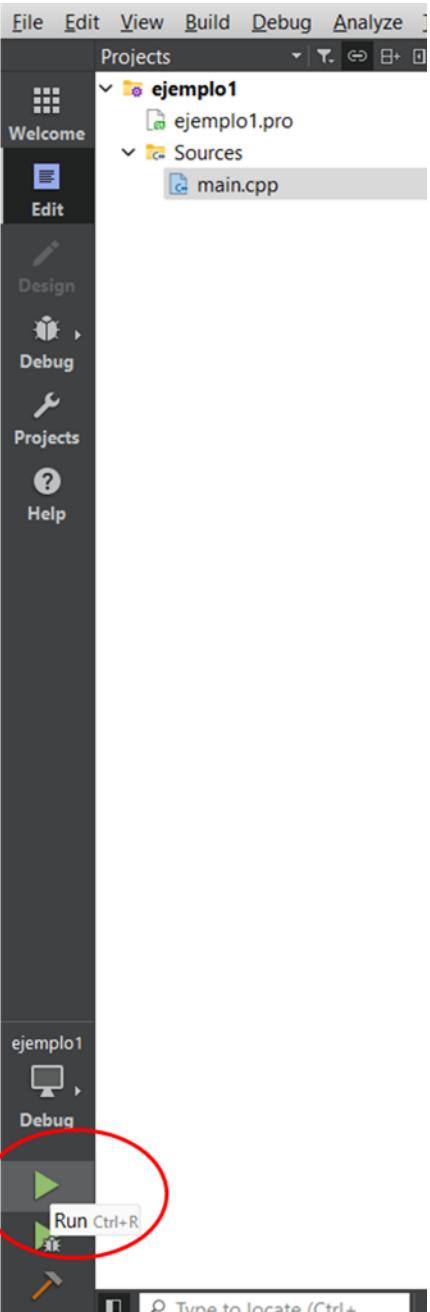
```
    QCoreApplication a(argc, argv); // Crear un objeto de la clase QCoreApplication llamado a
```

```
    return a.exec(); //exec genera un bucle para que la aplicación permanezca abierta hasta destruirla (es decir cerrar el aplicativo)
```

```
}
```

¿Cómo iniciar?

```
#include <QCoreApplication>
#include <QDebug>
int main(int argc, char *argv[]) //Recibe 2 argumentos, un entero y un arreglo de puntero de caracteres argc representa el
tamaño del arreglo argv
{
    QCoreApplication a(argc, argv); // Crear un objeto de la clase QCoreApplication llamado a
    qDebug()<<"Hola Mundo"; // Nos permite imprimir texto usando QT
    return a.exec(); //exec genera un bucle para que la aplicación permanezca abierta hasta destruirla (es decir cerrar el aplicativo)
}
```



❖Ejecutamos la aplicación

The screenshot shows the Qt Creator interface with the application output window open. The output log shows several entries, with the last one being 'Hola mundo', which is circled in red.

```
10:05:34: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe...
10:07:51: D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe crashed.

10:07:52: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe...
10:12:01: D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe crashed.

10:12:04: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe...
Hola mundo
```

¿Cómo iniciar?

```
#include <QCoreApplication>
#include <QDebug>
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv); // Recibe 2 argumentos, un entero y un arreglo de puntero de caracteres argc representa el tamaño del arreglo
    QString Saludo = "Hola Mundo"; // Crea variables de tipo texto cadena
    qDebug() << Saludo; // Nos permite imprimir texto usando QT
    return a.exec(); // exec genera un bucle para que la aplicación permanezca abierta hasta destruirla (es decir cerrar el aplicativo)
}
```

File Edit View Build Debug Analyze Tools Window Help

Projects

ejemplo1

ejemplo1.pro

Sources

main.cpp

```
1 #include <QCoreApplication>
2 #include<QDebug>
3 int main(int argc, char *argv[])
4 {
5     QCoreApplication a(argc, argv);
6     QString Saludo="Hola mundo";
7     qDebug()<< Saludo;
8
9
10}
11
```

Application Output

ejemplo1

```
10:05:34: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe...
10:07:51: D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe crashed.

10:07:52: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe...
10:12:01: D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe crashed.

10:12:04: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe...
Hola mundo
10:25:00: D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe crashed.

10:25:03: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe...
"Hello mundo"
```

ejemplo1

Debug

Type to locate (Ctrl+...)

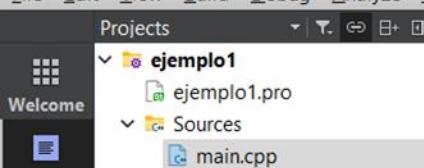
1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 Terminal 6 Version Control 7 Test Results 8 QML Debugger Console 9 General Messages

Line: 7, Col: 22

¿Cómo iniciar?

```
#include <QCoreApplication>
#include <QDebug>
int main(int argc, char *argv[]) //Recibe 2 argumentos, un entero y un arreglo de puntero de caracteres argc representa el tamaño del arreglo argv
{
    QCoreApplication a(argc, argv); // Crear un objeto de la clase QCoreApplication llamado a
    int x = 1;
    double y=3.14;
    char z= '#';
    QString texto; //Crea variables de tipo texto cadena
    texto = QString("%1 %2 %3").arg(x).arg(y).arg(z); //concatena las variables x,y,z en un texto
    qDebug()<< texto; // Nos permite imprimir texto usando QT
    return a.exec(); //exec genera un bucle para que la aplicación permanezca abierta hasta destruirla (es decir cerrar el aplicativo)
}
```

File Edit View Build Debug Analyze Tools Window Help



```
1 #include <QCoreApplication>
2 #include<QDebug>
3 int main(int argc, char *argv[])
4 {
5     QCoreApplication a(argc, argv);
6     int x= 1;
7     double y =3.14;
8     char z='#!';
9     QString texto;
10    texto = QString("%1 %2 %3").arg(x).arg(y).arg(z);
11    qDebug()<< texto;
12
13
14
15 }
```

Application Output | Filter + -

ejemplo1

"Hola mundo"

11:07:03: D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe crashed.

11:07:05: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe...
QString::arg: Argument missing: 1 %b 3.14, #
"1 %b 3.14"

11:07:31: D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe crashed.

11:07:34: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe...
"1 3.14 #"

11:07:49: D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe crashed.

11:07:52: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo1-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo1.exe...
"1 3.14 #"

Type to locate (Ctrl+...)

1 Issues 2 Search Results 3 Application Output 4 Compile Output 5 Terminal 6 Version Control 7 Test Results 8 QML Debugger Console 9 General Messages

¿Cómo iniciar? – Cómo convertir un string de C++ a un string en QT

```
#include <QCoreApplication>
#include <QDebug>
using namespace std;
int main(int argc, char *argv[]) //Recibe 2 argumentos, un entero y un arreglo de puntero de caracteres argc representa el tamaño del arreglo
argv
{
    QCoreApplication a(argc, argv); // Crear un objeto de la clase QCoreApplication llamado a
    string x = “cadena de texto en C++”;
    QString tx = x.c_str(); //Convierte el string en QString, c_str() define una secuencia de caracteres en el lenguaje C
    qDebug()<< tx; // Nos permite imprimir texto usando QT
    return a.exec(); //exec genera un bucle para que la aplicación permanezca abierta hasta destruirla (es decir cerrar el aplicativo)
}
```

¿Cómo iniciar? – Cómo convertir un QString a un string de c++

```
#include <QCoreApplication>
#include <QDebug>
using namespace std;
int main(int argc, char *argv[]) //Recibe 2 argumentos, un entero y un arreglo de puntero de caracteres argc representa el tamaño del arreglo
argv
{
    QCoreApplication a(argc, argv); // Crear un objeto de la clase QCoreApplication llamado a
    QString tx =“Cadena de texto en QT”;
    string x = tx.toStdString(); //Convierte la cadena de texto de QT a cadena d texto en C++
    qDebug()<< x; // Nos permite imprimir texto usando QT
    return a.exec(); //exec genera un bucle para que la aplicación permanezca abierta hasta destruirla (es decir cerrar el aplicativo)
}
```

¿Cómo iniciar? – Cómo buscar contenido de palabras en una cadena de texto

```
#include <QCoreApplication>
#include <QDebug>
int main(int argc, char *argv[]) //Recibe 2 argumentos, un entero y un arreglo de puntero de caracteres argc representa el
tamaño del arreglo argv
{
    QCoreApplication a(argc, argv); // Crear un objeto de la clase QCoreApplication llamado a
    QString tx =“Cadena de texto en QT”;
    bool verdadero = tx.contains(“QT”); //Busca si la palabra QT esta contenida en la cadena de texto
    qDebug()<< verdadero; // Nos permite imprimir texto usando QT
    return a.exec(); //exec genera un bucle para que la aplicación permanezca abierta hasta destruirla (es decir cerrar el aplicativo)
}
```

¿Cómo iniciar? – Cómo buscar contenido de la primera palabra en una cadena de texto

```
#include <QCoreApplication>
#include <QDebug>
int main(int argc, char *argv[]) //Recibe 2 argumentos, un entero y un arreglo de puntero de caracteres argc representa el
tamaño del arreglo argv
{
    QCoreApplication a(argc, argv); // Crear un objeto de la clase QCoreApplication llamado a
    QString tx =“Cadena de texto en QT”;
    bool verdadero = tx.startsWith(“Cadena”); //verifica si la primera palabra se llama Cadena
    qDebug()<< verdadero; // Nos permite imprimir texto usando QT
    return a.exec(); //exec genera un bucle para que la aplicación permanezca abierta hasta destruirla (es decir cerrar el aplicativo)
}
```

¿Cómo iniciar? – Cómo buscar índices de caracteres en una cadena de texto

```
#include <QCoreApplication>
#include <QDebug>
int main(int argc, char *argv[]) //Recibe 2 argumentos, un entero y un arreglo de puntero de caracteres argc representa el
tamaño del arreglo argv
{
    QCoreApplication a(argc, argv); // Crear un objeto de la clase QCoreApplication llamado a
    QString tx =“Cadena de texto en QT”;
    int indice = tx.indexOf(“e”); //Como hay varias e, en este caso muestra la posición de la primera e
    qDebug()<
```

¿Cómo iniciar? – Cómo buscar el carácter localizado en un determinado índice

```
#include <QCoreApplication>
#include <QDebug>
int main(int argc, char *argv[]) //Recibe 2 argumentos, un entero y un arreglo de puntero de caracteres argc representa el
tamaño del arreglo argv
{
    QCoreApplication a(argc, argv); // Crear un objeto de la clase QCoreApplication llamado a
    QString tx =“Cadena de texto en QT”;
    qDebug()<< tx.at(10); // Nos permite imprimir texto usando QT, at() busca el caracter localizado en una determinada posicion
    return a.exec(); //exec genera un bucle para que la aplicación permanezca abierta hasta destruirla (es decir cerrar el aplicativo)
}
```

EJEMPLO 2

```

#include <QCoreApplication>
#include <QDebug>

class Persona : public QObject //Creamos la clase Persona
{
public: //Especificamos los miembros (atributos y métodos) de la clase que son accesibles desde fuera de la clase
    Persona(QObject *Padre = NULL):QObject(Padre)
    {

    }
    //Metodos o funciones pertenecientes a la clase
    void setNombre(const QString &nombre)
    {m_nombre = nombre;}
    void setEdad(int edad)
    {m_edad = edad;}
    void setSalario(double salario)
    {m_salario = salario;}

    QString getNombre() const
    {return m_nombre;}
    int getEdad() const
    {return m_edad;}
    double getSalario() const
    {return m_salario;}

    void Imprime() const
    {
        qDebug()<< m_nombre << " " << m_edad << " " << m_salario;
    }

private:
    QString m_nombre;
    int m_edad;
    double m_salario;
};

```

¿Cómo iniciar? – Cómo clases en QT

```
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    Persona *Juan = new Persona; //Creamos al objeto Juan
    Persona *Pedro = new Persona(Juan); //Creamos al objeto Pedro el cual es hijo de Juan
    Persona *Luis = new Persona(Juan); //Creamos al objeto Luis el cual es hijo de Juan

    Juan -> setNombre("Juan");
    Juan -> setEdad(37);
    Juan -> setSalario(3600000);
    Juan -> Imprime();

    Pedro -> setNombre("Pedro");
    Pedro -> setEdad(16);
    Pedro -> setSalario(0);
    Pedro -> Imprime();

    Luis -> setNombre("Luis");
    Luis -> setEdad(10);
    Luis -> setSalario(0);
    Luis -> Imprime();

    return a.exec();
}
```

¿Cómo iniciar? – Cómo clases en QT

```
14:27:01: Starting D:\DOCS
"Juan"    37    3.6e+06
"Pedro"   16    0
"Luis"    10    0
```

EJEMPLO 3

¿Cómo iniciar? – Archivos en QT → lectura

The screenshot shows a Qt-based IDE interface with the following details:

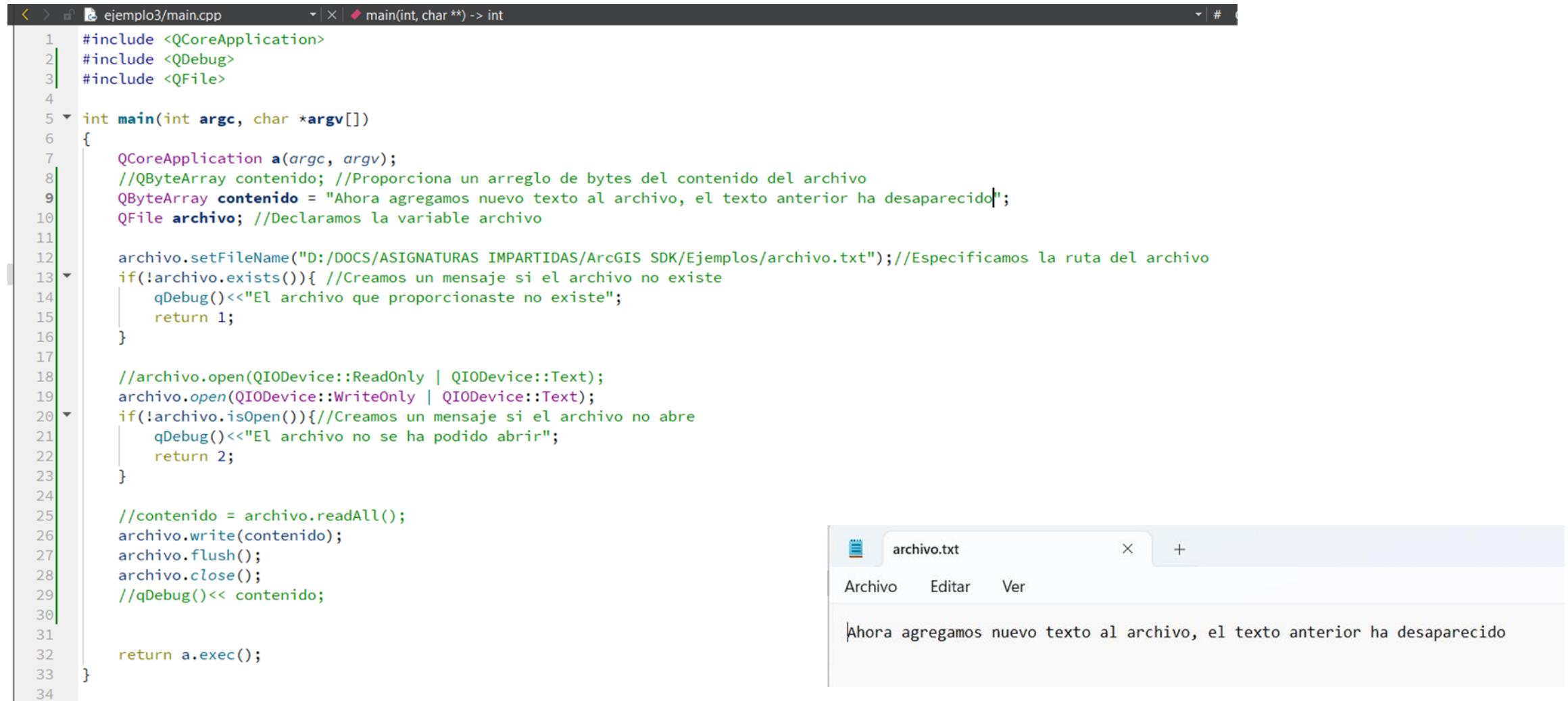
- Menu Bar:** Tools, Window, Help.
- Title Bar:** ejemplo3/main.cpp, main(int, char **) -> int.
- Status Bar:** # CRLF, Line: 24, Col: 26.
- Code Editor:** Displays the following C++ code:

```
1 #include <QCoreApplication>
2 #include <QDebug>
3 #include <QFile>
4
5 int main(int argc, char *argv[])
6 {
7     QCoreApplication a(argc, argv);
8     QByteArray contenido; //Proporciona un arreglo de bytes del contenido del archivo
9     QFile archivo; //Declaramos la variable archivo
10
11     archivo.setFileName("D:/DOCS/ASIGNATURAS IMPARTIDAS/ArcGIS SDK/Ejemplos/archivo.txt"); //Especificamos la ruta del archivo
12     if(!archivo.exists()){ //Creamos un mensaje si el archivo no existe
13         qDebug() << "El archivo que proporcionaste no existe";
14         return 1;
15     }
16
17     archivo.open(QIODevice::ReadOnly | QIODevice::Text);
18     if(!archivo.isOpen()){ //Creamos un mensaje si el archivo no abre
19         qDebug() << "El archivo no se ha podido abrir";
20         return 2;
21     }
22
23     contenido = archivo.readAll();
24     qDebug() << contenido;
25
26
27     return a.exec();
28 }
29
```

Application Output: Shows the output of the application's execution.

```
15:14:47: Starting D:/DOCS/ASIGNATURAS IMPARTIDAS/ArcGIS SDK/Ejemplos/build-ejemplo3-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo3.exe...
"Esto es un archivo de prueba"
```

¿Cómo iniciar? – Archivos en QT → escritura



The image shows a Qt-based IDE interface. On the left, the code editor displays `ejemplo3/main.cpp` with the following C++ code:

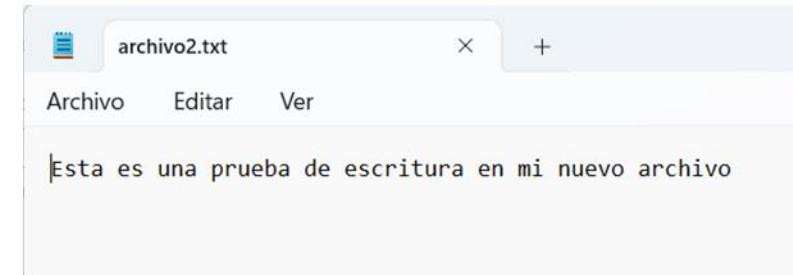
```
1 #include <QCoreApplication>
2 #include <QDebug>
3 #include <QFile>
4
5 int main(int argc, char *argv[])
6 {
7     QCoreApplication a(argc, argv);
8     //QByteArray contenido; //Proporciona un arreglo de bytes del contenido del archivo
9     QByteArray contenido = "Ahora agregamos nuevo texto al archivo, el texto anterior ha desaparecido";
10    QFile archivo; //Declaramos la variable archivo
11
12    archivo.setFileName("D:/DOCS/ASIGNATURAS IMPARTIDAS/ArcGIS SDK/Ejemplos/archivo.txt"); //Especificamos la ruta del archivo
13    if(!archivo.exists()){ //Creamos un mensaje si el archivo no existe
14        qDebug()<<"El archivo que proporcionaste no existe";
15        return 1;
16    }
17
18    //archivo.open(QIODevice::ReadOnly | QIODevice::Text);
19    archivo.open(QIODevice::WriteOnly | QIODevice::Text);
20    if(!archivo.isOpen()){//Creamos un mensaje si el archivo no abre
21        qDebug()<<"El archivo no se ha podido abrir";
22        return 2;
23    }
24
25    //contenido = archivo.readAll();
26    archivo.write(contenido);
27    archivo.flush();
28    archivo.close();
29    //qDebug()<< contenido;
30
31
32    return a.exec();
33
34 }
```

On the right, a file viewer window titled "archivo.txt" shows the content:

Ahora agregamos nuevo texto al archivo, el texto anterior ha desaparecido

¿Cómo iniciar? – Archivos en QT → crear archivo, escribir y leer

```
1 #include <QCoreApplication>
2 #include <QDebug>
3 #include <QFile>
4 #include<QTextStream>
5
6 int main(int argc, char *argv[])
7 {
8     QApplication a(argc, argv);
9     QTextStream entradaSalida;
10    QFile archivo; //Declaramos la variable archivo
11
12    archivo.setFileName("D:/DOCS/ASIGNATURAS IMPARTIDAS/ArcGIS SDK/Ejemplos/archivo2.txt"); //Damos nombre al archivo
13
14    archivo.open(QIODevice::ReadWrite | QIODevice::Text);
15    if(!archivo.isOpen()){//Creamos un mensaje si el archivo no abre
16        qDebug()<<"El archivo no se ha podido abrir";
17        return 1;
18    }
19
20    entradaSalida.setDevice(&archivo); //Asignamos la direccion del archivo archivo
21    entradaSalida<<"Esta es una prueba de escritura en mi nuevo archivo";
22    archivo.flush(); //Limpiamos
23    archivo.close(); //Cerramos
24
25
26    return a.exec();
27 }
28 }
```



¿Cómo iniciar? – Archivos en QT → crear archivo, escribir y leer

The screenshot shows a Qt IDE interface with the following details:

- Code Editor:** Displays C++ code for file operations. The code initializes a QApplication, creates QTextStream objects for input and output, opens a QFile named "archivo2.txt" for reading and writing, writes a message to it, and then reads the content back. It also includes error handling for file opening.
- Output Window:** Shows the application's output. A red box highlights the output line: "15:51:43: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo3-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo3.exe...". Below it, the message "Esta es una prueba de escritura en mi nuevo archivo" is displayed.

```
6 int main(int argc, char *argv[])
7 {
8     QApplication a(argc, argv);
9     QTextStream entradaSalida;
10    QFile archivo; //Declaramos la variable archivo
11    QString contenido;
12
13    archivo.setFileName("D:/DOCS/ASIGNATURAS IMPARTIDAS/ArcGIS SDK/Ejemplos/archivo2.txt");//Damos nombre al archivo
14
15    archivo.open(QIODevice::ReadWrite | QIODevice::Text);
16    if(!archivo.isOpen()){//Creamos un mensaje si el archivo no abre
17        qDebug()<<"El archivo no se ha podido abrir";
18        return 1;
19    }
20
21    entradaSalida.setDevice(&archivo); //Asignamos la direccion del archivo archivo
22    entradaSalida<<"Esta es una prueba de escritura en mi nuevo archivo";
23    archivo.flush();//Limpiamos
24    archivo.close();//Cerramos
25
26    archivo.open(QIODevice::ReadWrite | QIODevice::Text);
27    if(!archivo.isOpen()){//Creamos un mensaje si el archivo no abre
28        qDebug()<<"El archivo no se ha podido abrir";
29        return 2;
30    }
31
32    contenido=entradaSalida.readAll();//Vamos a leer lo que escribimos
33    qDebug() << contenido;
34    archivo.flush();//Limpiamos
35    archivo.close();//Cerramos
36
37    return a.exec();
38
39 }
```

Application Output

ejemplo1 exemplo2 ejemplo3

15:51:43: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo3-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo3.exe...
"Esta es una prueba de escritura en mi nuevo archivo"

EJEMPLO 4

¿Cómo iniciar? – Listas en QT

The screenshot shows a code editor window with the following details:

- Title Bar:** Shows the file name "ejemplo4/main.cpp" and the function signature "main(int, char **) -> int".
- Code Area:** Displays the following C++ code:

```
1 #include <QCoreApplication>
2 #include <QDebug>
3 int main(int argc, char *argv[])
4 {
5     QCoreApplication a(argc, argv);
6     QList<int> L; //Vamos a crear una lista de valores enteros declarada en la variable L
7     L << 1 << 2 << 3 << 4 << 5 << 6 << 7 << 8;
8     qDebug()<<L;
9     return a.exec();
10 }
11
```
- Bottom Line:** Shows the expression "QList(1, 2, 3, 4, 5, 6, 7, 8)" which is the output of the qDebug() call.

QList(1, 2, 3, 4, 5, 6, 7, 8)

¿Cómo iniciar? – Listas en QT

```
1 #include <QCoreApplication>
2 #include <QDebug>
3 int main(int argc, char *argv[])
4 {
5     QCoreApplication a(argc, argv);
6     QList<int> L; //Vamos a crear una lista de valores enteros declarada en la variable L
7     L << 1 << 2 << 3 << 4 << 5 << 6 << 7 << 8;
8     QList<int>::Iterator i; //Vamos a crear el iterador i para iterar la lista
9
10    int suma =0;//vamos hacer la suma de los numeros de la lista
11
12    for( i = L.begin(); i != L.end(); i++ )
13        suma += *i;
14    qDebug()<<L;
15    qDebug() << suma;
16    return a.exec();
17 }
18 }
```

```
QList(1, 2, 3, 4, 5, 6, 7, 8)
```

¿Cómo iniciar? – Listas en QT

```
1 #include <QCoreApplication>
2 #include <QDebug>
3 int main(int argc, char *argv[])
4 {
5     QCoreApplication a(argc, argv);
6     QList<int> L; //Vamos a crear una lista de valores enteros declarada en la variable L
7     L << 1 << 2 << 3 << 4 << 5 << 6 << 7 << 8;
8     QList<int>::Iterator i; //Vamos a crear el iterador i para iterar la lista
9
10    int suma =0;//vamos hacer la suma de los numeros de la lista
11
12    for( i = L.begin(); i != L.end(); i++ )
13        suma += *i;
14    qDebug()<<L;
15    qDebug() << suma;
16
17    L.remove(2); //Vamos a remover el dato que esta en la tercera posicion
18    qDebug()<<L;
19
20    return a.exec();
21 }
22 }
```

```
QList(1, 2, 3, 4, 5, 6, 7, 8)
36
QList(1, 2, 4, 5, 6, 7, 8)
```

¿Cómo iniciar? – Listas en QT

```
> ejemplos/main.cpp | X | main(int, char **) -> int
1 #include <QCoreApplication>
2 #include <QDebug>
3 int main(int argc, char *argv[])
4 {
5     QCoreApplication a(argc, argv);
6     QList<int> L; //Vamos a crear una lista de valores enteros declarada en la variable L
7     L << 1 << 2 << 3 << 4 << 5 << 6 << 7 << 8;
8     QList<int>::Iterator i; //Vamos a crear el iterador i para iterar la lista
9
10    int suma =0;//vamos hacer la suma de los numeros de la lista
11
12    for( i = L.begin(); i != L.end(); i++ )
13        suma += *i;
14    qDebug()<<L;
15    qDebug() << suma;
16
17    L.remove(2); //Vamos a remover el dato que esta en la tercera posicion
18    qDebug()<<L;
19
20    foreach(int i, L){//similar al loop for
21        qDebug()<<i;
22    }
23
24    return a.exec();
25 }
```

QList(1, 2, 3, 4, 5, 6, 7, 8)
36
QList(1, 2, 4, 5, 6, 7, 8)
1
2
4
5
6
7
8

EJEMPLO 5

¿Cómo iniciar? – Registros en QT

The screenshot shows a Qt-based IDE interface. On the left, the code editor displays `main.cpp` with the following content:

```
1 #include <QCoreApplication>
2 #include<QTextStream>
3 #include<QDebug>
4 #include<QFile>
5 #include<QMap>
6 #include<ctime>
7
8 int main(int argc, char *argv[])
9 {
10     QCoreApplication a(argc, argv);
11     QTextStream entradaSalida; //Stream de entrada y salida
12     QFile archivo; //Archivo
13     QMap<int, int> registro; //Registro
14     //Vamos a registrar el numero que esta presente en el archivo y la cantidad de veces que el numero esta presente
15
16     srand(time(NULL));
17
18     archivo.setFileName("D:/DOCS/ASIGNATURAS IMPARTIDAS/ArcGIS SDK/Ejemplos/numeros.txt");
19     archivo.open(QIODevice::WriteOnly | QIODevice::Text);
20
21     entradaSalida.setDevice(&archivo); //Asigna el archivo al dispositivo de entrada y salida
22
23     for(int i=0; i<10000;i++)
24         entradaSalida << rand()%10+1 << ' ';
25
26     archivo.flush();
27     archivo.close();
28
29     return a.exec();
30 }
31
```

A yellow warning box highlights the line `QMap<int, int> registro;` with the message `unused QMap<int, int> [clazy-unused-non-trivial-variable]`. The code editor also shows line numbers from 1 to 31.

On the right, a file viewer window titled "numeros.txt" displays a large list of integers, each followed by a space. The list starts with 2, 10, 7, 8, 3, 10, 2, 10, 7, 4, 6, 8, 5, 4, 8, 3, 2, 4, 5, 5, 10, 2, 10, 10, 6, 6, 5, 2, 8, 3, 8, 1, 5, 4, 2, 8, 4, 3, 10, 7, 1, 4, 3, 9, 6, 5, 8, 2, 10, 8, 1, 5, 3, 9, 6, 1, 2, 2, 8, 7, 1, 2, 8, 3, 2, 3, 5, 7, 1, 5, 9, 2, 6, 3, 6, 7, 1, 7, 10, 7, 3, 6, 7, 10, 9, 10, 4, 6, 8, 10, 6, 2, 4, 8, 3, 7, 4, 8, 1, 2, 10, 8, 1, 2, 1, 3, 1, 9, 1, 6, 2, 9, 1, 5, 7, 6, 1, 4, 6, 8, 6, 8, 10, 6, 1, 10, 6, 8, 1, 10, 6, 10, 4, 8, 1, 9, 1, 1, 9, 7, 7, 1, 4, 4, 4, 1, 1, 8, 4, 8, 2, 4, 10, 1, 2, 6, 1, 6, 8, 9, 5, 6, 7, 2, 8, 10, 2, 5, 8, 1, 4, 6, 10, 6, 2, 2, 3, 10, 8, 7, 7, 1, 5, 1, 10, 6, 8, 7, 3, 8, 3, 3, 4, 9, 2, 6, 3, 9, 2, 9, 7, 6, 3, 8, 5, 10, 9, 5, 2, 1, 8, 1, 1, 4, 4, 7, 4, 3, 4, 2, 7, 1, 10, 5, 1, 5, 6, 3, 6, 10, 3, 3, 10, 8, 1, 9, 4, 6, 10, 4, 9, 6, 6, 5, 1, 4, 10, 4, 4, 7, 5, 1, 4, 7, 7, 8, 4, 4, 3, 7, 2, 4, 3, 7, 3, 9, 6, 2, 4, 1, 1, 6, 3, 3, 5, 7, 4, 9, 2, 10, 9, 10, 8, 2, 1, 10, 3, 4, 7, 1, 2, 9, 1, 8, 7, 4, 6, 6, 2, 4, 6, 4, 3, 6, 7, 6, 7, 9, 7, 10, 4, 5, 6, 9, 4, 2, 6, 7, 5, 9, 5, 7, 9, 6, 5, 2, 3, 10, 4, 3, 3, 9, 5, 7, 2, 4, 10, 1, 8, 3, 6, 4, 6, 2, 7, 10, 3, 5, 8, 9, 3, 5, 9, 7, 4, 10, 6, 7, 3, 4, 3, 8, 2, 5, 3, 5, 9, 5, 5, 6, 8, 1, 5, 9, 4, 3, 2, 9, 8, 5, 6, 3, 5, 9, 2, 2, 7, 1, 6, 9, 8, 5, 7, 9, 2, 3, 10, 1, 1, 7, 3, 6, 1, 1, 1, 4, 2, 10, 2, 1, 4, 7, 3, 2, 8, 3, 5, 8, 7, 1, 7, 6, 8, 1, 7, 2, 10, 8, 9, 10, 2, 1, 7, 9, 8, 4, 7, 9, 10, 8, 7, 4, 3, 5, 3, 7, 1, 10, 9, 2, 4, 9, 6, 1, 3, 2, 9, 2, 6, 5, 7, 1, 9, 10, 6, 9, 1, 10, 9, 6, 2, 2, 3, 6, 6, 4, 8, 3, 8, 7, 8, 2, 3, 1, 4, 2, 5, 6, 6, 5, 2, 3, 10, 6, 9, 6, 10, 4, 10, 4, 5, 7, 3, 6, 1, 1, 7, 2, 2, 10, 4, 3, 1, 10, 4, 5, 9, 6, 4, 8, 5, 2, 7, 2, 5, 3, 9, 3, 5, 3, 10, 2, 9, 2, 3, 5, 8, 10, 8, 6, 7, 9, 3, 6, 1, 5, 10, 1, 3, 3, 9, 8, 9, 1, 3, 7, 8, 3, 3, 10, 7, 10, 1, 2, 3, 5, 2, 9, 6, 6, 7, 2, 7, 4, 4, 3, 4, 8, 10, 8, 10, 3, 1, 8, 5, 6, 2, 8, 1, 8, 8, 10, 2, 7, 5, 1, 1, 2, 2, 9, 3, 4, 4, 10, 6, 3, 2, 7, 2, 3, 9, 4, 6, 4, 6, 8, 6, 3, 7, 10, 2, 5, 9, 4, 5, 4, 2, 4, 9, 6, 4, 10, 6, 5, 3, 10, 7, 2, 1, 4, 9, 8, 8, 6, 3, 1, 4, 7, 3, 6, 7, 9, 7, 9, 7, 2, 6, 6, 4, 1, 9, 3, 1, 2, 4, 7, 3, 10, 8, 7, 7, 1, 5, 10, 6, 4, 4, 9, 7, 2, 6, 2, 7, 7, 9, 3, 3, 1, 2, 4, 6, 7, 1, 6, 8, 6, 3, 9, 3, 9, 1, 10, 5, 9, 5, 2, 7, 3, 4, 10, 3, 6, 5, 1, 10, 5, 9, 10, 3, 2, 5, 3, 5, 9, 5, 3, 8, 6, 4, 7, 3, 2, 3, 9, 6, 6, 8, 3, 5, 4, 8, 3, 10, 3, 9, 7, 8, 2, 3, 8, 5, 5, 9, 5, 7, 1, 7, 5, 8, 2, 9, 9, 6, 5, 6, 1, 2, 4, 4, 3, 2, 7, 5, 7, 1, 7, 10, 4, 2, 2, 6, 2, 2, 5, 8, 6, 1, 10, 4, 4, 3, 2, 8, 4, 7, 10, 5, 2, 9, 4, 8, 3, 10, 1, 6, 1, 2, 6, 1, 10, 8, 10, 4, 1, 10, 4, 4, 1, 4, 9, 10, 4, 2, 2, 6, 2, 2, 5, 8, 6, 1, 10, 4, 4, 3, 2, 8, 4, 7, 7, 1, 3, 6, 7, 10, 1, 4, 4, 3, 5, 5, 8, 10, 4, 4, 8, 5, 3, 7, 2, 2, 6, 1, 8, 8, 9, 6, 3, 5, 8, 2, 3, 4, 10, 8, 1, 8, 7, 4, 3, 1, 9, 3, 4, 4, 10, 7, 3, 8, 10, 1, 9, 3, 5, 3, 9, 8, 5, 10, 7, 5, 5, 2, 4, 4, 10, 1, 10, 7, 10, 5, 6, 2, 10, 7, 9, 3, 5, 4, 3, 7, 1, 2, 5, 2, 1, 8, 1, 5

The file viewer window has tabs for "Archivo", "Editar", and "Ver". The status bar at the bottom indicates "Ln 1, Col 1" and "20978 caracteres".

¿Cómo iniciar? – Registros en QT

```
> main.cpp main(int, char **) -> int # CRLF Line: 3
1 #include <QCoreApplication>
2 #include<QTextStream>
3 #include<QDebug>
4 #include<QFile>
5 #include<QMap>
6 #include<ctime>
7
8 int main(int argc, char *argv[])
{
9     QApplication a(argc, argv);
10    QTextStream entradaSalida; //Stream de entrada y salida
11    QFile archivo; //Archivo
12    QMap<int, int>::Iterator iterar; //iterador del registro
13    QMap<int, int> registro; //Registro
14    //Vamos a registrar el numero que esta presente en el archivo y la cantidad de veces que el numero esta presente
15
16    int n, valor, suma=0;
17
18    srand(time(NULL));
19
20    archivo.setFileName("D:/DOCS/ASIGNATURAS IMPARTIDAS/ArcGIS SDK/Ejemplos/numeros.txt");
21    archivo.open(QIODevice::WriteOnly| QIODevice::Text);
22
23    entradaSalida.setDevice(&archivo); //Asigna el archivo al dispositivo de entrada y salida
24
25    for(int i=0; i<10000;i++)
26        |    entradaSalida << rand()%10+1 << ' ';
27        |    //generamos numeros aleatorios entre 1 y 10
28
29    archivo.flush();
30    archivo.close();
31
```

¿Cómo iniciar? – Registros en QT

```
archivo.flush();
archivo.close();

//Vamos a guardar el registro en el contenido
archivo.open(QIODevice::ReadOnly|QIODevice::Text);

while(!entradaSalida.atEnd()){
    entradaSalida>>n;
    registro[n]++; //observamos la cantidad de veces que esta presente
}

registro.erase(registro.find(0)); //En el archivo, al final de la linea, hay un espacio, ese espacio es tomado como 0,

for(iterar=registro.begin(); iterar != registro.end(); iterar++){
    valor = iterar.value();
    suma += valor;
    qDebug()<<iterar.key() << ":" << valor;
}

qDebug()<< "Suma "<< suma;
qDebug() << "Total "<< 10000;

return a.exec();
```

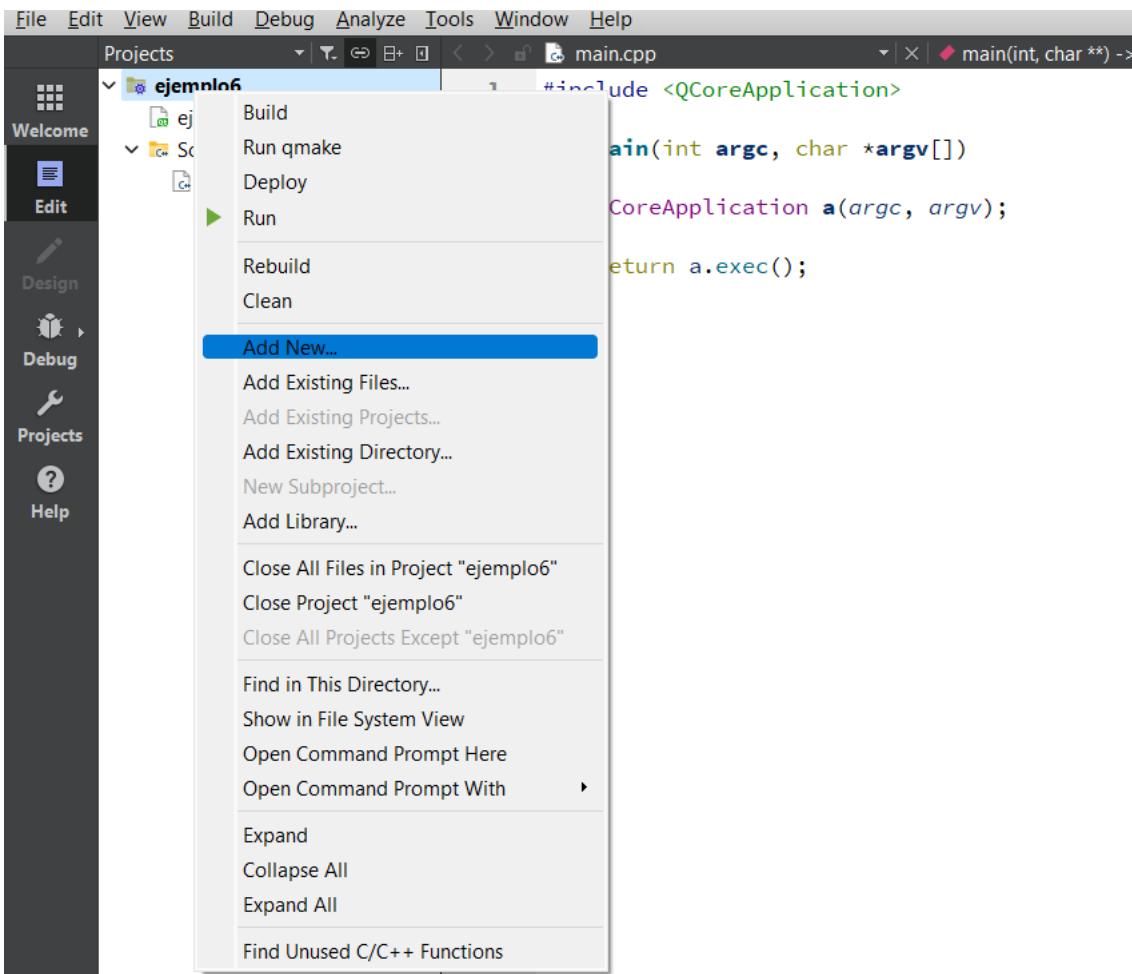
```
1 : 956
2 : 962
3 : 982
4 : 1050
5 : 1020
6 : 1018
7 : 1012
8 : 979
9 : 998
10 : 1023
Suma 10000
Total 10000
```

EJEMPLO 6

¿Cómo iniciar? – Signals y Slots en QT

- ❖ Los *signals* y *slots* son un mecanismo fundamental para conectar y comunicar componentes en una aplicación.
- ❖ Las *señales* son notificaciones emitidas por los *widgets* cuando ocurre algo. Por ejemplo, un botón envía una *señal* de “botón pulsado” cuando un usuario hace clic en él. En otras palabras, las *señales* son eventos que indican que algo ha sucedido en un widget.
- ❖ Los *receptores*, también conocidos como *slots*, son métodos o funciones que actúan como oyente para las señales. Cuando una *señal* se emite, los *slots* asociados a esa *señal* se activan. Por ejemplo, podemos conectar la *señal* “clic en un botón” a un método específico que se ejecutará cuando el botón sea pulsado.

¿Cómo iniciar? – Signals y Slots en QT

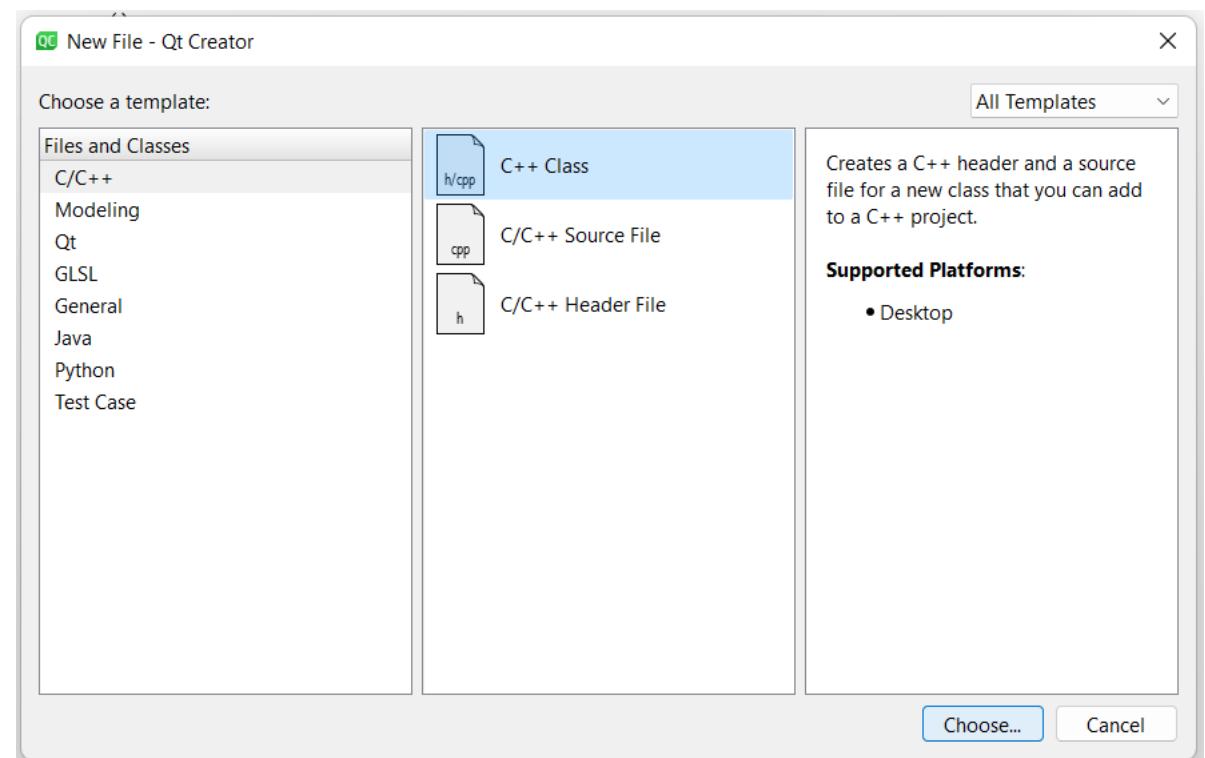


The screenshot shows the Qt Creator interface. On the left is the project navigation bar with tabs for Welcome, Edit, Design, Debug, Projects, and Help. The 'Projects' tab is selected, showing a project named 'ejemplo6'. The project tree shows files like 'ej' and 'So'. The main editor area contains the following C++ code:

```
#include <QCoreApplication>

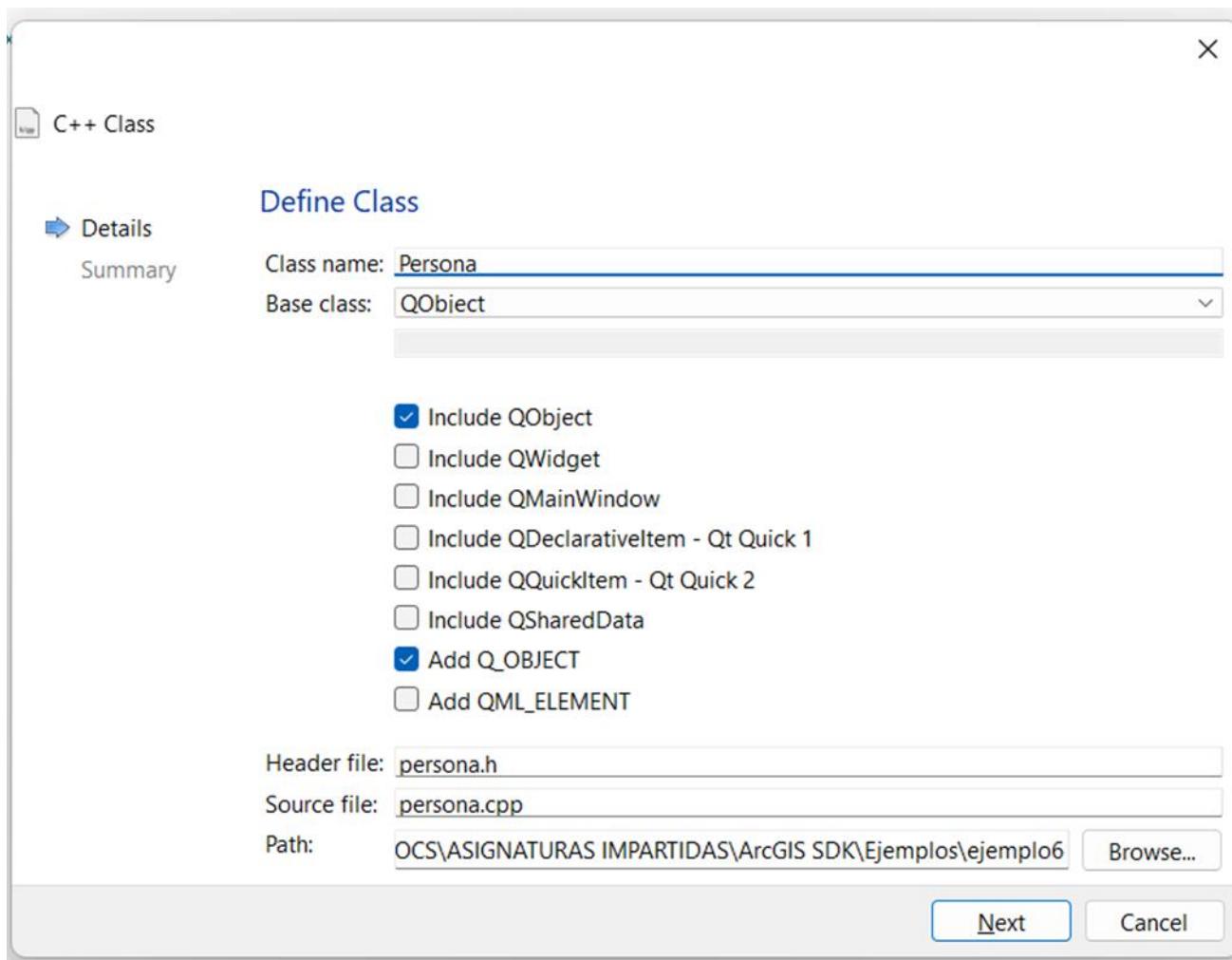
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    return a.exec();
}
```



Vamos agregar una clase dando clic derecho en el archivo de nuestro Proyecto. Seleccionamos *Add New ...* y luego *C++ Class*

¿Cómo iniciar? – Signals y Slots en QT



Vamos a crear una clase Persona, como clase base seleccionamos el Qobject y conservamos las opciones que se muestran en la imagen.

The screenshot shows the Qt Creator IDE with the file 'persona.cpp' open. The code defines a class 'Persona' that inherits from 'QObject'. The constructor is defined as follows:

```
#include "persona.h"
Persona::Persona(QObject *parent)
    : QObject{parent}
```

¿Cómo iniciar? – Signals y Slots en QT

The screenshot shows the Qt Creator IDE interface. On the left is the project navigation bar with icons for Welcome, Edit, Design, Debug, Projects, and Help. The main area shows a project named "ejemplo6" with files "ejemplo6.pro", "Headers/persona.h", and "Sources/main.cpp", "persona.cpp". The "persona.h" file is open in the code editor. The code is:

```
#ifndef PERSONA_H
#define PERSONA_H

#include <QObject>

class Persona : public QObject
{
    Q_OBJECT
public:
    explicit Persona(QObject *parent = nullptr);

signals:

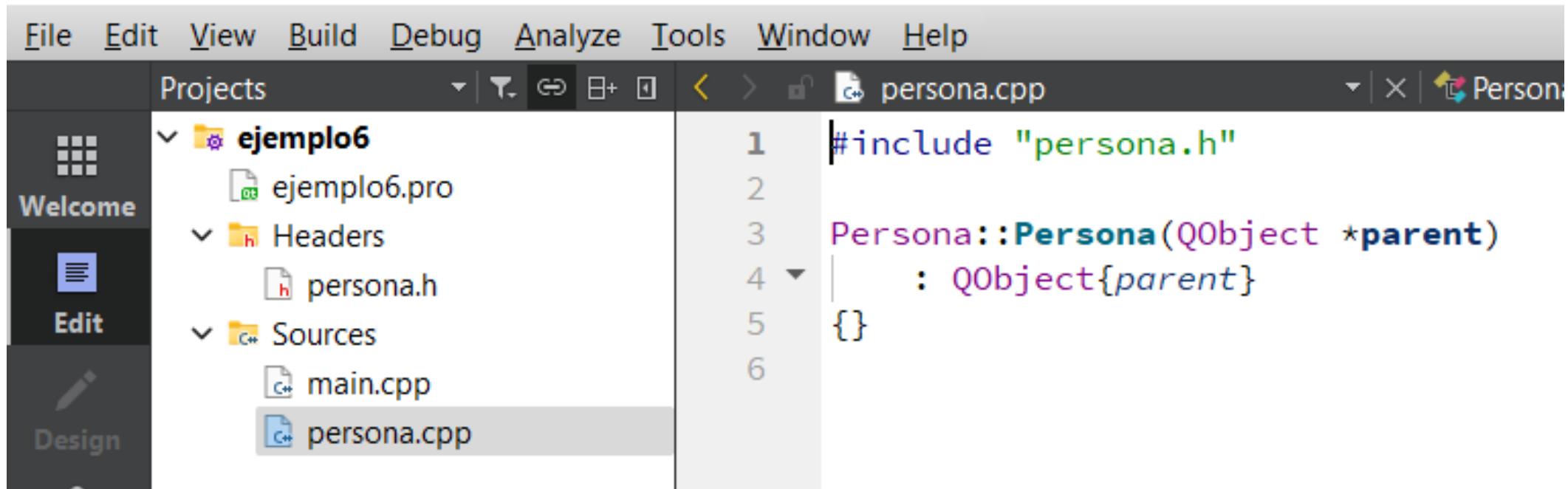
};

#endif // PERSONA_H
```

Annotations with arrows point to the "Q_OBJECT" macro and the "QObject" base class. The "Q_OBJECT" macro is circled with a blue oval, and an arrow points to it with the text "Macro". The "QObject" base class is circled with a blue oval, and an arrow points to it with the text "Heredada de QObject".

Persona al ser un objeto de QT acepta señales y receptores

¿Cómo iniciar? – Signals y Slots en QT



The screenshot shows a Qt-based IDE interface. The menu bar includes File, Edit, View, Build, Debug, Analyze, Tools, Window, and Help. The title bar shows "Projects" and the file "persona.cpp". The left sidebar has icons for Welcome, Edit, and Design. The Projects view shows a project named "ejemplo6" with files "ejemplo6.pro", "Headers/persona.h", "Sources/main.cpp", and "Sources/persona.cpp". The persona.cpp file is open in the main editor area, displaying the following code:

```
#include "persona.h"
Persona::Persona(QObject *parent)
: QObject{parent}
{}
```

Constructor implementado en el archivo
persona.cpp

¿Cómo iniciar? – Signals y Slots en QT



The screenshot shows a Qt IDE interface with the following details:

- Menu Bar:** View, Build, Debug, Analyze, Tools, Window, Help.
- Projects:** ejemplo6 (selected), ejemplo6.pro, Headers (selected), persona.h, Sources, main.cpp, persona.cpp.
- Editor Area:** Shows the content of the persona.h file.

```
1 #ifndef PERSONA_H
2 #define PERSONA_H
3
4 #include <QObject>
5
6 class Persona : public QObject
7 {
8     Q_OBJECT //nos permite tener signals y slots
9 public:
10     explicit Persona(QObject *parent = nullptr);
11
12     //Métodos
13     void setNombre(const QString &nombre){
14         m_nombre = nombre;
15     }
16
17     void habla(const QString &palabras);
18
19 signals: //Las señales siempre inician con una función void
20
21     void hablo(QString); //Una persona cuando hable va a emitir una señal
22
23 public slots:
24
25     void escucha (const QString &palabras); //este metodo para recepcion de señales se implementa en el archivo persona.cpp
26
27 private:
28     QString m_nombre; //nombre de la persona
29 };
30
31 #endif // PERSONA_H
```

En el archivo header (.h) creamos los métodos para nuestro *signals* y *slots*

¿Cómo iniciar? – Signals y Slots en QT

The screenshot shows a Qt IDE interface with the following details:

- Menu Bar:** File, Edit, View, Build, Debug, Analyze, Tools, Window, Help.
- Toolbar:** Projects, Welcome, Edit, Design, Debug, Projects, Help.
- Project Explorer:** Shows a project named "ejemplo6" with files "ejemplo6.pro", "Headers/persona.h", and "Sources/main.cpp", "persona.cpp".
- Code Editor:** The file "persona.cpp" is open, showing C++ code implementing signals and slots for the "Persona" class.
- Code Content:**

```
1 #include "persona.h"
2 #include<QDebug>
3
4 Persona::Persona(QObject *parent)
5     : QObject{parent}
6
7 void Persona::escucha(const QString &palabras){
8     qDebug()<< m_nombre << "ha escuchado: "<< palabras;
9 }
10
11 void Persona::habla(const QString &palabras){
12     emit hablo(palabras);
13 }
14
15 }
```
- Status Bar:** Shows the signal "Person::habla(const QString &) -> void".

Implementamos los métodos creados en el archivo *header* en el archivo *persona*

¿Cómo iniciar? – Signals y Slots en QT

ip @ ejemplo6 - Qt Creator

View Build Debug Analyze Tools Window Help

Projects ejemplo6 main.cpp main(int, char **) -> int

#include <QCoreApplication>
#include "persona.h"

int main(int argc, char *argv[]){
 QCoreApplication a(argc, argv);

 QObject *ElPadre = new QObject;

 Persona *Miguel = new Persona(ElPadre); //hijo del objeto padre
 Persona *David = new Persona(ElPadre); //hijo del objeto padre

 QObject::connect(Miguel, SIGNAL(habla(QString)), David, SLOT(escucha(QString)));

 Miguel -> habla("Que tal David");

 delete ElPadre;

 return a.exec();
}

Application Output | Filter + -

ejemplo6 x
11:38:03: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo6-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo6.exe...
"" ha escuchado: "Que tal David"

¿Cómo iniciar? – Signals y Slots en QT

The screenshot shows a Qt IDE interface with the following components:

- Project Explorer:** Shows a project named "ejemplo6" with files "ejemplo6.pro", "Headers/persona.h", "Sources/main.cpp", and "Sources/persona.cpp".
- Code Editor:** Displays the content of main.cpp. The code demonstrates the use of Signals and Slots. It includes includes for QCoreApplication and persona.h, defines the main function, creates a QCoreApplication object 'a', and initializes two Persona objects 'Miguel' and 'David'. It connects the SIGNAL(habla(QString)) of Miguel to the SLOT(escucha(QString)) of David. Finally, it calls a.exec() to run the application.
- Application Output:** Shows the following log:

```
ejemplo6 x
"" ha escuchado: "Que tal David"
11:40:48: D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo6-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo6.exe crashed.

11:40:50: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo6-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo6.exe...
"David" ha escuchado: "Que tal David"
```

¿Cómo iniciar? – Signals y Slots en QT

```
1 #include <QCoreApplication>
2 #include "persona.h"
3
4 int main(int argc, char *argv[])
5 {
6     QCoreApplication a(argc, argv);
7
8     QObject *ElPadre = new QObject;
9
10    Persona *Miguel = new Persona(ElPadre); //hijo del objeto padre
11    Persona *David = new Persona(ElPadre); //hijo del objeto padre
12
13    Miguel->setNombre("Miguel");
14    David->setNombre("David");
15
16    QObject::connect(Miguel, SIGNAL(habla(QString)), David, SLOT(escucha(QString))); //conexiones
17    QObject::connect(David, SIGNAL(habla(QString)), Miguel, SLOT(escucha(QString))); //conexiones
18
19    Miguel -> habla("Que tal David");
20
21    David -> habla("Bien y vos");
22
23    delete ElPadre;
24
25    return a.exec();
26 }
27
```



The screenshot shows a terminal window titled 'Application Output' with the process name 'ejemplo6'. The window displays the following text:

```
ejemplo6
11:44:25: D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo6-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo6.exe crashed.

11:44:27: Starting D:\DOCS\ASIGNATURAS IMPARTIDAS\ArcGIS SDK\Ejemplos\build-ejemplo6-Desktop_Qt_6_5_3_MSVC2019_64bit-Debug\debug\ejemplo6.exe...
"David" ha escuchado: "Que tal David"
"Miguel" ha escuchado: "Bien y vos"
```