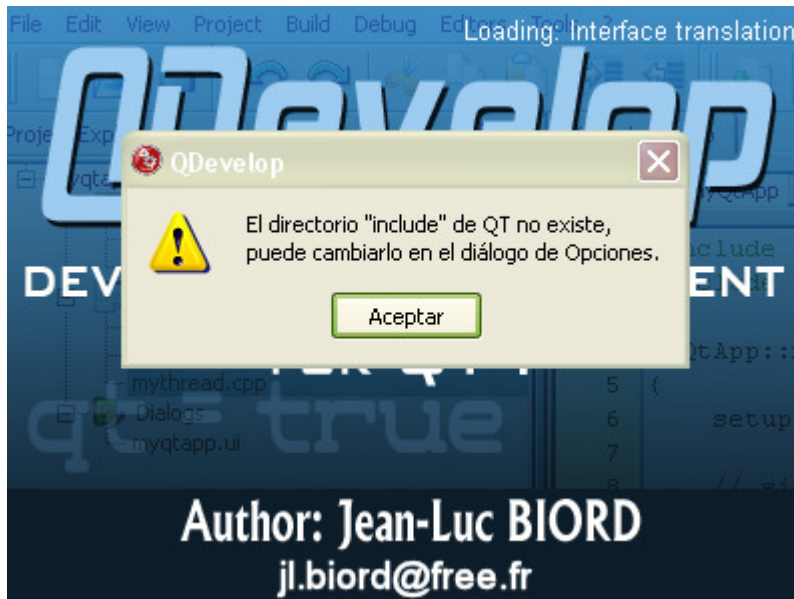


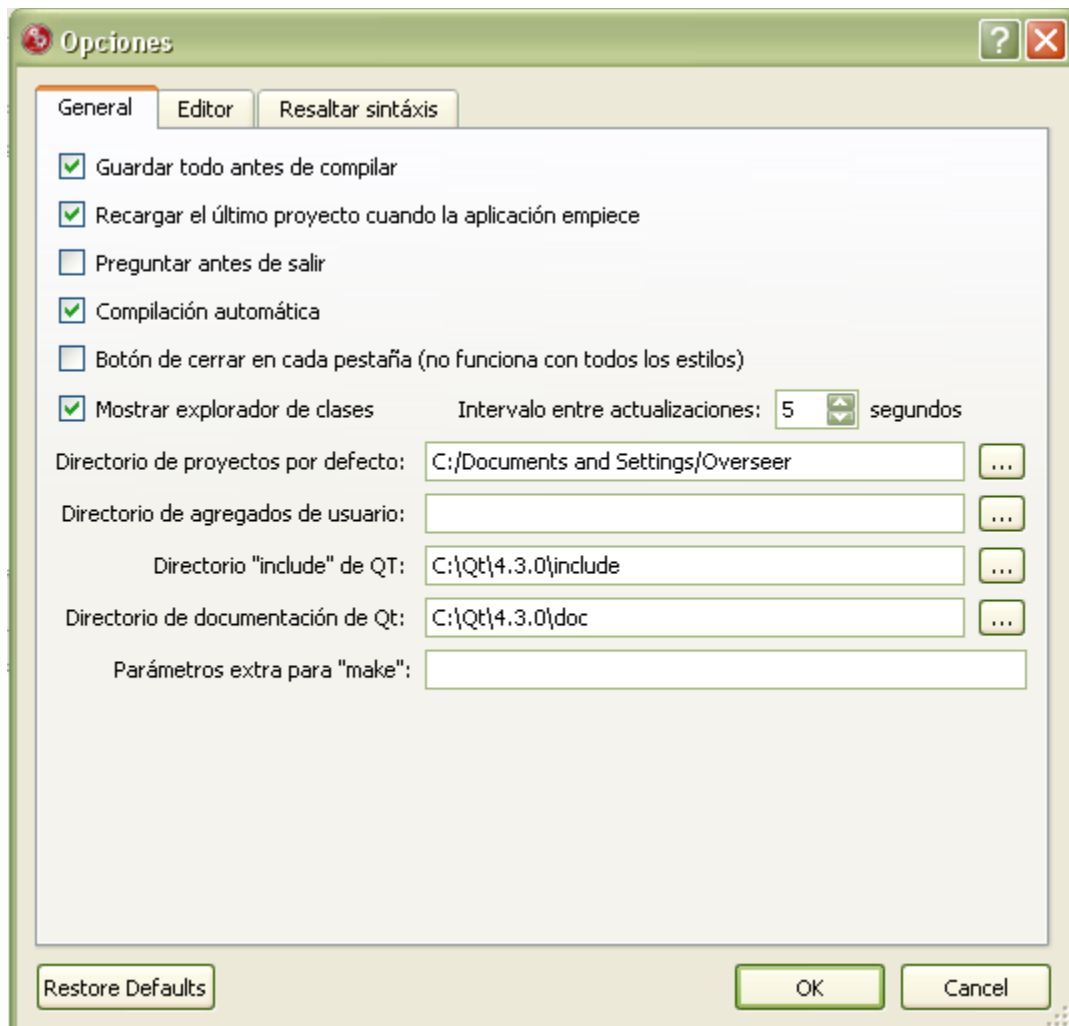
## INSTALAR QT 4.4.0 EN TU PC PARA TRABAJAR CON QDevelop en XP

La siguiente información le permitirá instalar Qt 4.4.0 (última versión a la fecha, 25/julio/2008) y poderlo manejar con el programa de distribución libre QDevelop. **NOTA:** Es importante que lea cada punto completamente antes de realizar la acción correspondiente.

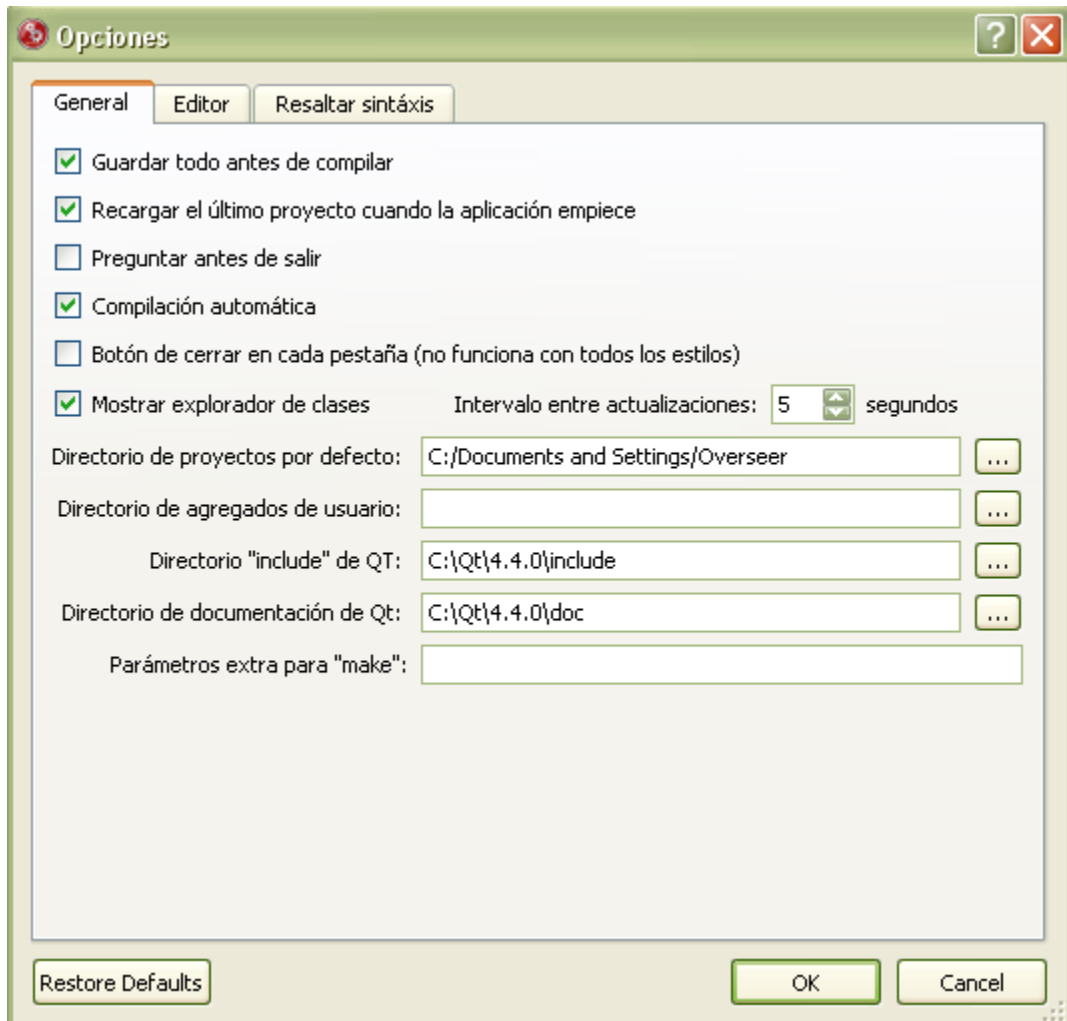
1. Descargue Qt de la página de TROLLTECH; será mejor que descargue la versión que ya viene con WinGW <http://ftp.ntua.gr/pub/X11/Qt/qt/source/qt-win-opensource-4.4.0-mingw.exe>, instálelo y esta a su vez le pedirá instalar también WinGW.
2. Descargue los siguientes programas e instálelos (aun no abra QDevelop porque es necesario configurar otros elementos primero, sólo instálelo):
  - QDevelop -> <http://qdevelop.free.fr/download/index.php3>
  - GDB -> [http://sourceforge.net/project/downloading.php?groupname=mingw&filename=gdb-6.3-2.exe&use\\_mirror=ufpr](http://sourceforge.net/project/downloading.php?groupname=mingw&filename=gdb-6.3-2.exe&use_mirror=ufpr)  
Cuando lo instale este le pedirá que la ubicación de instalación sea C:\MinGW, acepte.
  - CTAGS -> <http://prdownloads.sourceforge.net/ctags/ec554w32.zip?download>  
Este es un archive .zip, descomprímalo y la carpeta que está adentro (ctags554), déjela en la ubicación de C:\ctags554 (C:\ es por defecto en muchos PC el disco duro, si tiene otro, simplemente colóquelo en el suyo, eg. F:\ctags554).
3. Ahora vaya a MiPC botón derecho ->propiedades->opciones avanzadas->variables de entorno, y en el 'path' incluya los siguientes parámetros:
  - C:\MinGW\bin;C:\Qt\4.4.0\bin;C:\ctags554;
4. Es momento de la compilación de Qt, este proceso es largo y varía de acuerdo a las especificaciones de la máquina en la cual se genere, vaya a Inicio->Todos los programas-> Qt by Trolltech v4.4.0 (OpenSource)-> Qt 4.4.0 (Build Debug Libraries) a continuación se abrirá un ventana del símbolo del sistema y empezará a aparecer la compilación de Qt; como ya dije, esto puede tomar mucho tiempo (aprox 2 horas ó 2 horas y media). **IMPORTANTE:** Si lo desea, puede configurar primero Qt y después realizar la compilación de Qt; en mi caso no lo hice y solamente lo mande compilar y ya, pero si desea configurarlo primero, entonces abra un símbolo del sistema (Inicio->Accesorios->Símbolo del sistema) vaya a la ubicación donde instalo Qt (en mi caso era C:\Qt\4.4.0), luego copie **configure -help**, aparecerá una lista con todas las posibles opciones de configuración. Después de leerlas y decidirse cuales configurar, entonces cópielas después de la palabra **configure**, por ejemplo: **configure -fast -qt-gif -no-dsp -saveconfig** es importante no olvidar la última parte **-saveconfig** puesto que ésta guardara la configuración deseada. Ahora si puede ir a Inicio->Todos los programas-> Qt by Trolltech v4.4.0 (OpenSource)-> Qt 4.4.0 (Build Debug Libraries) y compilar la librería de Qt.
5. Bueno, ya tenemos compilada la librería e instalados todos los programas; ahora, abra QDevelop, este mostrará el siguiente mensaje:



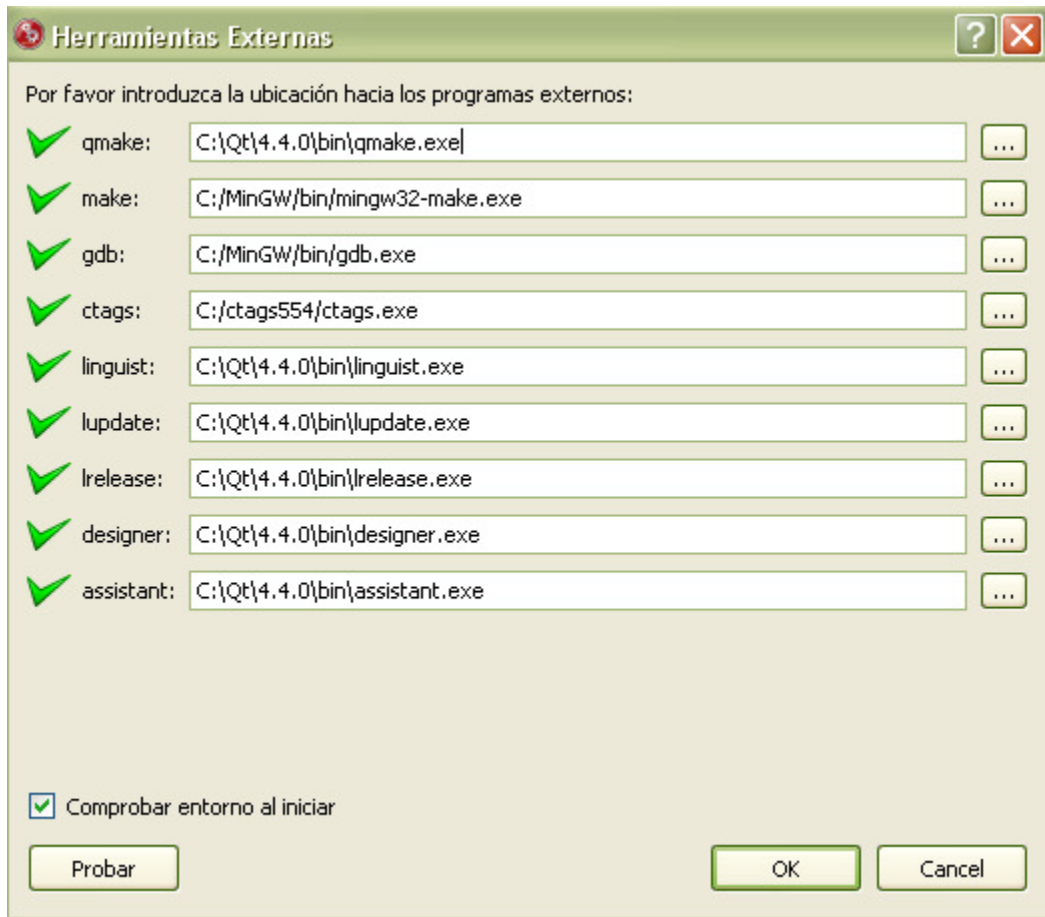
No importa, acepte, luego vaya a herramientas->opciones:





Como puede observar, en la parte que dice **Directorio "include" de QT**: dicho directorio no existe, así que lo cambiamos por el nuestro: C:\Qt\4.4.0\include:

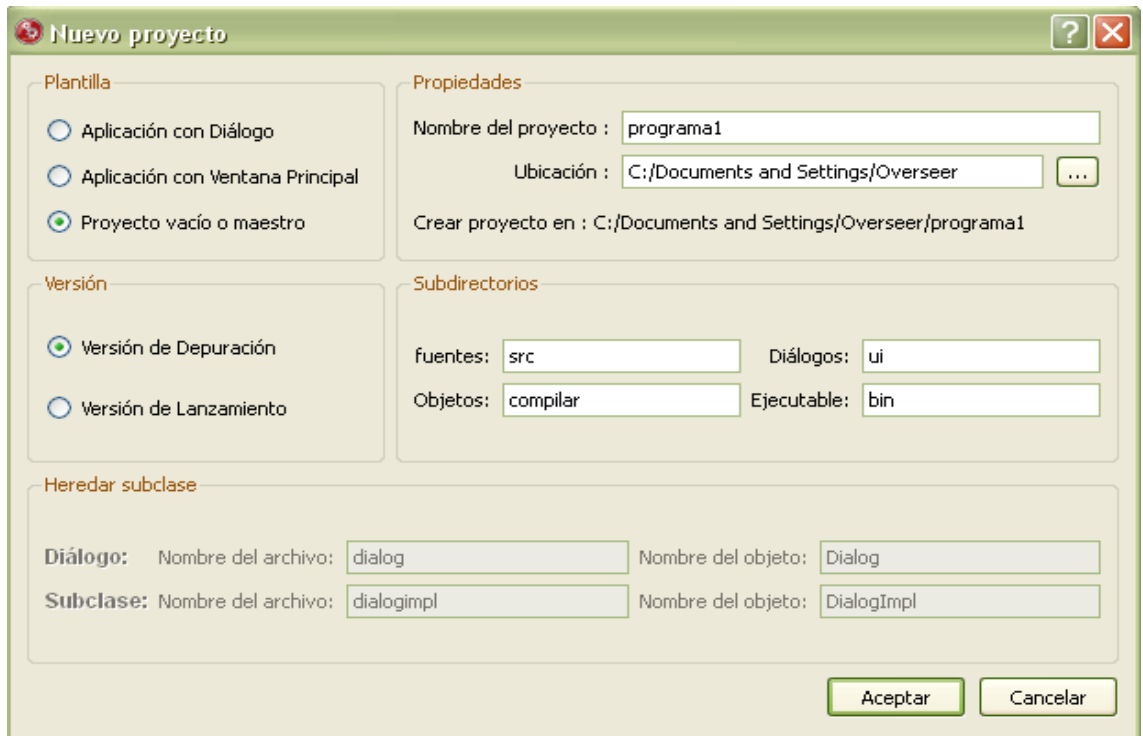


Oprima el botón OK, ahora vaya a Herramientas->Herramientas Externas... y configúrelo de acuerdo a la siguiente imagen:

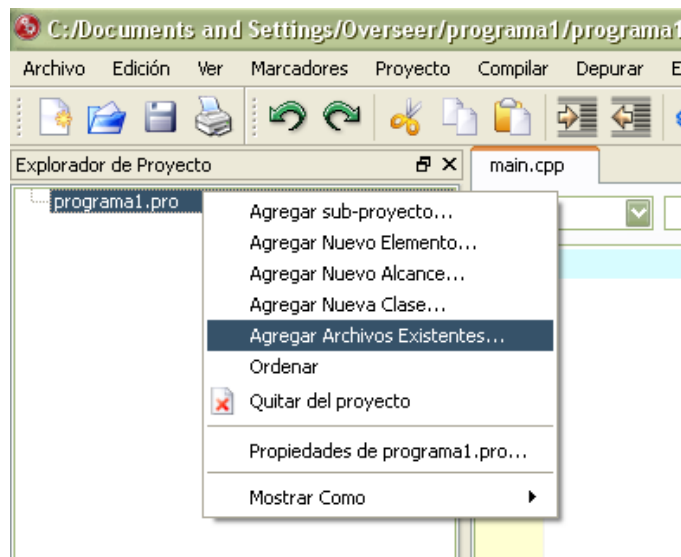


Puede que cuando sea la primera vez que configura esta parte, aparezcan unas  en vez de los , para cambiarlos introduzca las debidas direcciones en cada espacio y luego oprima el botón **Probar** y de esta manera irán cambiando; es importante que TODAS estén aprobadas para que funcione correctamente, luego oprima el botón OK.

6. Listo, ya tiene QDevelop listo para ser usado. A continuación mostraremos un ejemplo para comprobar que Qt fue compilado debidamente y funciona como debe ser. Vaya a Proyecto->Nuevo Proyecto... y configure las opciones de acuerdo a la siguiente imagen:



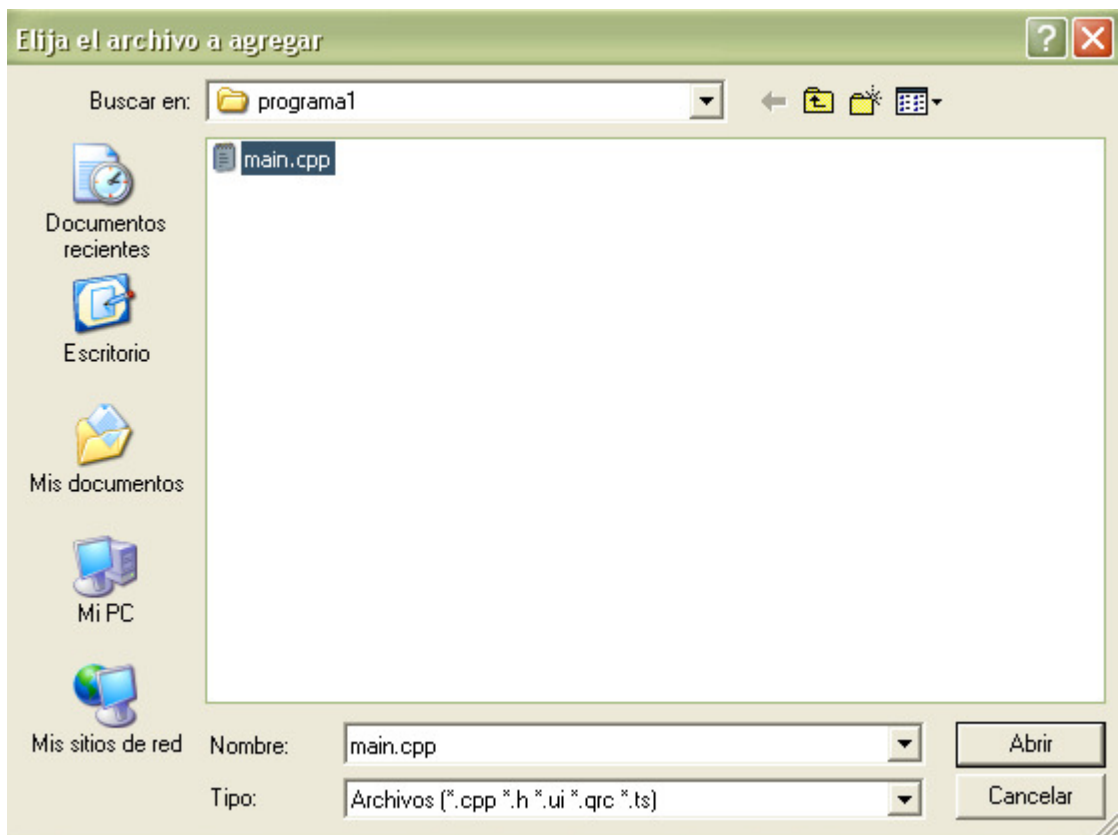
Se coloco en Versión Depuración puesto que yo siempre trabajo Depurando mis programas, si no lo desea, puede dejar la versión en Lanzamiento. Oprima el botón Aceptar. Observe que en la parte de la izquierda en el **Explorador de Proyecto**, aparece programa1.pro. Ahora vaya a Archivo->Nuevo Archivo... y en la dirección donde creó el proyecto (en mi caso es C:\Documents and Settings\Overseer\programa1) cree un archivo llamado **main.cpp** luego oprima el botón GUARDAR. Este último archivo creado existe dentro del proyecto, pero no está agregado al mismo, para eso vamos al **Explorador de Proyecto** (ventana de la izquierda) y damos click derecho sobre programa1.pro:



Aparecerá un menú desplegable como el de la imagen, luego vamos a **Agregar Archivos Existentes...**



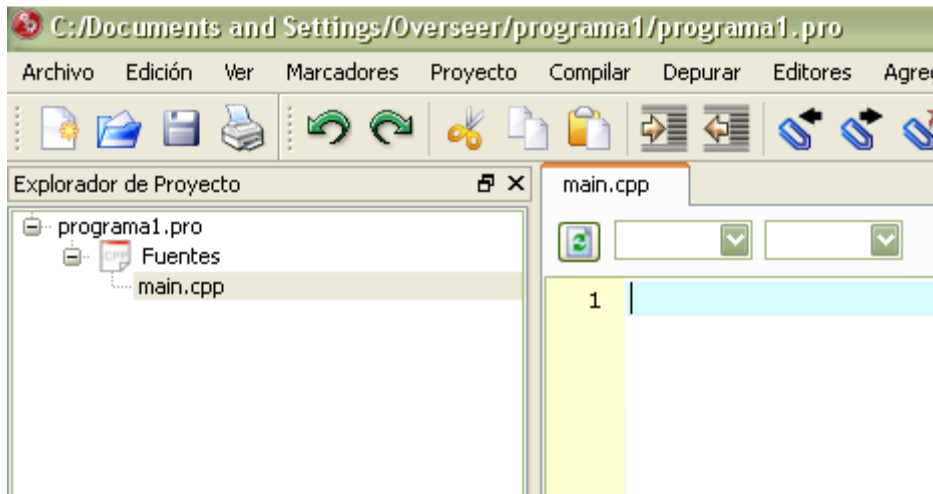
Aparecera el menú de la imagen anterior, damos click en el botón con los puntos ...



Seleccionamos el archivo **main.cpp** y lo abrimos




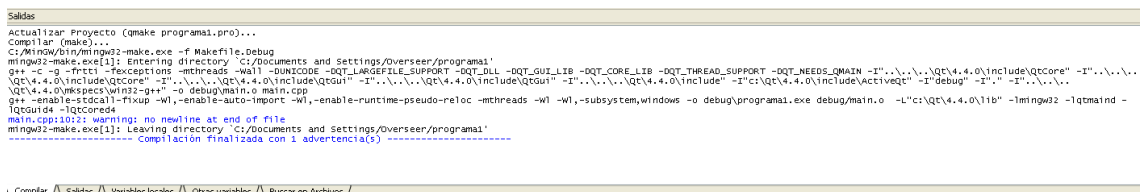
Aceptamos y después aparecerá en el Explorador de Proyecto (ventana de la izquierda) así:




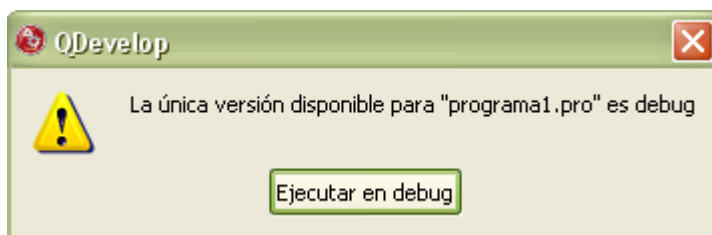
Ahora dentro del main.cpp copiamos el siguiente código:

```
#include <QApplication>
#include <QLabel>
int main( int argc, char *argv[] )
{
    QApplication app( argc, argv );
    QLabel *label = new QLabel( "Primer Programa" );
    label->show();
    return app.exec();
}
```

Luego Oprimir F7 (compilar) o selecciona el botón ; entonces en la parte de abajo aparecerá la compilación:



Si no hay problema todo debe aparecer en azul, si algo sale en rojo, compruebe si está bien copiado el código. Ahora oprima Shift+F5 ó el botón , aparecerá un mensaje



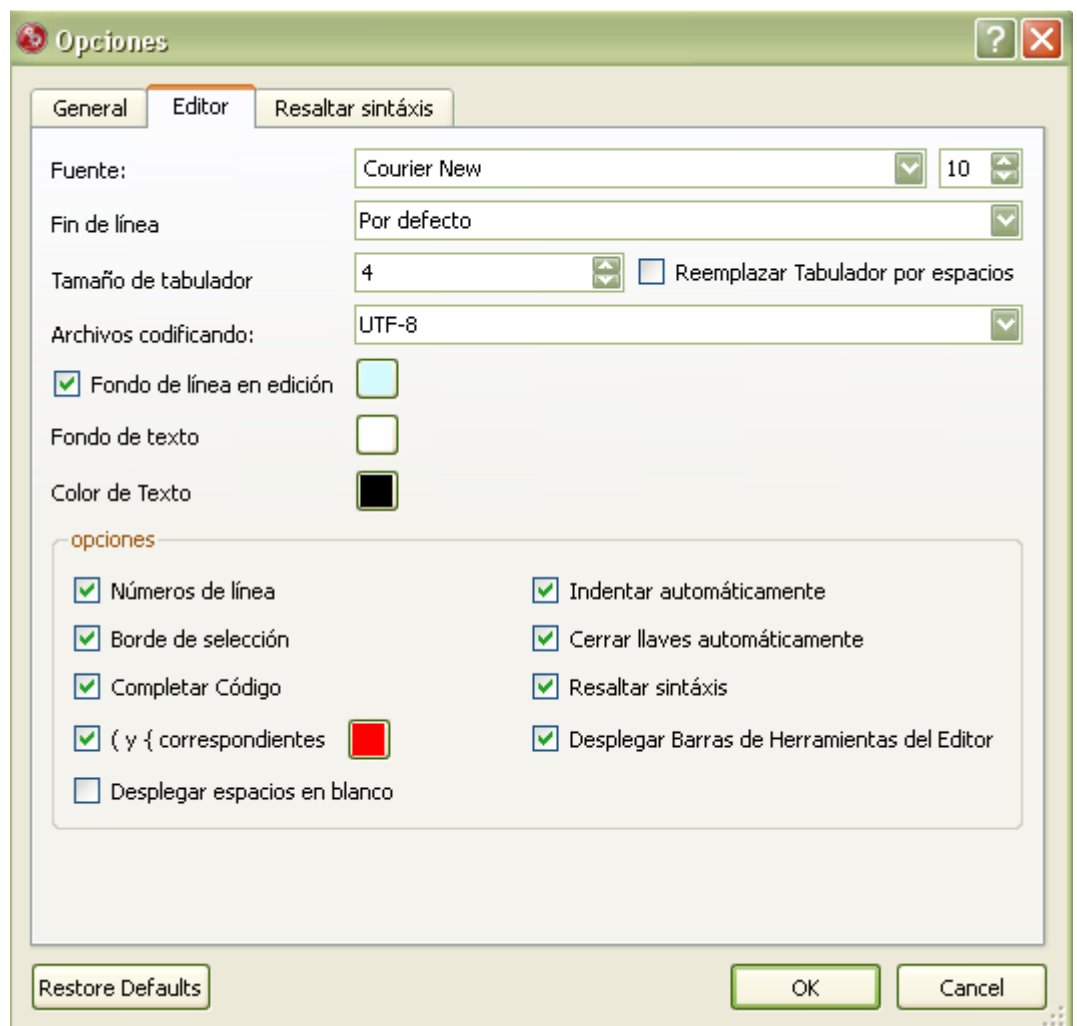
Oprima el botón **Ejecutar en debug** (puesto que esta fue la versión que elegimos cuando creamos el proyecto) y luego aparecerá la pequeña ventana de **programa1**



LISTO TODO SALIO BIEN Y PODEMOS YA TRABAJAR CON QT!!!

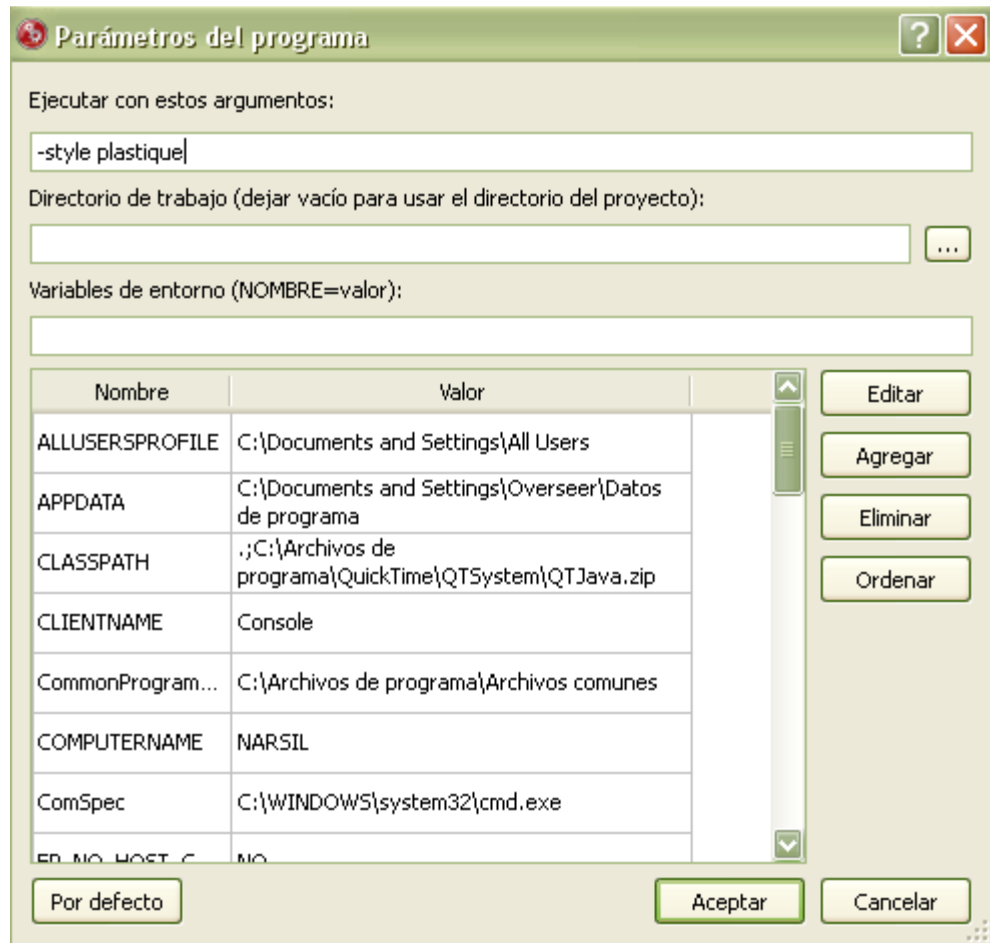
## 7. CONFIGURANDO QDevelop


- No sé si se dieron cuenta pero cuando escriben el código por primera vez en QDevelop, aparecen unos puntos grises casi blancos, para quitarlos entonces ir a Herramientas->Opciones... luego seleccionamos la pestaña de Editor y quitamos la selección del cuadro que dice **Desplegar espacios en blanco**



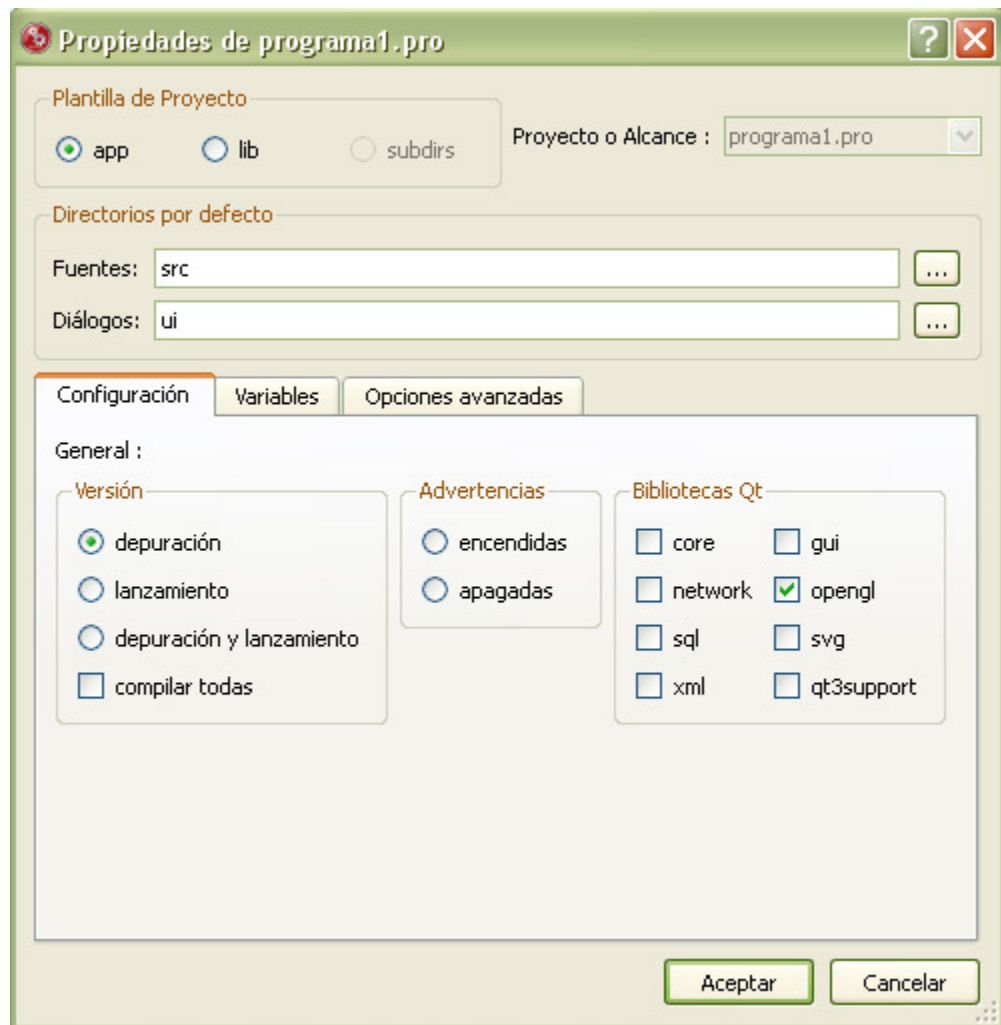
En esta parte pueden configurar más cosas si lo desean.

- Si de pronto ya han trabajado con Qt antes, saben que se pueden mostrar los programas ejecutados en diferentes versiones, aquí vamos a mostrar una: *plastique*, para colocarla, vamos a Depurar->Parámetros... Y en la parte que dice **Ejecutar con estos argumentos** escribir **-style plastique**

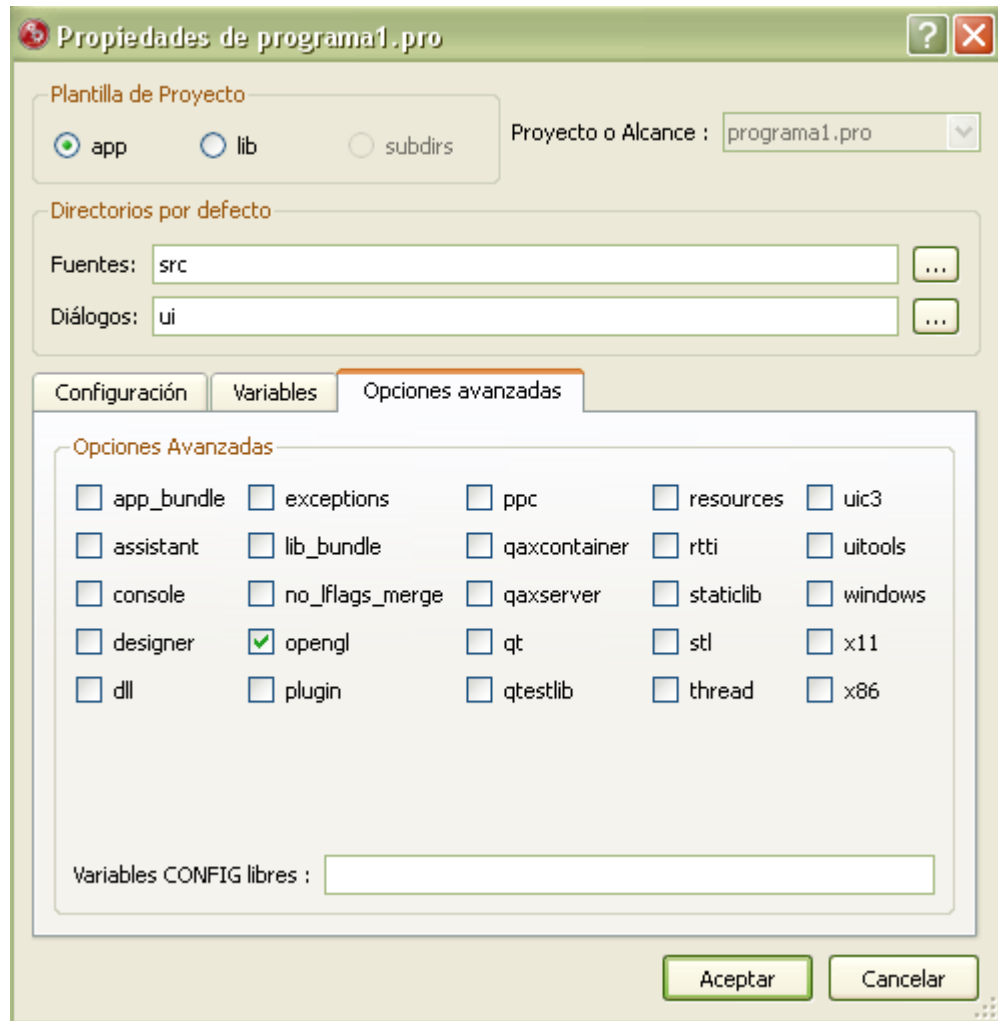


Luego **ACEPTAR** y oprimimos Shift+F5 ó el botón  y listo. **NOTA:** Con nuestro primer ejemplo, no podemos notar verdaderamente la diferencia, pero si creamos otros objetos como por ejemplo QPushButton si podríamos notar la diferencia.

- En el caso que trabajen con OpenGL, entonces deben configurar QDevelop para dicha librería, ir a Proyecto->Propiedades de programa1.pro... y hacemos click en el cuadro que dice opengl dentro de **Bibliotecas Qt** en la pestaña de **Configuración**



Luego también en la pestaña de **Opciones avanzadas** hacemos lo mismo:



## 8. DEPURANDO EN QDevelop

Supongamos que en nuestro programa queremos mostrar un Label con los números del 1 al 10 (un programa realmente muy sencillo), almacenaremos el valor de los números en la variable **cont**, ahora supongamos que necesitamos saber el comportamiento de esta variable durante la ejecución del programa, para eso usamos los **Puntos de Ruptura** (o en inglés **BreakPoint**). Bueno, el código es el siguiente:

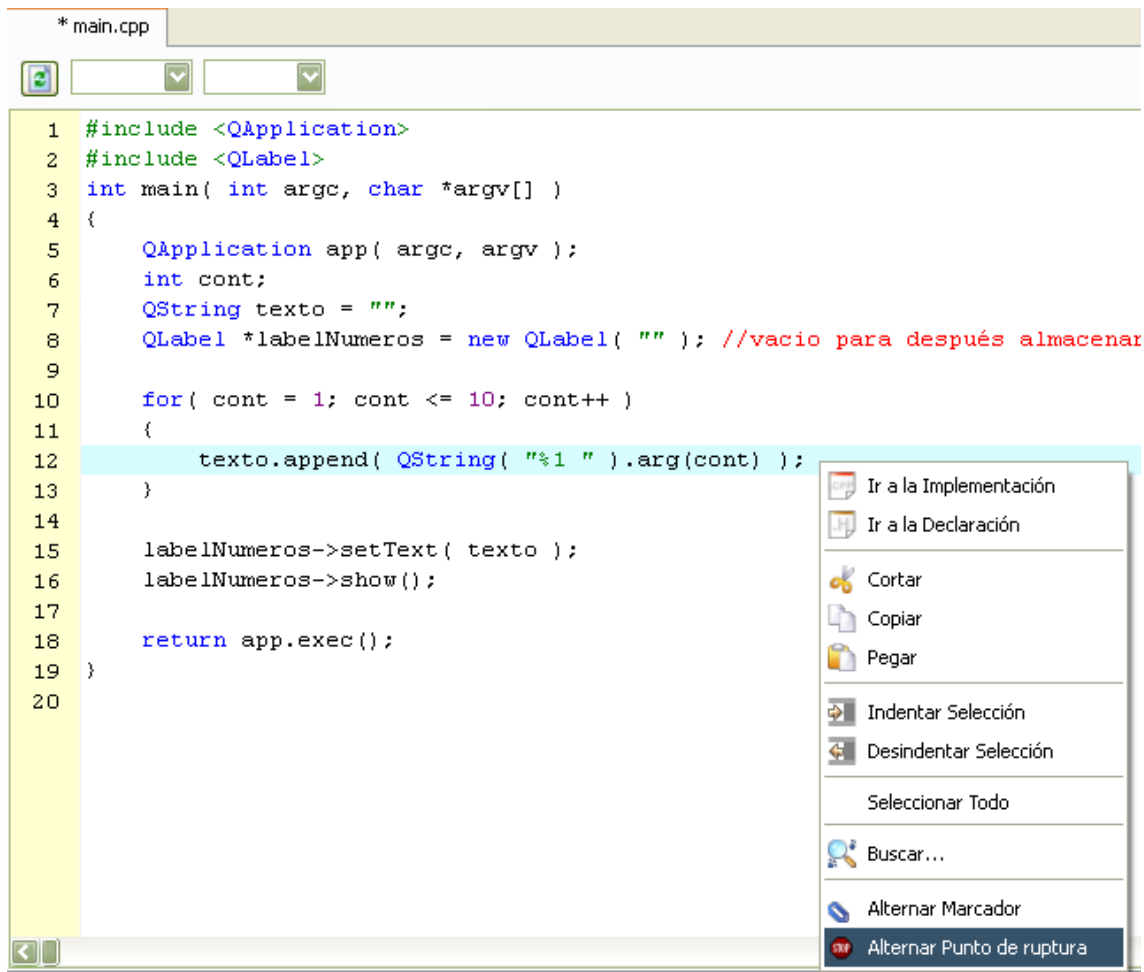
```
#include <QApplication>
#include <QLabel>
int main( int argc, char *argv[] )
{
    QApplication app( argc, argv );
    int cont;
    QString texto = "";
    QLabel *labelNumeros = new QLabel( "" ); //vacío para guardar números

    for( cont = 1; cont <= 10; cont++ )
    {
        texto.append( QString( "%1 " ).arg(cont) );
    }

    labelNumeros->setText( texto );
    labelNumeros->show();

    return app.exec();
}
```

```
*main.cpp
1 #include <QApplication>
2 #include <QLabel>
3 int main( int argc, char *argv[] )
4 {
5     QApplication app( argc, argv );
6     int cont;
7     QString texto = "";
8     QLabel *labelNumeros = new QLabel( "" ); //vacío para después almacenar
9
10    for( cont = 1; cont <= 10; cont++ )
11    {
12        texto.append( QString( "%1 " ).arg(cont) );
13    }
14
15    labelNumeros->setText( texto );
16    labelNumeros->show();
17
18    return app.exec();
19 }
20
```



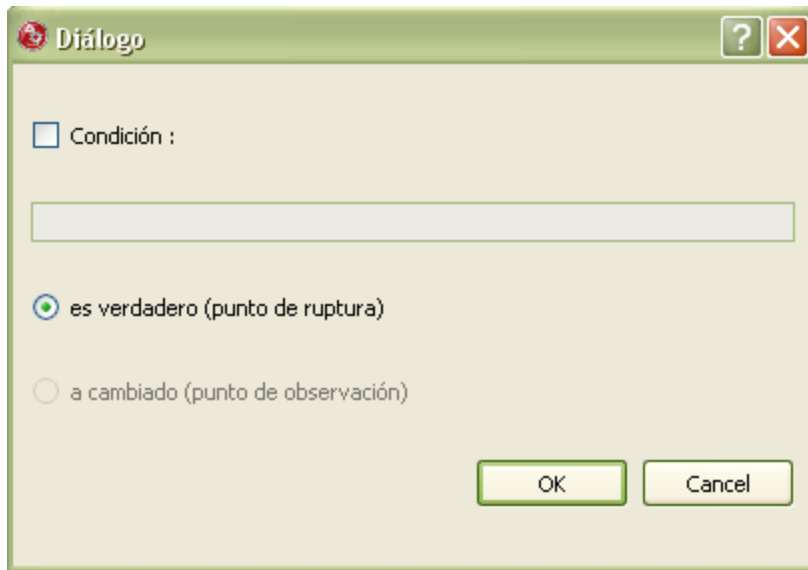
En la línea 12 hacemos click derecho y en el menú que aparece, en la parte de abajo, seleccionamos **Alternar Punto de Ruptura** apareciendo de la siguiente forma:

```
* main.cpp
1 #include <QApplication>
2 #include <QLabel>
3 int main( int argc, char *argv[] )
4 {
5     QApplication app( argc, argv );
6     int cont;
7     QString texto = "";
8     QLabel *labelNumeros = new QLabel( "" ); //vacio para
9
10    for( cont = 1; cont <= 10; cont++ )
11    {
12        texto.append( QString( "%1 " ).arg(cont) );
13    }
14
15    labelNumeros->setText( texto );
16    labelNumeros->show();
17
18    return app.exec();
19 }
20
```

A continuación hacemos click derecho sobre el icono del punto de ruptura (el de STOP), desplegando el siguiente menú:

```
11
12 texto.append( QString( "%1 " ).arg(cont) );
13
14     ext( texto );
15     ();
16
17
18     return app.exec();
19 }
20
```

Seleccionamos **Condición de Punto de Ruptura** aparece el siguiente menú:



Seleccionamos **es verdadero (punto de ruptura)** y luego damos OK; ahora hacemos shift+F5 y después que empiece la depuración se verá de la siguiente forma:

```

12 >> for( cont = 1; cont <= 10; cont++ )
13 >> {
14 >>     texto.append( QString( "%1 " ).arg(cont) );
15 >> }
16 >>
17 >> labelNumeros->setText( texto );
18 >> labelNumeros->show();
19 >>

```

Ahora es preciso mirar en la parte de debajo de QDevelop, **Salidas** donde hay varias pestañas; seleccionamos la que dice **Variables locales**, allí aparecerán algunas de las variables locales que tiene nuestro programa:

Salidas			
Nombre	Tipo	Dirección	Valor
argc	int	(int *) 0x22fe10	1 □
argv	char **	(char **) 0x3d40a0	(char **) 0x3d40a0 □
app	QApplication	(QApplication *) 0x22fde0	{<QCoreApplication...
cont	int	(int *) 0x22fddc	1 □
texto	QString	(QString *) 0x22fdc0	
labelNumeros	class QLabel *	(QLabel *) 0x48518f0	(QLabel *) 0x48518f0 □

Compilar Salidas Variables locales Otras variables Buscar en Archivos

Vemos que **cont** tiene el valor de 1, ahora presionamos la tecla F10 (ver el Menu Depurar para más opciones) y lo volvemos a oprimir hasta que la variable cambie a medida que va recorriendo todo el **for**. Ahora después de varias veces de oprimir F10 podemos ver como la variable **text** va guardando los valores:

Salidas			
Nombre	Tipo	Dirección	Valor
argc	int	(int *) 0x22fe10	1 □
argv	char **	(char **) 0x3d40a0	(char **) 0x3d40a0 □
app	QApplication	(QApplication *) 0x22fde0	{<QCoreApplicat...
cont	int	(int *) 0x22fddc	6 □
texto	QString	(QString *) 0x22fdc0	1 2 3 4 5
labelNumeros	QLabel *	(QLabel *) 0x48518f0	(QLabel *) 0x48518f0 □

[Compilar](#)
[Salidas](#)
[Variables locales](#)
[Otras variables](#)
[Buscar en Archivos](#)

Listo, ya pudimos ver el valor de las variables mientras depurábamos el programa, si se quiere ver el resultado del programa, sólo hay que quitar el Punto de Ruptura en la línea 12, hacer click derecho y presionar en el menú desplegable **Alternar Punto de Ruptura**. Y Ahora presionar Shift+F5 para detener la depuración y luego volver a oprimir Shift+F5 para correr el programa. La salida es:

