UNIVERSIDAD DE OVIEDO
Departamento de Ingeniería Eléctrica,
Electrónica, de Computadores y Sistemas

Instalación y uso de QtCreator para programación en lenguaje C (modo consola)

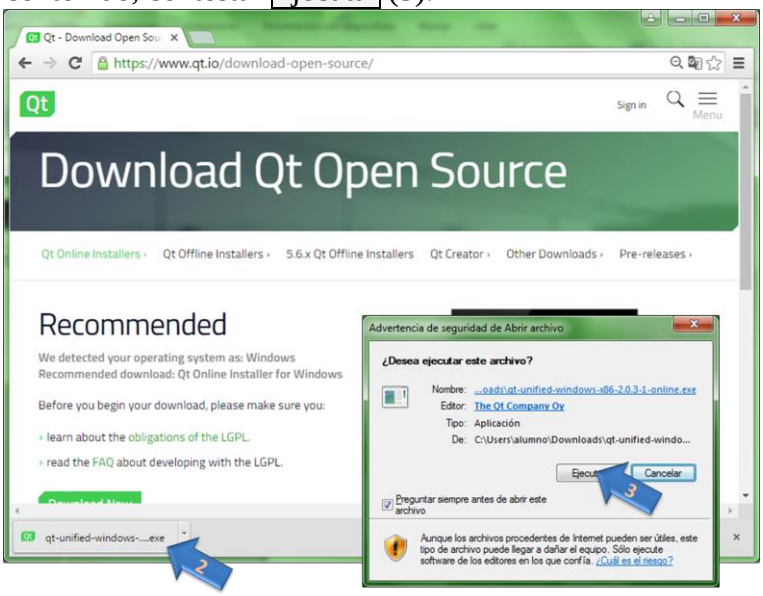
Ignacio Alvarez García – Septiembre 2016

INDICE

1. Instalación

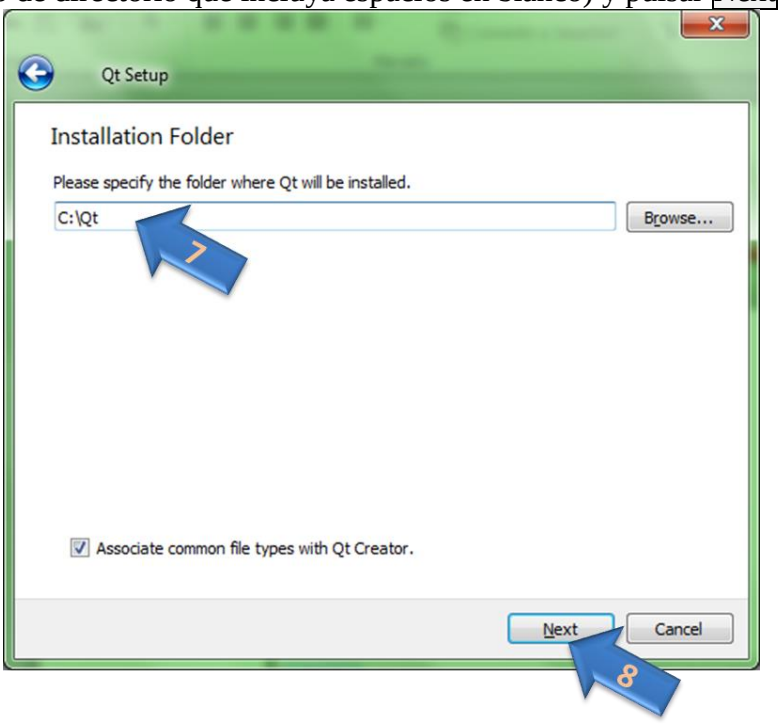
El instalador de descarga de la página https://www.qt.io/download-open-source , presionando el botón Download Now (1).

Una vez finalizada la descarga, abrir el archivo obtenido (2). Windows preguntará si se desea ejecutar el contenido, contestar Ejecutar (3).



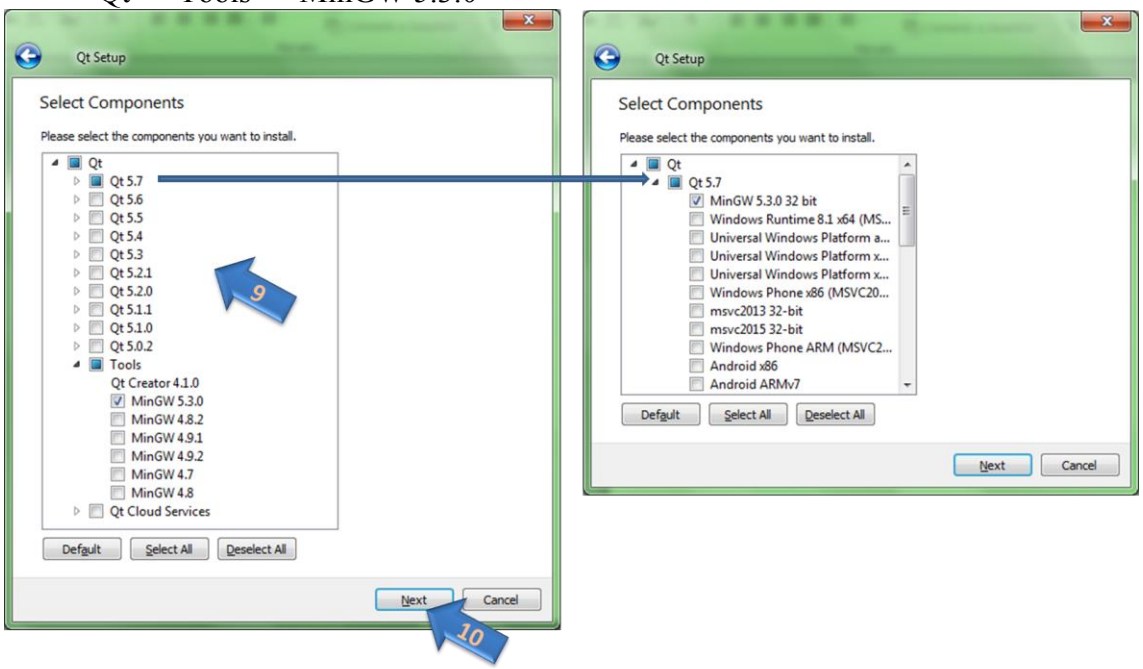
En la pantalla de bienvenida, pulsar Next (4), a continuación Skip (5) y Next (6).

Seleccionar el directorio destino de la instalación (7) (**atención**, para evitar problemas no utilice un camino de directorio que incluya espacios en blanco) y pulsar Next (8).

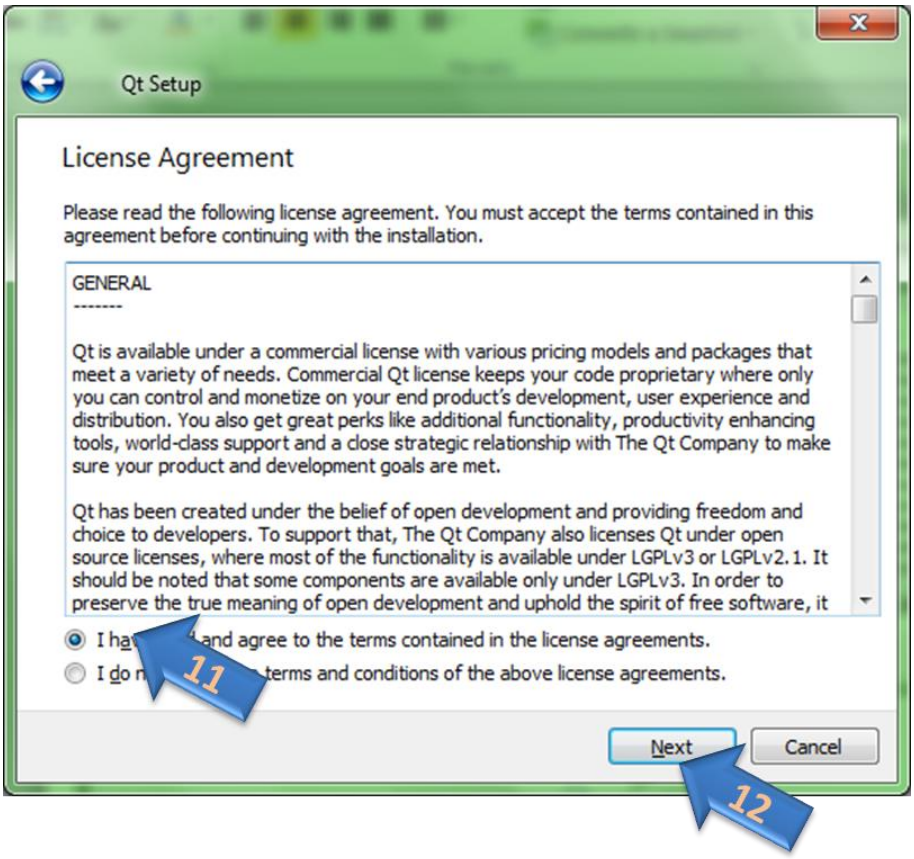


A continuación se presentan las opciones de instalación. Se deben seleccionar los siguientes elementos (9) antes de proceder con Next (10):

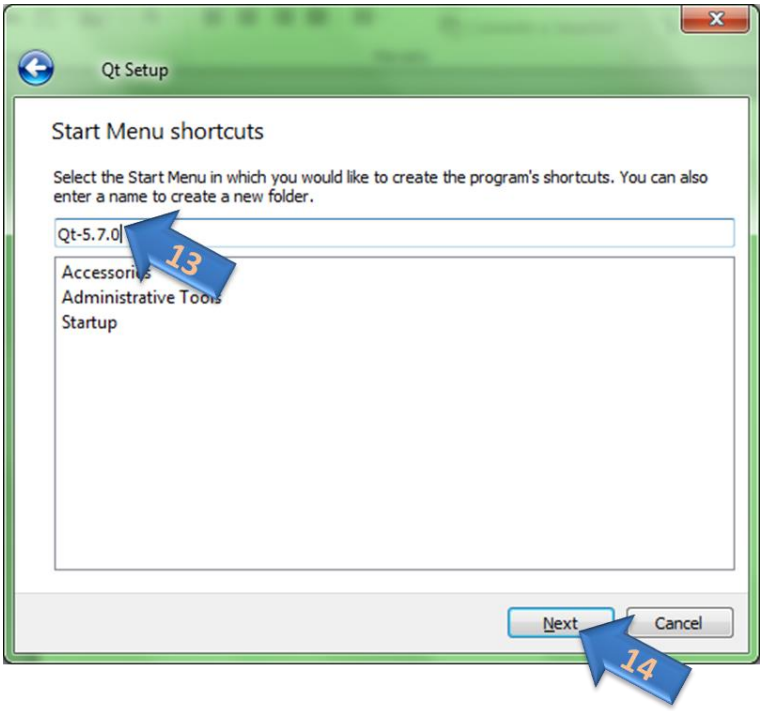
- Qt → Qt 5.7 → MinGW 5.3.0 32 bit
- Qt → Tools → MinGW 5.3.0



Otros componentes pueden se r de utilidad para realizar aplicaciones para otras plataformas (Windows Phone, Android ARM, etc.).
Aceptar acuerdo de licencia (11) y pulsar Next (12).



Indicar el nombre que queremos que aparezca en los menús de Windows (13), y pulsar Next (14).

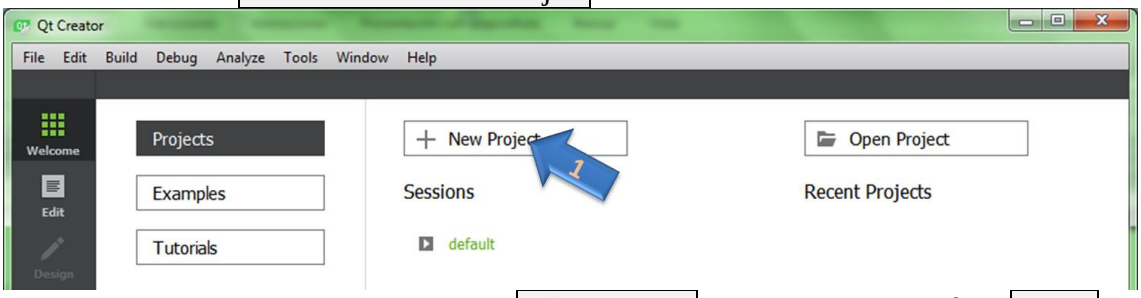


Por último, pulsar el botón Install (16). La instalación descargará los componentes y los copiará en el directorio creado, tomando unos 20-30 min para completar la operación. Una vez finalizada, dispondremos del programa Qt Creator en nuestro equipo.

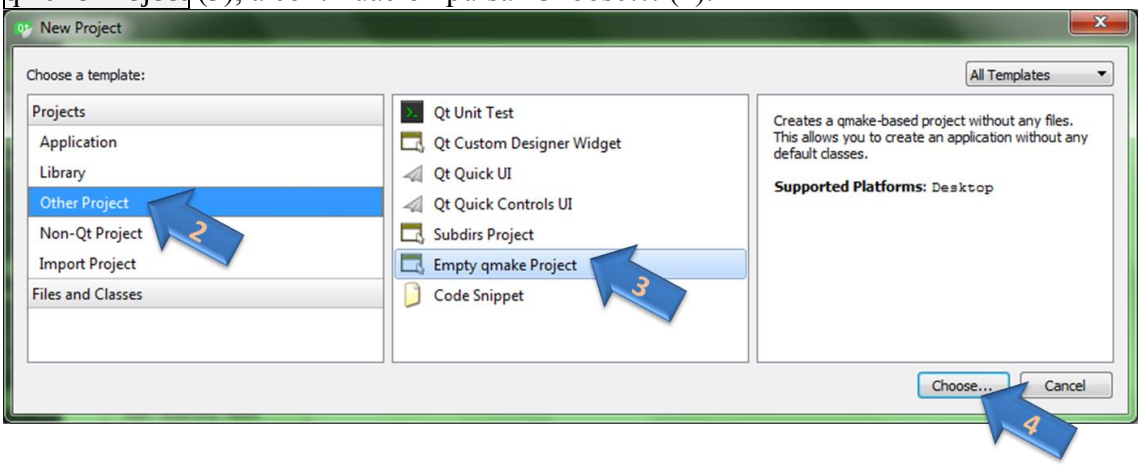
2. Crear un programa en modo consola

Ejecutar el programa Qt Creator y realizar los siguientes pasos para crear un programa en lenguaje C:

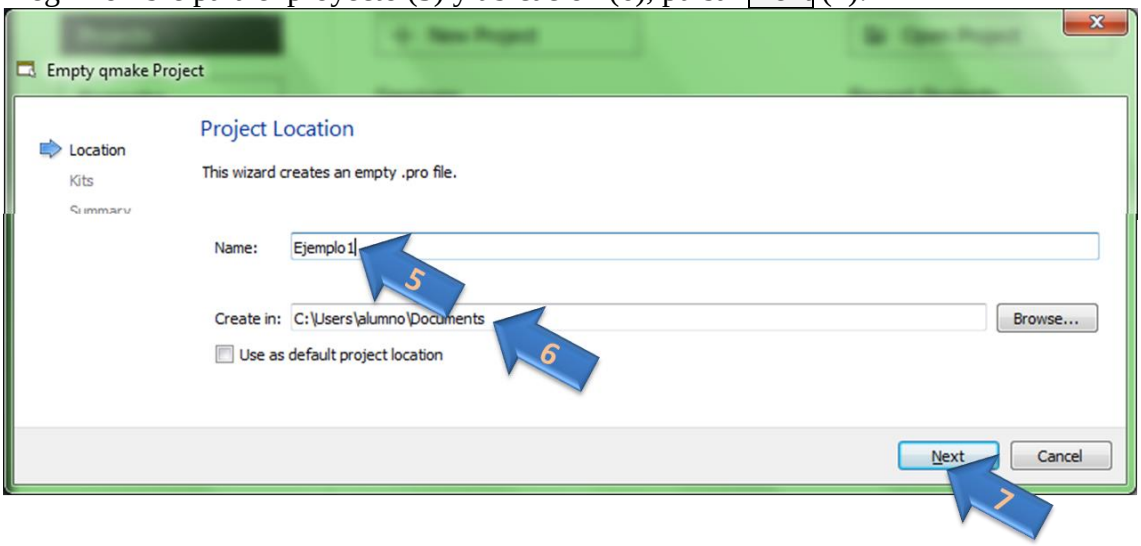
- Crear nuevo proyecto pulsando New Project (1) en la ventana de bienvenida, o bien mediante el menú File → New File or Project.



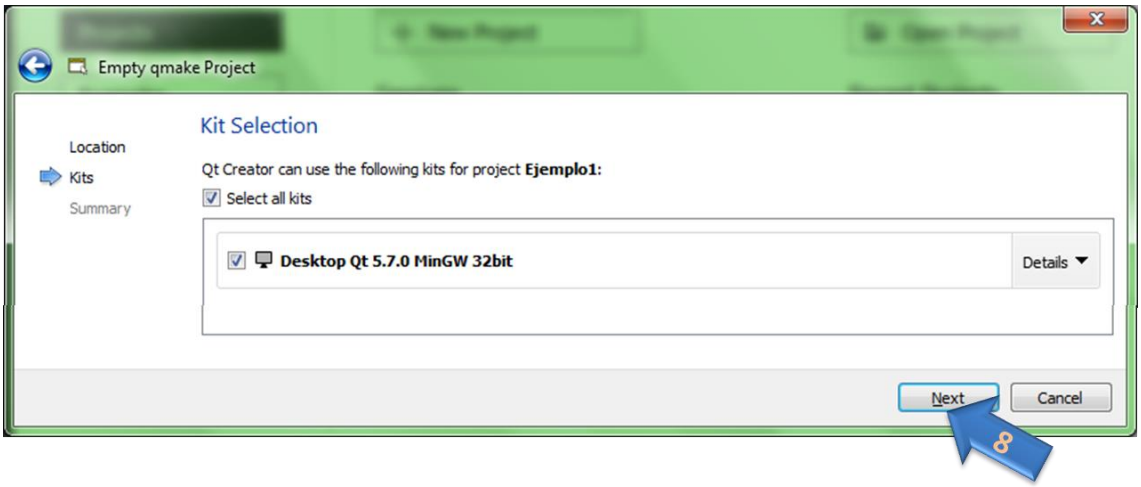
- Seleccionar las opciones de proyecto Other Project (2), y dentro de éste Empty qmake Project (3), a continuación pulsar Choose… (4).



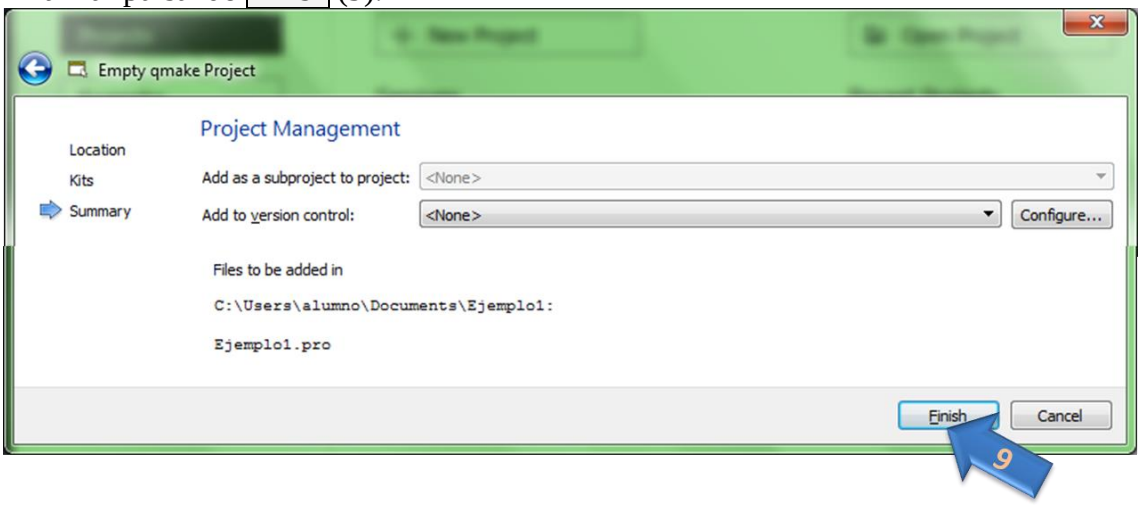
- Elegir nombre para el proyecto (5) y ubicación (6), pulsar Next (7):



- Si en las opciones de instalación sólo se eligió MingGW, aparecerá un único Kit de compilación. Pulsar Next (8) (si se añadieron otras opciones de instalación, eliminar los check del resto de kits antes de pulsar Next).

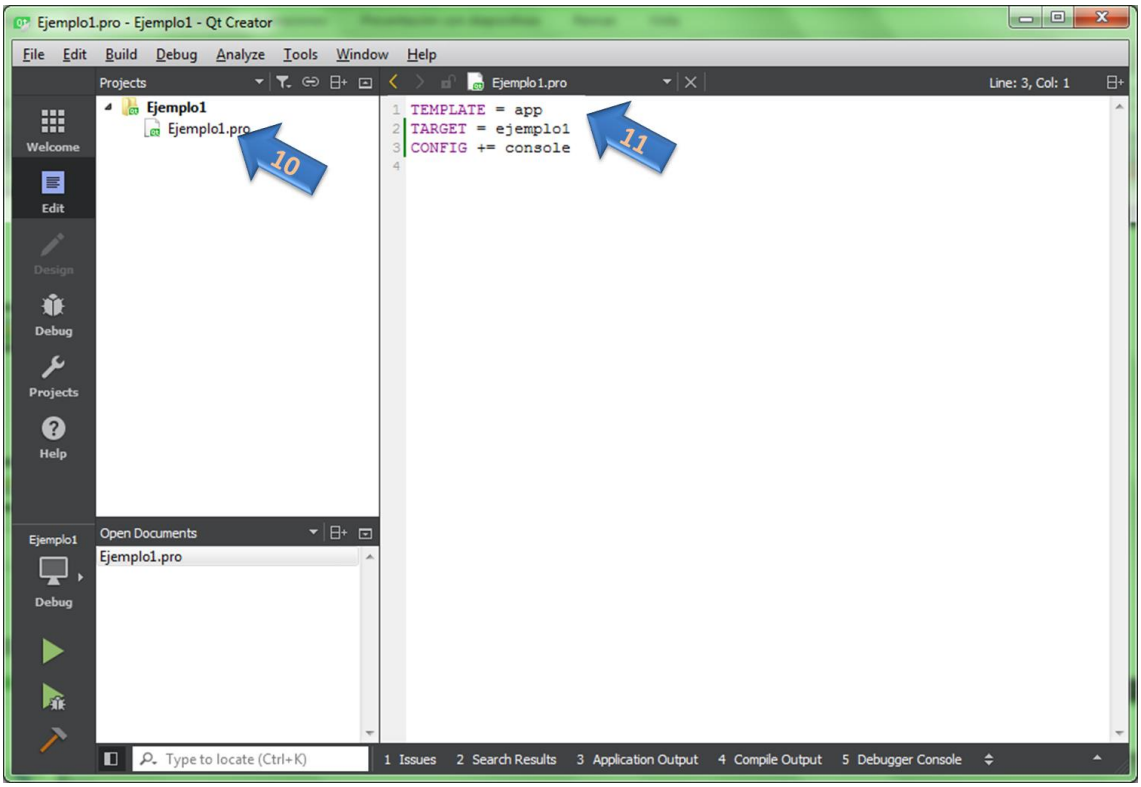


- Finalizar pulsando Finish (9).

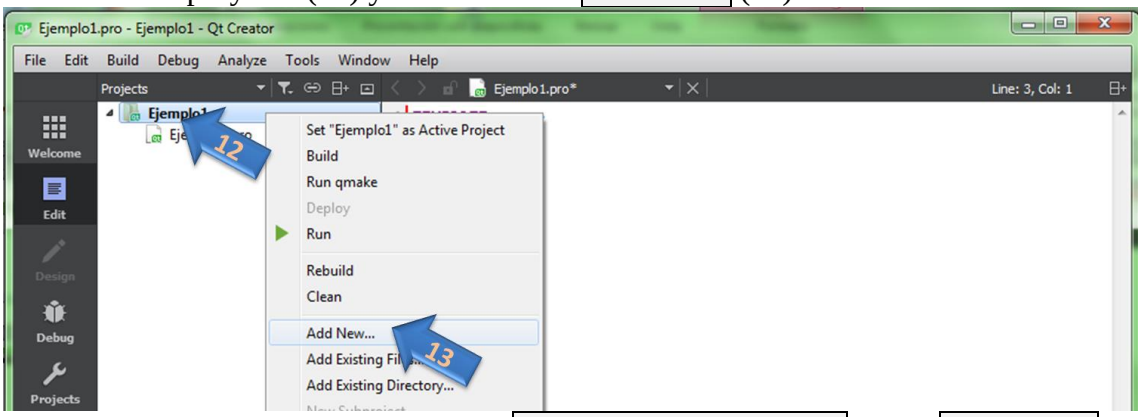


- Aparecerá una ventana con un menú gráfico en la parte izquierda, una ventana de proyectos a su lado, y una ventana de trabajo a la derecha.
- Indicar las opciones de nuestro proyecto en el archivo .pro creado. Para ello, se hace doble-click sobre el elemento en la ventana de proyecto (10) y se escribe el texto indicado en la ventana de trabajo (11). Las líneas a escribir son:

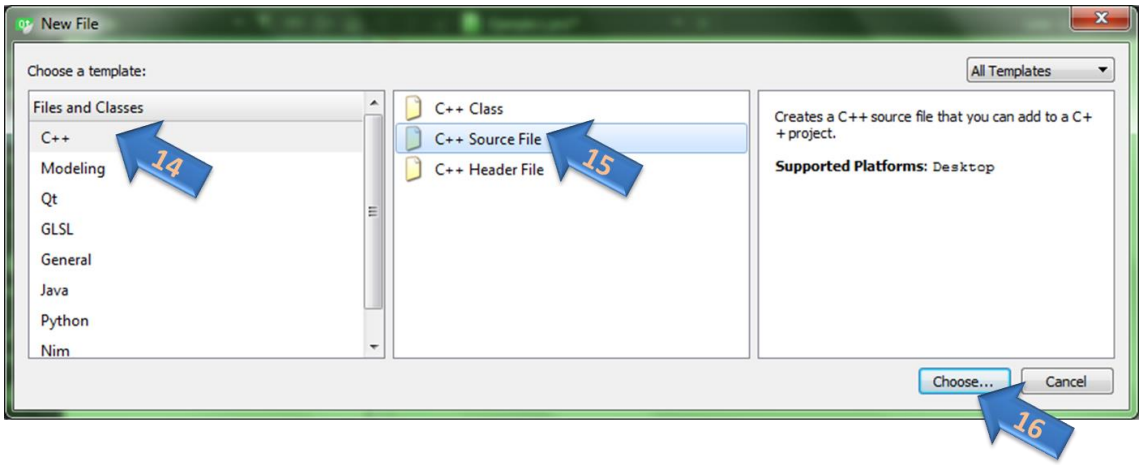
`TEMPLATE = app`	*Indica que el objeto de nuestro proyecto es una aplicación o programa; se puede escribir lib en lugar de app para realizar una librería*
`TARGET = ejemplo1`	*Nombre del ejecutable que deseamos crear*
`CONFIG += console`	*Añadir (a la configuración base) que deseamos un programa modo consola*



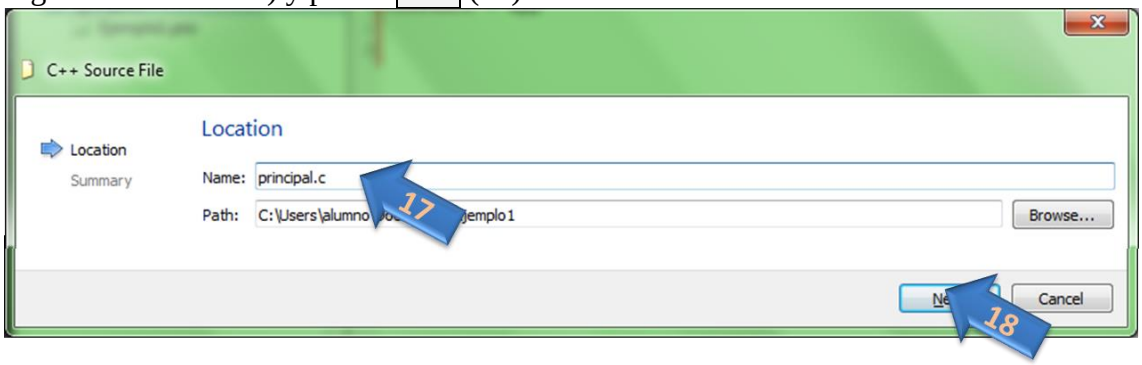
- Añadir un nuevo archivo de código fuente, pulsando el botón derecho del ratón sobre el nombre del proyecto (12) y seleccionando Add New... (13) en el menú contextual.



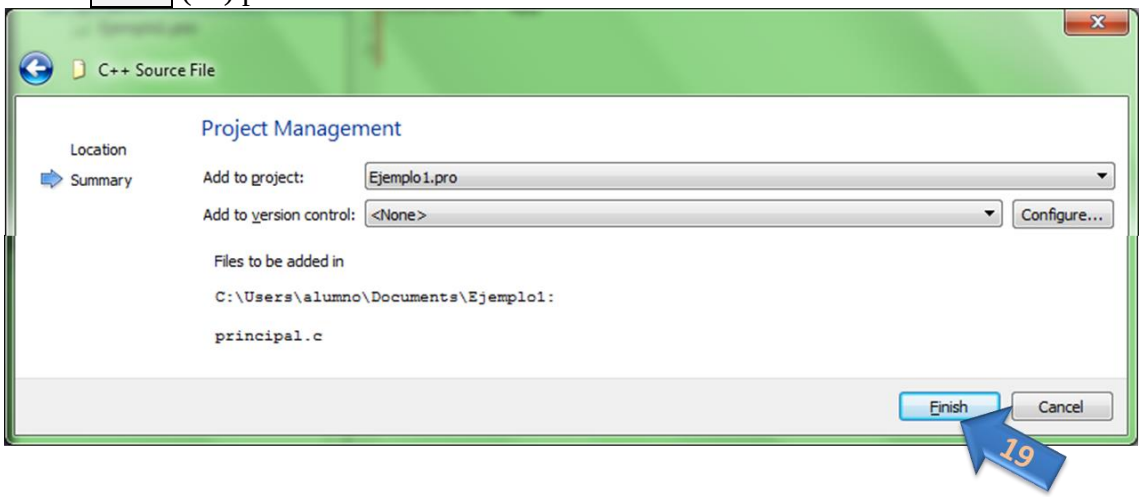
- En el diálogo que aparece, seleccionar Files And Classes →C++ (14) , C++ Source File (15) y presionar Choose… (16).



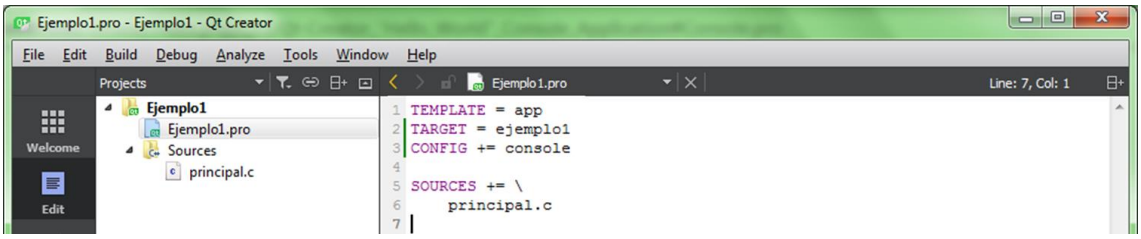
- Dar nombre al archivo de código fuente, terminando con extensión .c (17) (la **extensión .c** es muy importante, ya que el entorno utilizará el compilador adecuado según esta extensión) y pulsar Next (18).



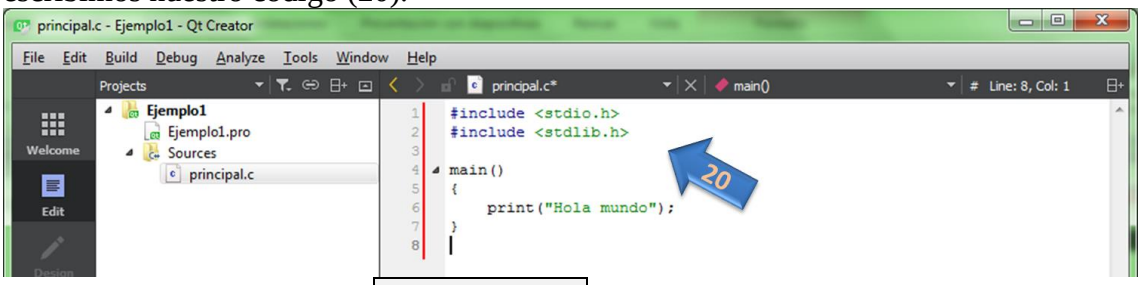
- Pulsar Finish (19) para terminar la adición de nuevo archivo.



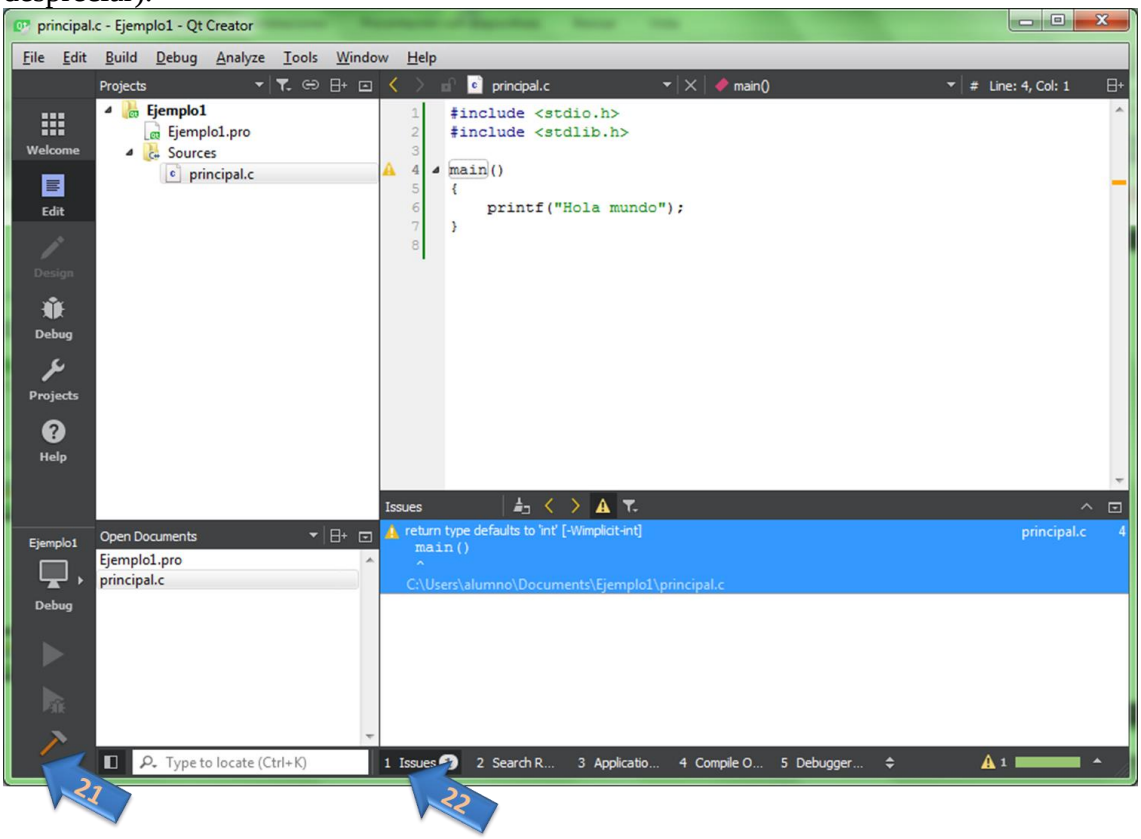
- Se puede ver que se ha agregado en la ventana de proyecto una carpeta Sources que contiene nuestro archivo (principal.c). Si volvemos a la edición del archivo de proyecto (.pro), comprobamos que ha sido modificado automáticamente.



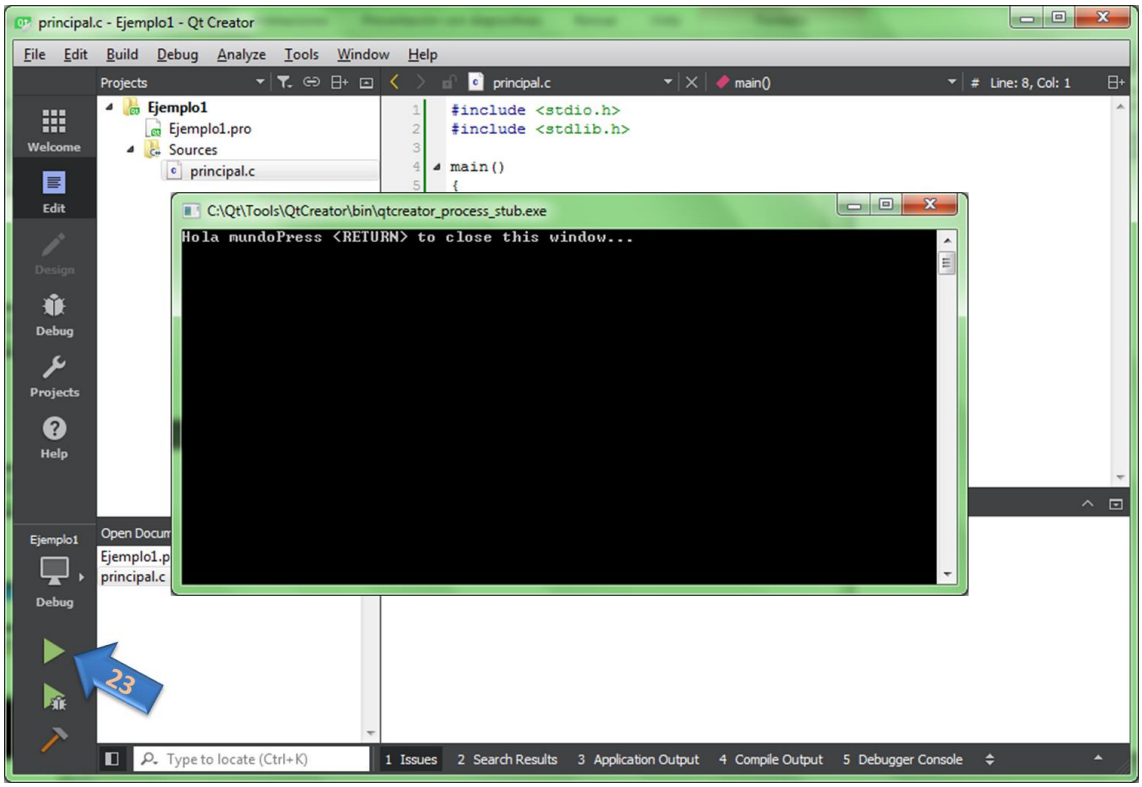
- Volvemos a principal.c (haciendo doble –click en la ventana de proyecto) y escribimos nuestro código (20).



- Una vez escrito y guardado (File → Save All), se procede a la compilación pulsando el icono martillo que aparece abajo a la izquierda (21) (opción de menú alternativa Build → Build All). El resultado de la compilación se comprueba pulsando sobre la ventana Issues (22) (en este caso hay un aviso o warning, que de momento se puede despreciar).

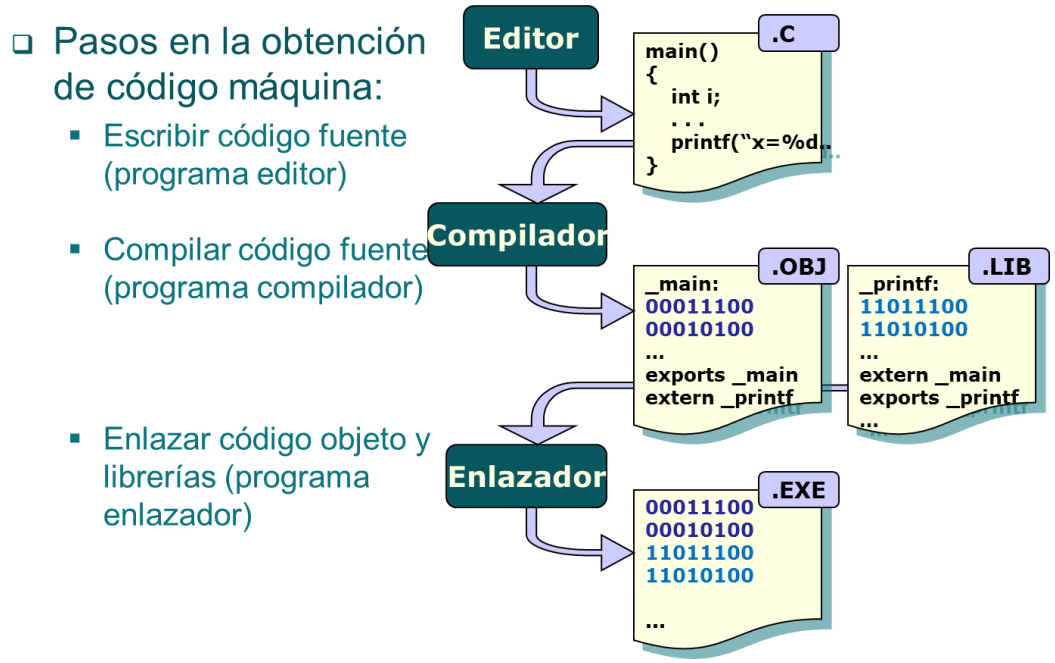


- Finalmente, se puede ejecutar el programa generado (si no hay errores) pulsando el botón de ejecución (triángulo verde abajo a la izquierda) (23) o con la opción de menú Build → Run. Aparecerá una ventana de consola donde se ejecuta nuestro programa.


principal.c - Ejemplo1 - Qt Creator
File Edit Build Debug Analyze Tools Window Help
Projects
Ejemplo1
Ejemplo1.pro
Sources
principal.c
principal.c
main()
Line: 8, Col: 1
#include <stdio.h>
#include <stdlib.h>
main()
{
C:\Qt\Tools\QtCreator\bin\qtcreator_process_stub.exe
Hola mundoPress <RETURN> to close this window...
Welcome
Edit
Design
Debug
Projects
Help
Ejemplo1
Debug
23
Type to locate (Ctrl+K)
1 Issues
2 Search Results
3 Application Output
4 Compile Output
5 Debugger Console

3. Edición, compilación y enlazado

3.1. Pasos en la generación de un programa



3.2. Edición del código

A la hora de escribir el código fuente correspondiente al programa que se desea realizar, se debe prestar especial atención a:

- Sintaxis: el compilador sólo aceptará el código si su sintaxis es correcta.
- Indentación: estructurar los bloques en columnas (usar Tab) de forma que el código sea fácilmente legible.
- Comentarios: facilitan la comprensión de lo que se ha escrito

El entorno facilita la visualización de lo que se está haciendo correctamente:

- Las palabras clave de C aparecerán en color verde grisáceo.
- Los comentarios (texto que no se compila) aparecerán en verde intenso.
- Las cadenas de texto aparecerán en color verde intenso.
- Las directivas y constantes en color azul.
- El texto que no parece estar conforme a la sintaxis aparece subrayado en rojo

Ejemplo **(no copiar/pegar directamente del PDF, pueden aparecer caracteres invisibles que el compilador no entienda):**

```
#include <stdio.h>
#include <stdlib.h>

#define TAM_TABLA      5

int Factorial(int n)
// Función Factorial(). Calcula el factorial de un nº entero
// Parámetros de entrada:
// n (int): entero del que queremos calcular el factorial
// Valor devuelto:
// (int): entero con el valor del factorial
{
    int i,result;
    for (i=1,result=1;i<n;i++)
        result=result*i;
    return result;
}

main()
// Programa principal
{
    int i;
    int x[TAM_TABLA],fact[TAM_TABLA];

    for (i=0;i<TAM_TABLA;i++)
    {
        printf("Introduzca x[%d] = ",i);
        scanf("%d",&x[i]);
        fact[i]=Factorial(x[i]);
        printf("El factorial de %d es %d",x[i],fact[i]);
    }
}
```

Una vez escrito y comprobado el código, salvar el archivo. En caso de archivos grandes, se debe salvar a intervalos regulares para evitar pérdida de información en caso de fallo en el ordenador o la alimentación.

3.3. Compilar y enlazar

Tras la creación del código, usar la opción de menú Build -> Build Project (Ctrl-B) o sobre el icono martillo de la parte inferior izquierda para invocar al compilador.
El compilador intentará producir el código máquina a partir del texto fuente, pero no podrá si hay errores sintácticos en el código.
Se debe comprobar que no se producen errores en las ventanas 1-Issues o 4-Compile Output , ambas accesibles en la zona inferior de la ventana de trabajo.
Para todos los errores, en la ventana indicará el archivo y la línea de código donde se detectó. Haciendo doble-click sobre el error, nos lleva automáticamente a su posición en la ventana de edición.
¡ATENCION! El compilador no sabe lo que pretendíamos escribir, sólo es capaz de detectar que el texto no cumple la sintaxis y las reglas del lenguaje C. No siempre el texto y la posición del error indican exactamente lo que se debe corregir.

3.4. Errores más típicos de compilación

- Errores de sintaxis: falta de ; al final de una sentencia, apertura y cierre descompensado de paréntesis, corchetes, comillas, etc.

```
scanf("%d",&x[i])    // Falta el ;
fact[i]=Factorial(x[i]);
```

```
...\principal.c:29: error: expected ';' before 'fact'
  fact[i]=Factorial(x[i]);
  ^
```

- Nombre de variable o palabra clave incorrecto. Recordar que el compilador distingue mayúsculas de minúsculas. Ejemplo:

```
result=resul*i;    // Falta la t en el 2° result
```

```
...\principal.c:15: error: 'resul' undeclared (first use in this function)
  result=resul*i;
         ^
```

- Utilización incorrecta de un operador (falta de operandos, operandos de tipo inadecuado, etc.). Ejemplo:

```
result=result*;    // Falta 2° operando para el producto
```

```
...\principal.c:15: error: expected expression before ';' token
  result=result*;
                ^
```

3.5. Advertencias (warning) más típicas en compilación

- Declaración de una variable que no se utiliza después en el código:

```
int z; // Ninguna instrucción hace después referencia a z
```

```
...\principal.c:23: warning: unused variable 'z' [-Wunused-variable]
int z;
    ^
```

- Uso de una variable sin inicializar. Ejemplo:

```
int a,b;
a=b+2;       // al no dar antes valor a b, no se sabe lo
             // que valdrá a tras ejecutar la instrucción
```

```
...\principal.c:23: warning: 'b' is used uninitialized in this function [-Wuninitialized]
a=b+2;
  ^
```

3.6. Errores más típicos de enlazado (link)

El enlazado sólo se realiza si la compilación ha sido correcta. Los errores de enlazado siempre terminan con uno final:

```
collect2.exe:-1: error: error: ld returned 1 exit status
```

- Falta de función main()

```
crt0_c.c:-1: error: undefined reference to `WinMain@16'
collect2.exe:-1: error: error: ld returned 1 exit status
```

- Nombre de función incorrecto (recordar que el compilador distingue mayúsculas de minúsculas), suele venir acompañado de un warning. Ejemplo:

```
fact[i]=factorial(x[i]); // La función está definida como
                         // Factorial() – F mayúscula
```

⚠ ….\principal.c:32: warning: implicit declaration of function 'factorial' [-Wimplicit-function-declaration]
fact[i]=factorial(x[i]);
^
❗ …\principal.c:32: error: undefined reference to `factorial'
❗ collect2.exe:-1: error: error: ld returned 1 exit status

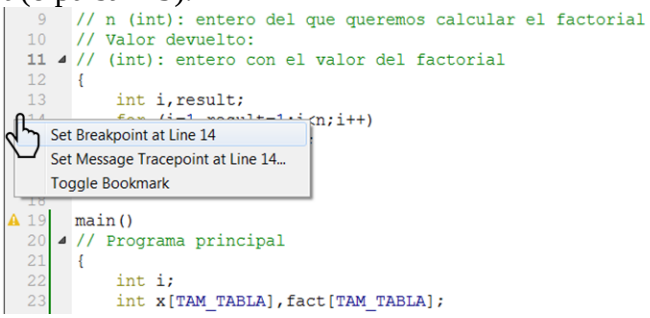
4. Ejecutar el programa

Probar la ejecución del programa mediante la selección de la opción de menú Build ->Run (Ctrl-R) o el icono triángulo (abajo a la izquierda, el que no lleva un 'bicho' encima).
En ese momento se desplegará una nueva ventana, en modo consola, en la que aparecerá el resultado de la ejecución programa.

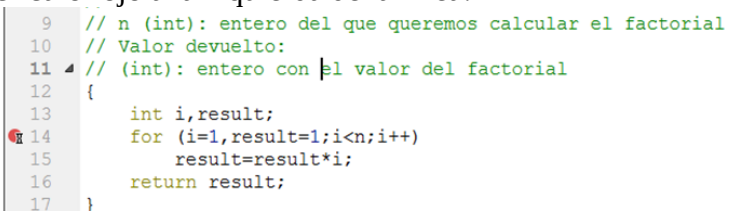
4.1. Depuración de errores en ejecución

El hecho de que un programa compile correctamente **no implica que haga lo que se espera**. En caso de que la ejecución no se corresponda con lo esperado, es necesario Depurar (Debug), esto es, buscar los errores de ejecución. Las herramientas disponibles para ello son:

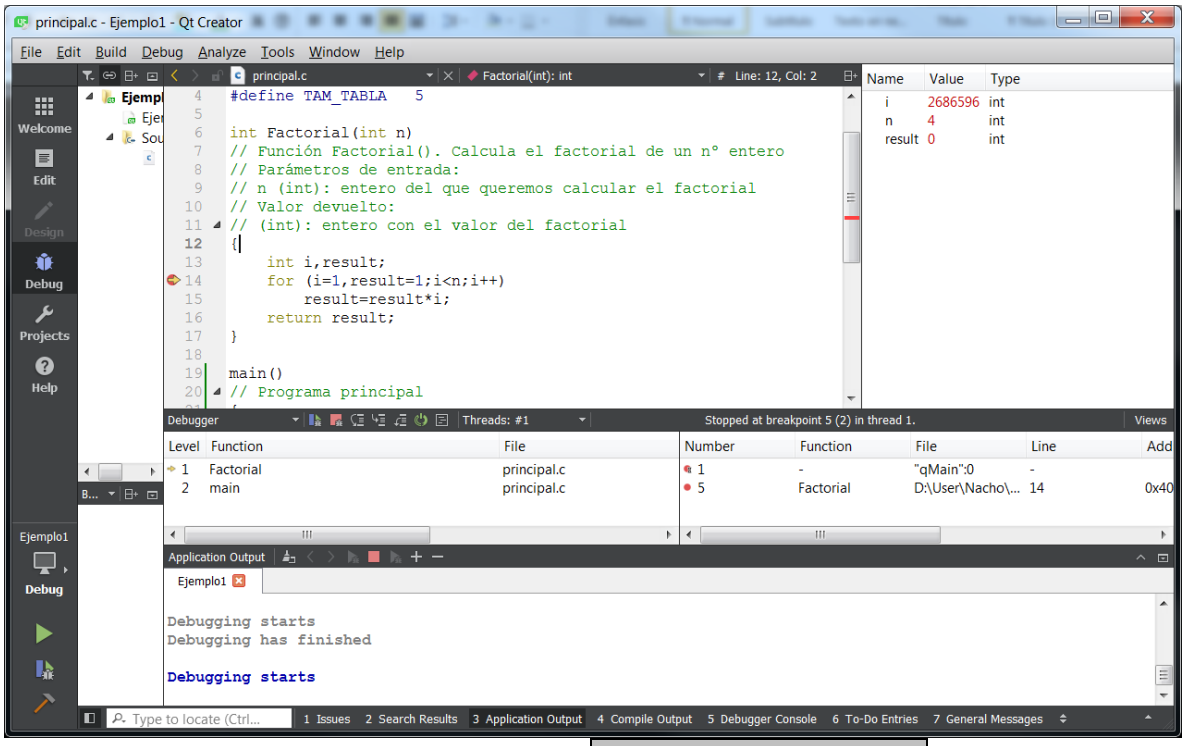
- Ejecución hasta punto de interrupción (breakpoint). Seleccionar la línea de código donde queremos que se detenga la ejecución, colocándose a la izquierda del número de línea (el icono del ratón se pasa a ver como una mano), y pulsar botón-derecho + Set BreakPoint (o pulsar F9).



Aparecerá un círculo rojo a la izquierda de la línea:



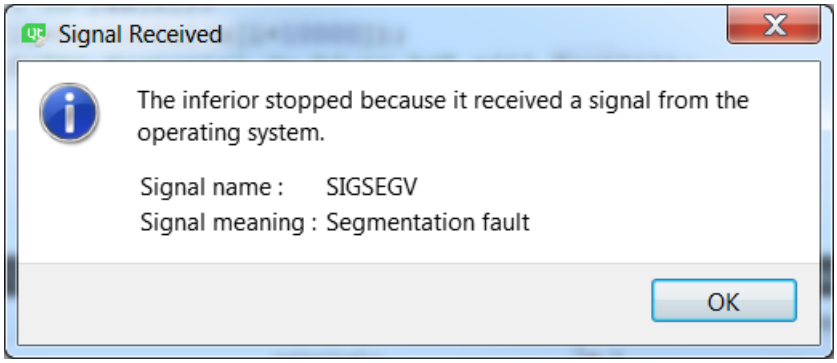
que indica que la ejecución se detendrá cuando se llegue a este punto (si se ha iniciado con ejecución en modo depuración), y se podrá continuar paso a paso o hasta el siguiente punto de interrupción. El punto de interrupción se quita con una operación similar.
Una vez establecido el punto de interrupción, iniciar el programa con Debug -> Start Debugging [F5], la ejecución se detendrá al alcanzar el punto de interrupción, en cuyo momento la organización de las ventanas pasa a modo Debug, tal y como se muestra en la siguiente figura, y disponemos de las siguientes opciones:



- Ejecución paso a paso (step over). Menú Debug -> Step Over [F10]. La ejecución se detiene tras la ejecución de la siguiente línea de código.
- Ejecución paso a paso (step into). Seleccionar menú Debug -> Step Into [F11]. La ejecución se detiene tras la ejecución de cada línea de código, pero saltará a la ejecución interna de una función que sea llamada en esa línea.
- Continuar hasta siguiente punto de interrupción (nuevamente F5 o con Debug -> Continue [F5]).
- Terminar la ejecución (Debug -> Stop Debugger).
- Visualización de variables. En la ventana de la derecha se muestran las variables relevantes en el momento de ejecución actual, y sus valores actuales. Según se avanza (paso a paso, hasta siguiente punto de interrupción) se puede comprobar en esta ventana los nuevos valores de las variables.
- Visualización de pila de llamadas. En la ventana que queda justo bajo el código se puede comprobar el estado de la pila (stack) hasta el punto de ejecución actual, y también seleccionar el contexto de pila (variables locales, parámetros) de la función que se quiere visualizar.
- Puntos de interrupción condicionales. En modo depuración, la ventana Puntos de interrupción en la zona inferior derecha de la pantalla permite habilitar, inhabilitar y borrar puntos de interrupción. Adicionalmente, se puede seleccionar un punto de interrupción con el botón derecho del ratón y hacerlo condicional, esto es, que sólo se detenga la ejecución cuando se cumpla una determinada condición (valor de variable, nº de pasadas/visitas por ese punto).

4.2. Parada abrupta del programa ante error de ejecución

En algunas ocasiones, alguna de las instrucciones del programa no puede ser ejecutada porque violaría la seguridad o excede las posibilidades del computador, siendo la más típica el acceso a una posición de memoria inválida.
En estos casos, aparecerá un aviso como el que sigue:

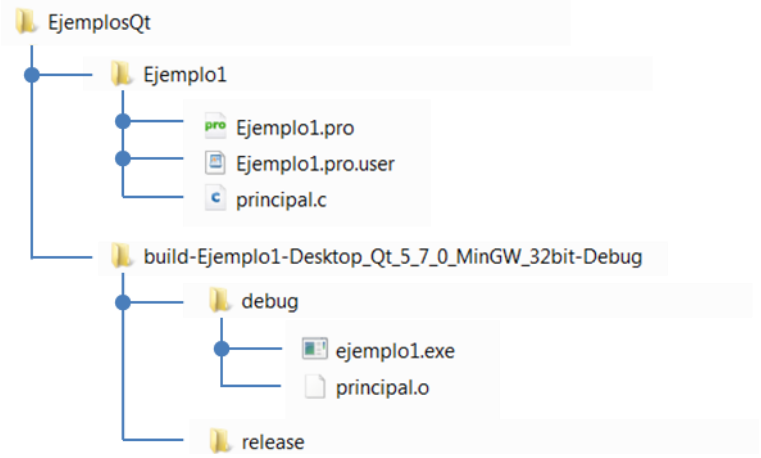


Tras pulsar Ok, se puede acudir a la pila de llamadas para comprobar el punto donde el programa no ha podido ejecutarse, y la pila de llamadas que ha llevado a ese punto:

¡ ATENCION ! En ciertas ocasiones, el error puede provocar daños en la pila de llamadas que hagan imposible al depurador localizar la función que se estaba ejecutando. En estos casos, la única solución es la depuración paso a paso hasta encontrar la instrucción que produce la excepción.

5. Mover un desarrollo a otros equipos

Cuando se ha realizado un programa, la estructura de directorios donde lo hemos desarrollado queda de manera similar a la siguiente:



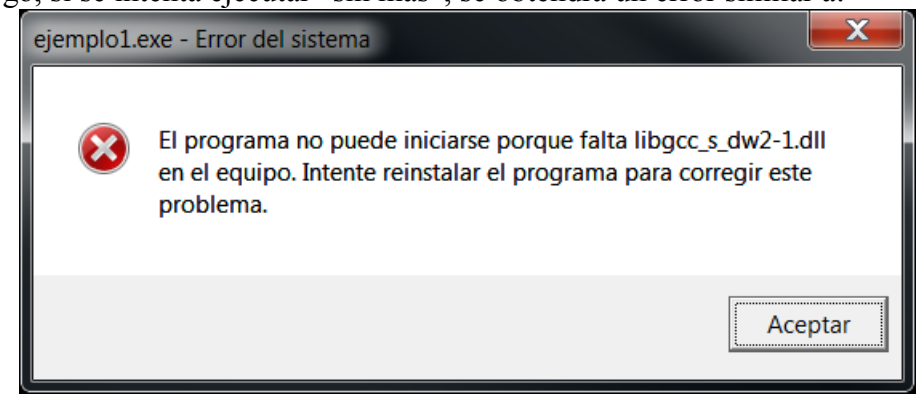
En el directorio de trabajo, especificado en el paso (6) de "2. Crear un programa en modo consola", aparece un directorio con el nombre del proyecto (indicado en el paso 5), dentro del cual se encuentra nuestro archivo de proyecto (.pro), nuestro(s) archivo(s) de código fuente (.c), y un archivo adicional .pro.user (contiene características específicas del proyecto).

Adicionalmente, aparece un 2º directorio build- ….. , que contiene los resultados de compilación: código objeto (.o) y ejecutable (.exe).

5.1. Utilizar el ejecutable fuera del entorno

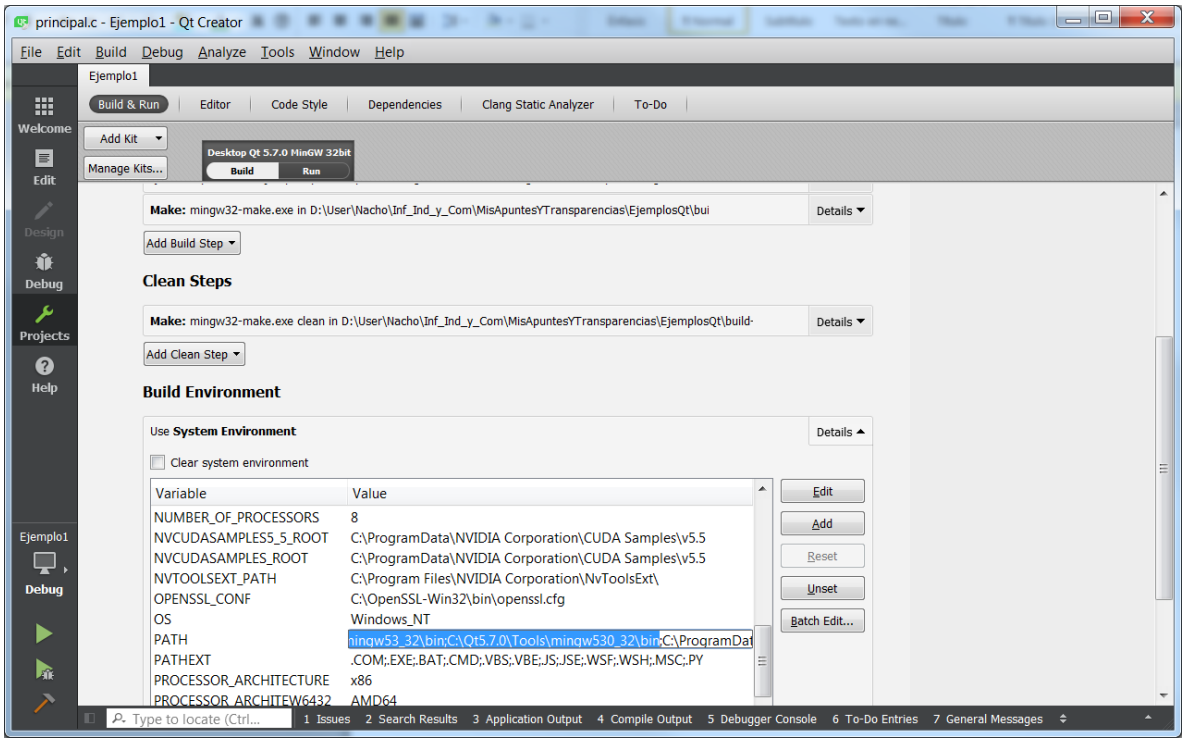
El archivo .exe es el único necesario para la ejecución del programa fuera del entorno. Se puede copiar o mover a otro equipo o directorio, y haciendo doble-click sobre él se produce su ejecución.

Sin embargo, si se intenta ejecutar "sin más", se obtendrá un error similar a:

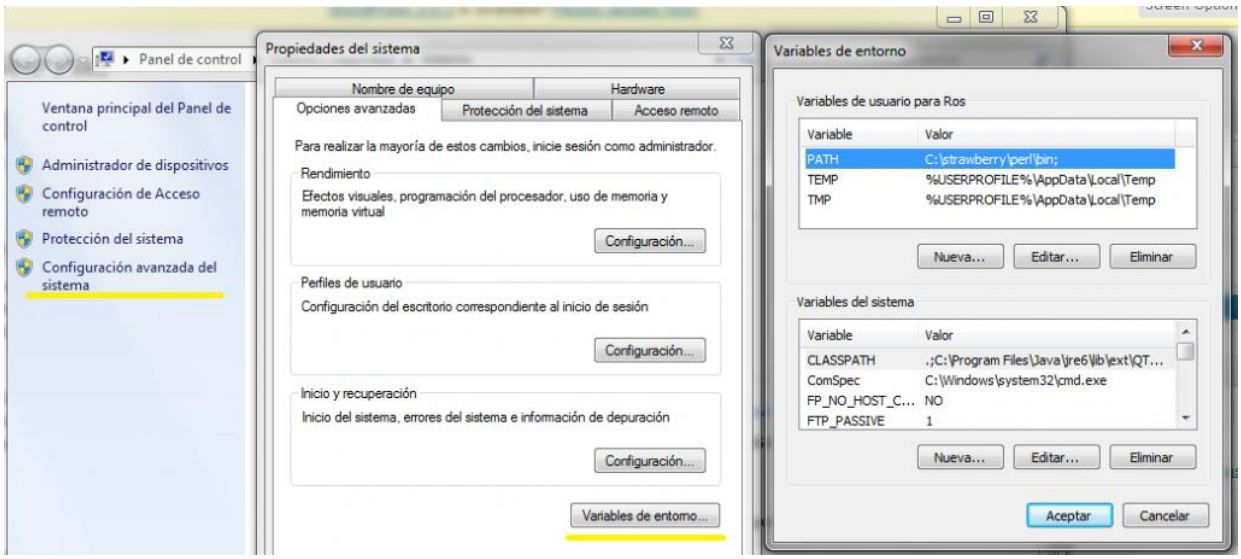


Esto ocurre porque el programa necesita una serie de librerías de enlace dinámico (DLL), cuya ubicación es conocida cuando se ejecuta dentro del entorno Qt Creator, pero no fuera de él.

Si lo queremos ejecutar en el mismo PC de desarrollo habrá que indicar la ubicación de estas librerías, que se puede saber mirando el contenido de PATH en Projects -> Build Environment:



Una vez comprobado el PATH específico de las DLL de Qt (típicamente será algo como C:\Qt\5.7\mingw53_32\bin;C:\Qt\Tools\mingw530_32\bin), se puede añadir al entorno del usuario para que siempre esté presente. En Windows:
Equipo → botón-derecho → Propiedades → Configuración avanzada del sistema → Variables de Entorno → Variables de Usuario → PATH



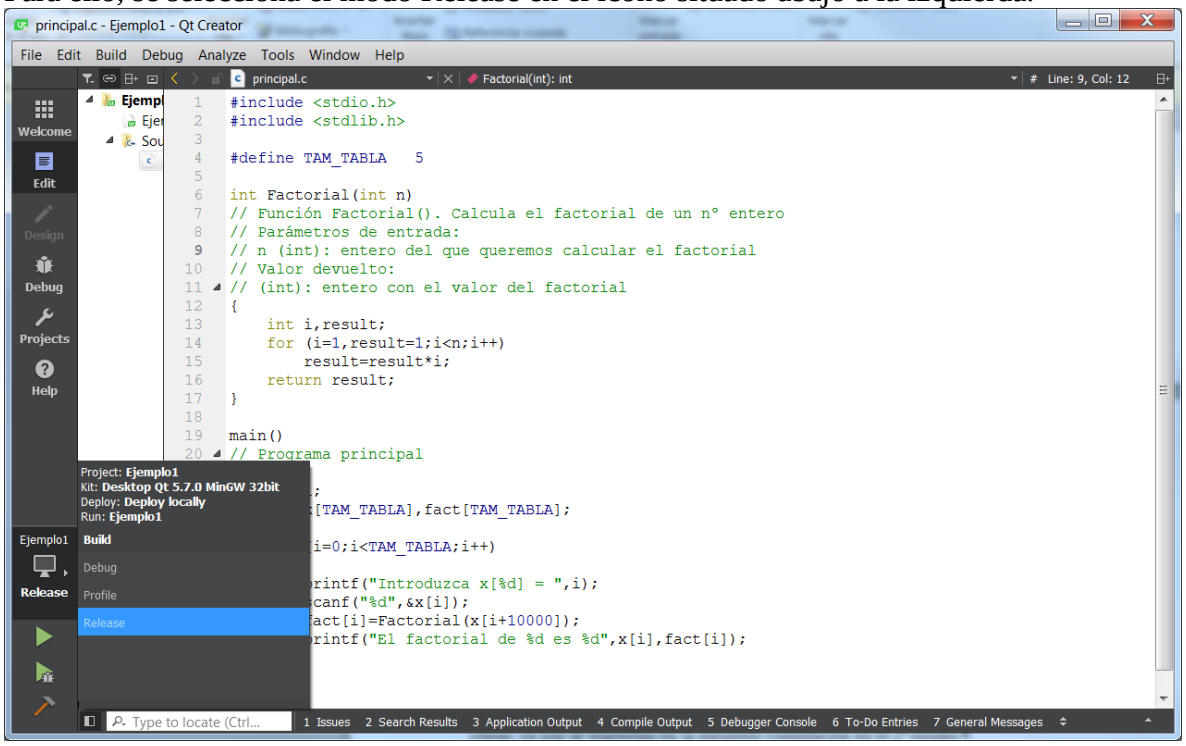
PATH = C:\Qt\5.7\mingw53_32\bin;C:\Qt\Tools\mingw530_32\bin

Para que esta nueva configuración tenga efecto, es posible que sea necesario cerrar y reiniciar la sesión de usuario en Windows.

5.2. Distribuir el ejecutable (a otros equipos)

Si queremos ejecutar el programa en otro PC que no tenga QtCreator, habrá que copiar los directorios anteriores (los añadidos al PATH) al nuevo PC. Además, para este caso es conveniente compilar en modo Release (sin información de depuración, ya que no se va a utilizar).

Para ello, se selecciona el modo Release en el icono situado abajo a la izquierda.



Una vez recompilado, el ejecutable a copiar será el que se encuentre en el directorio build-…..-Release\Release.

5.3. Copiar el desarrollo a otro equipo

Si se desea continuar el desarrollo en otro equipo, habrá que copiar únicamente el directorio de proyecto (Ejemplo1 en este caso), excepto el archivo .pro.user . El resto no se deben copiar, ya que se regeneran en la siguiente compilación en el 2º equipo.

Haciendo doble-click sobre el archivo .pro del proyecto se abre con Qt Creator, pero al no disponer del archivo .pro.user (que está ligado a cada equipo), habrá que volver a seleccionar el kit de compilación (paso (8) de "2. Crear un programa en modo consola") antes de continuar.

6. Ampliar los usos de Qt Creator

Aunque en esta asignatura sólo se utilizará la programación en lenguaje C, modo consola, el entorno permite otros usos, que cada usuario puede explorar:

- Creación de programas para entorno gráfico en Windows (con sus ventanas, como cualquier aplicación de Windows). Requiere conocimientos de C++ y de entorno gráfico. Se puede encontrar una introducción a ambos en los enlaces:
  - http://isa.uniovi.es/~ialvarez/Curso/descargas/000-ProgramacionCpp.pdf
  - http://isa.uniovi.es/~ialvarez/Curso/descargas/002-ProgramacionGraficaQt.pdf
- Creación de programas para otros dispositivos y/o Sistemas Operativos, sin modificar el código fuente, utilizando otros "kits" de compilación. Hay kits disponibles para :
  - PC/Linux
  - ARM/Linux
  - ARM/Android
  - Apple/iOS
  - Windows Phone

  Ver http://doc.qt.io/qt-5/gettingstarted.html para instrucciones de instalación y creación de programas con estos kits.