uc3m | Universidad Carlos III de Madrid

Informática Industrial I

Grado en Ingeniería en Electrónica Industrial y Automática

Álvaro Castro González

Nicolas Burus

Mohamed Abderrahim

José Carlos Castillo Montoya

Práctica 1

Introducción al entorno de desarrollo QtCreator

A lo largo de los últimos años, han colaborado en impartir las prácticas presenciales, y por tanto proporcionar comentarios para mejorar algunos aspectos de este documento los profesores siguientes:

Ioannis Douratsos
Fares Abu-Dakka
Juan Carlos González Victores
Avinash Ranganath
Raúl Perula

Práctica 1

- Introducción al entorno de desarrollo QtCreator

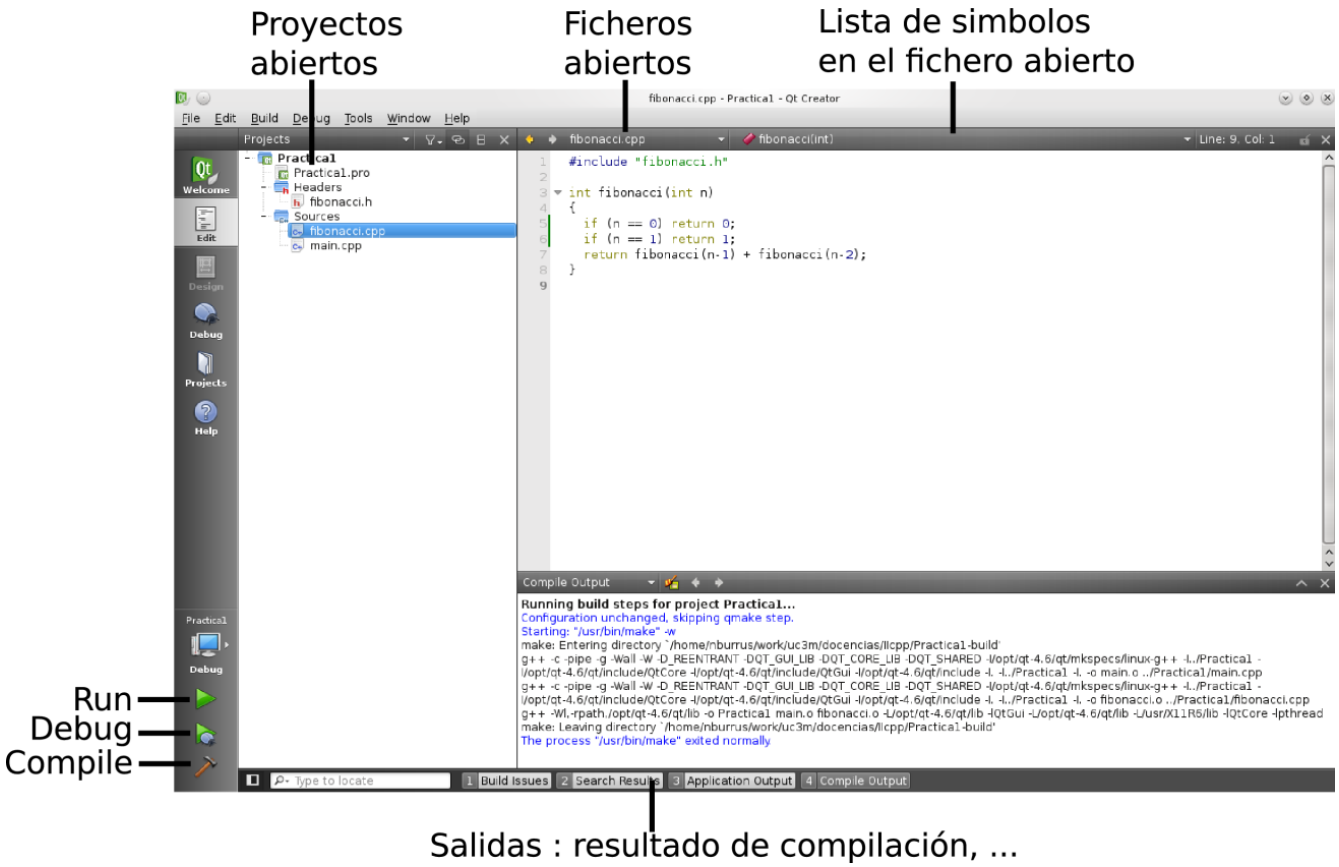
Para programar de manera eficiente en C++, muchos programadores utilizan entornos que facilitan las tareas de edición, diseño de interfaces gráficas, compilación, depuración y ejecución de los programas desarrollados. Entre los más populares se encuentran Eclipse CDT (open source, Windows/Linux/Mac) y QtCreator (open source, Windows/Linux/Mac), Microsoft Visual Studio (propietario, de pago, solo para Windows), Apple Xcode (propietario, gratis, solo para Mac).

En las prácticas de esta asignatura se va a trabajar con el entorno de desarrollo QtCreator, ya que es bastante sencillo y funciona en SO Linux.

El objetivo de esta primera práctica es familiarizarse con esta herramienta y crear vuestros primeros programas en C++. Esta práctica se desarrolla paso a paso para editando y ejecutando los ejemplos de programación que se proponen en cada apartado.

1. Presentación de la ventana principal

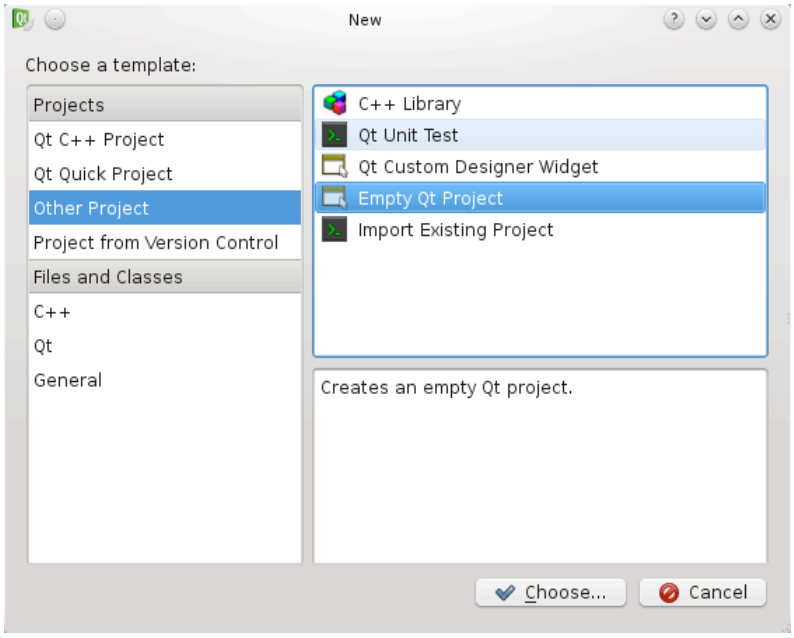
1.1. Arrancar QtCreator, seleccionar la pestaña ***Edit*** para obtener la ventana principal del entorno.

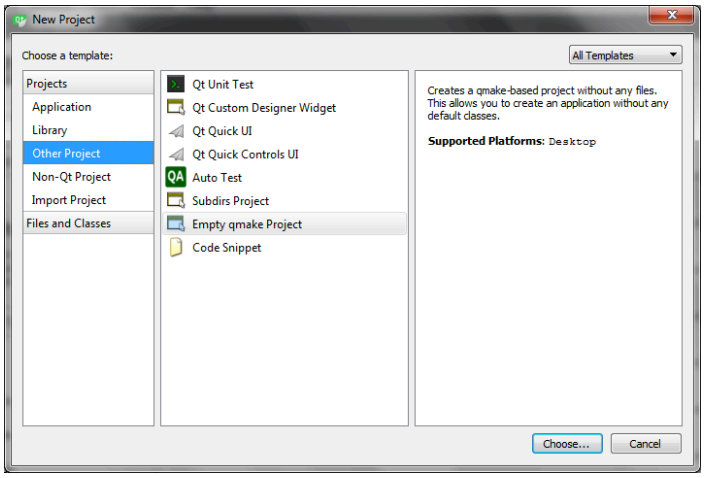


2. Crear un nuevo proyecto

La primera etapa es la creación de un nuevo proyecto. Para ello:

1. Seleccionar ***File/New File or Project***.
2. Seleccionar ***Other Project → Empty Qt Project***.
3. O seleccionar ***Other Project → Empty qmake Project***. (en nuevas versiones)

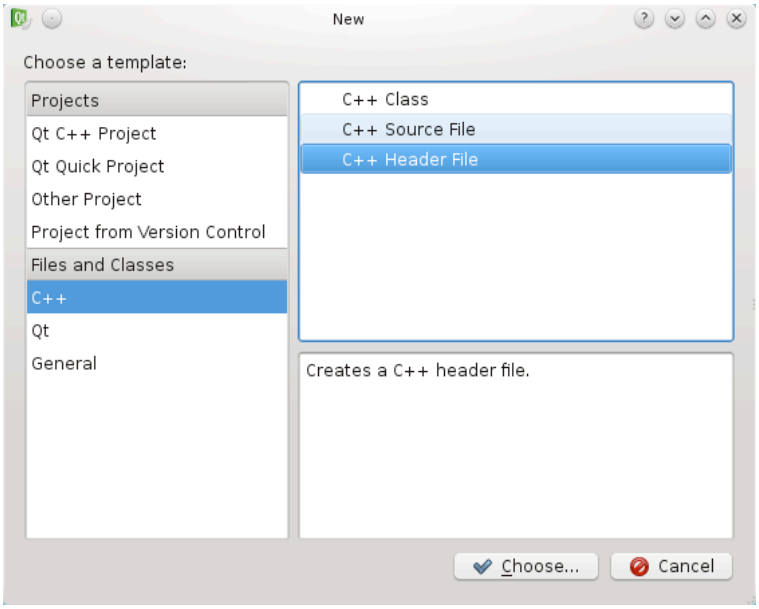




4. Darle el nombre ***practica1*** y elegir una carpeta. Darle a siguiente hasta finalizar.

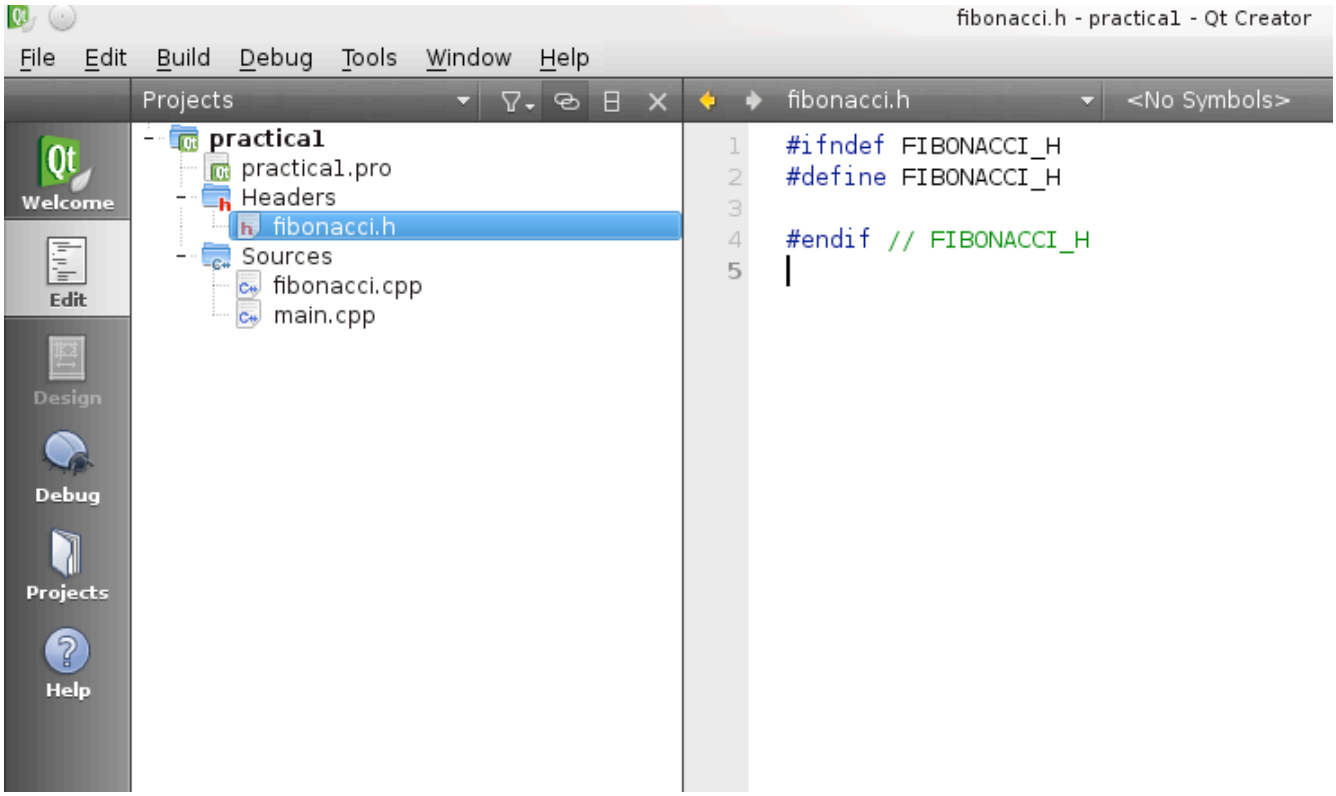
Ahora tenemos un proyecto vacío al que vamos a añadir dos ficheros (un fichero de cabecera y un fichero fuente) que calculen la función de Fibonacci de un entero.

5. Seleccionar ***File/New File or Project***.



6. Seleccionar ***C++ / C++ Header File***, nombrarle ***fibonacci.h***.
7. Hacer lo mismo con ***C++ / C++ Source File*** para añadir un fichero fuente de nombre ***fibonacci.cpp***.
8. Añadir por último otro fichero fuente y llamarlo ***main.cpp***.

   Se debería obtener un aspecto del entorno similar a la siguiente imagen:



3. Edición del código

Ahora vamos a editar el código. Para ello:

1. Seleccionar el fichero ***fibonacci.h*** y añadir la siguiente declaración de la función a implementar:

```
int fibonacci(int n);
```

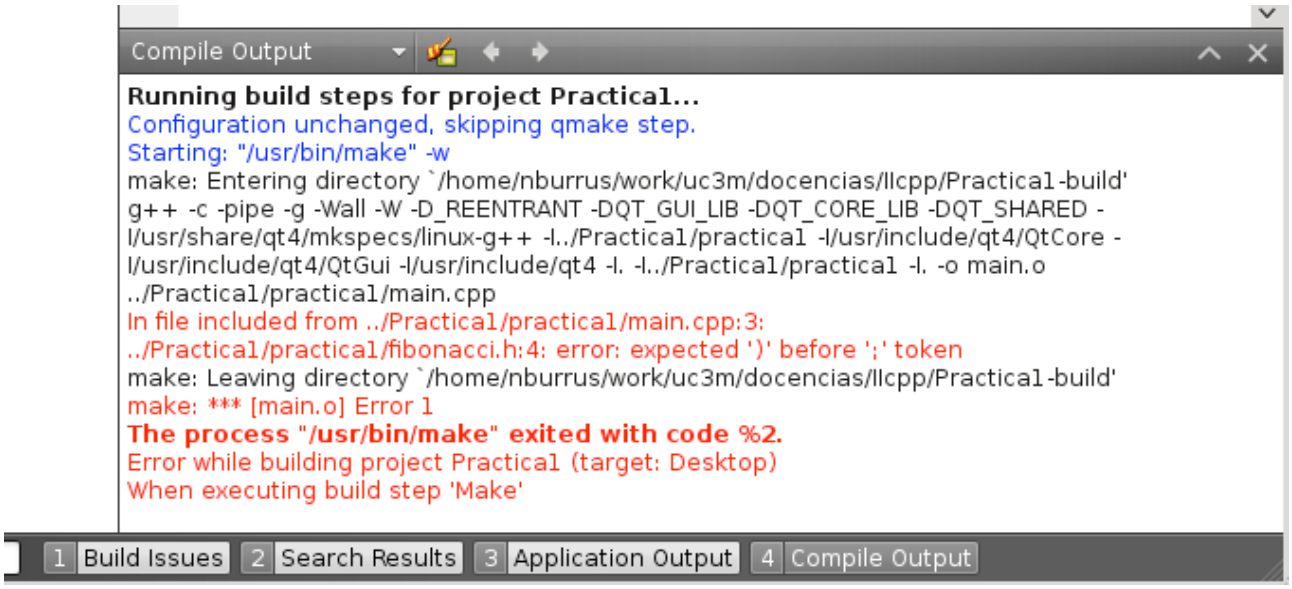
2. Implementar esta función en ***fibonacci.cpp***.
3. Además, implementar una función ***main*** en el fichero ***main.cpp***. El programa debe pedir un número por teclado y mostrar su secuencia de Fibonacci.
4. Mientras se edita el código, existen varios atajos de teclado muy útiles a la hora del manejo del entorno:
   4.1. ***Ctrl+Espacio***: completa automáticamente la palabra actual. Por ejemplo, tecleando "fib + Ctrt+Espacio" en el ***main.cpp*** el editor debería proponer "fibonacci" y recordar los argumentos de la función.
   4.2. ***F2***: permite saltar a la definición de una función o de cualquier símbolo.
   4.3. ***F4***: sirve para pasar del fichero de cabecera al fichero fuente que le corresponde, por ejemplo desde *fibonacci.h* hacia *fibonacci.cpp*.
   4.4. ***Ctrl+I***: sirve para alinear automáticamente una región de código.

4. Compilar

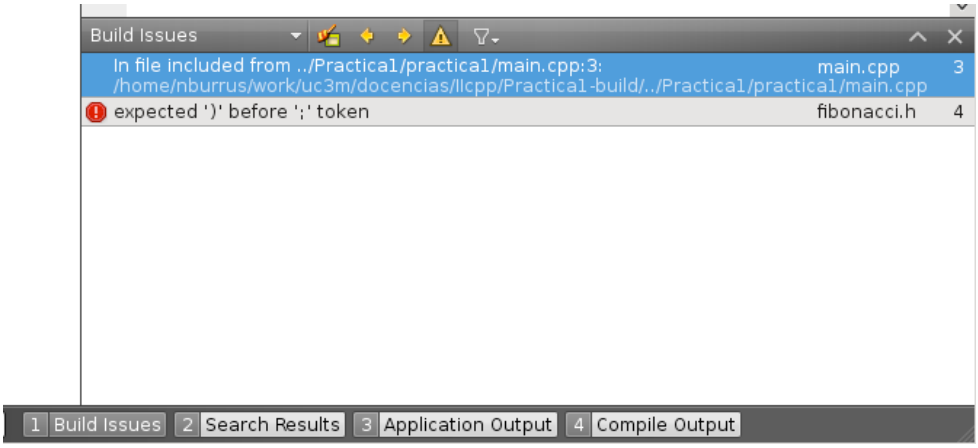
Para compilar el programa se puede hacer clic en el icono del *martillo* (abajo izquierda), utilizando el atajo de teclado ***Ctrl+B*** o a través del menú ***Build→Build All***.

Para ver el resultado de la compilación y ver así si se ha realizado correctamente, abrir la pestaña *Compile Output* (abajo izquierda, opción 4) pulsando sobre el icono o utilizar el atajo de teclado ***Alt+4***.

En esta ventana se puede ver el detalle de la salida de la compilación ejecutada por QtCreator.



En caso de que exista un problema de compilación, se pueden ver los mensajes de error o advertencia aquí aunque QtCreator también proporciona una vista especial para los errores en la pestaña *Build Issues (abajo izquierda, opción 1)*.



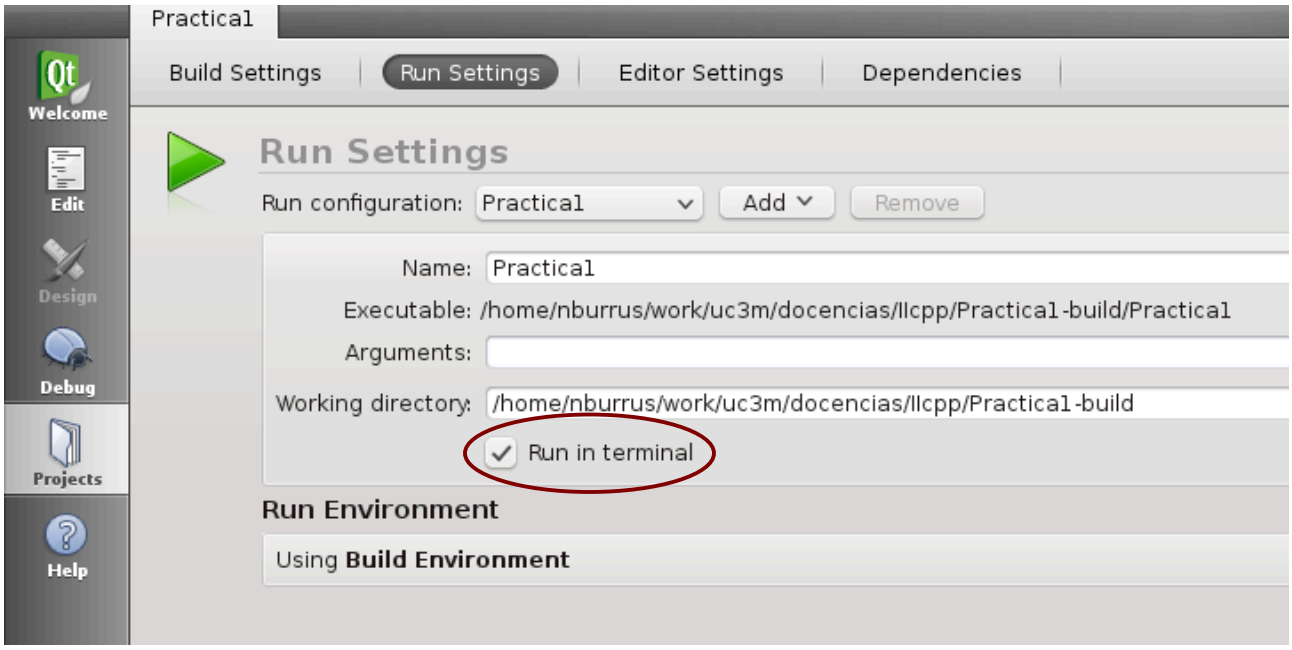
Al hacer clic sobre alguno de los errores se abrirá directamente el fichero en la línea correspondiente al problema.

5. Ejecutar

Para ejecutar un programa se puede hacer clic en el *triángulo verde* (abajo izquierda), utilizando el atajo de teclado ***Ctrl+R***, o yendo al menú ***Build→Run***.

La salida del programa se mostrará en la pestaña *Application Output (abajo izquierda, opción 3)*.

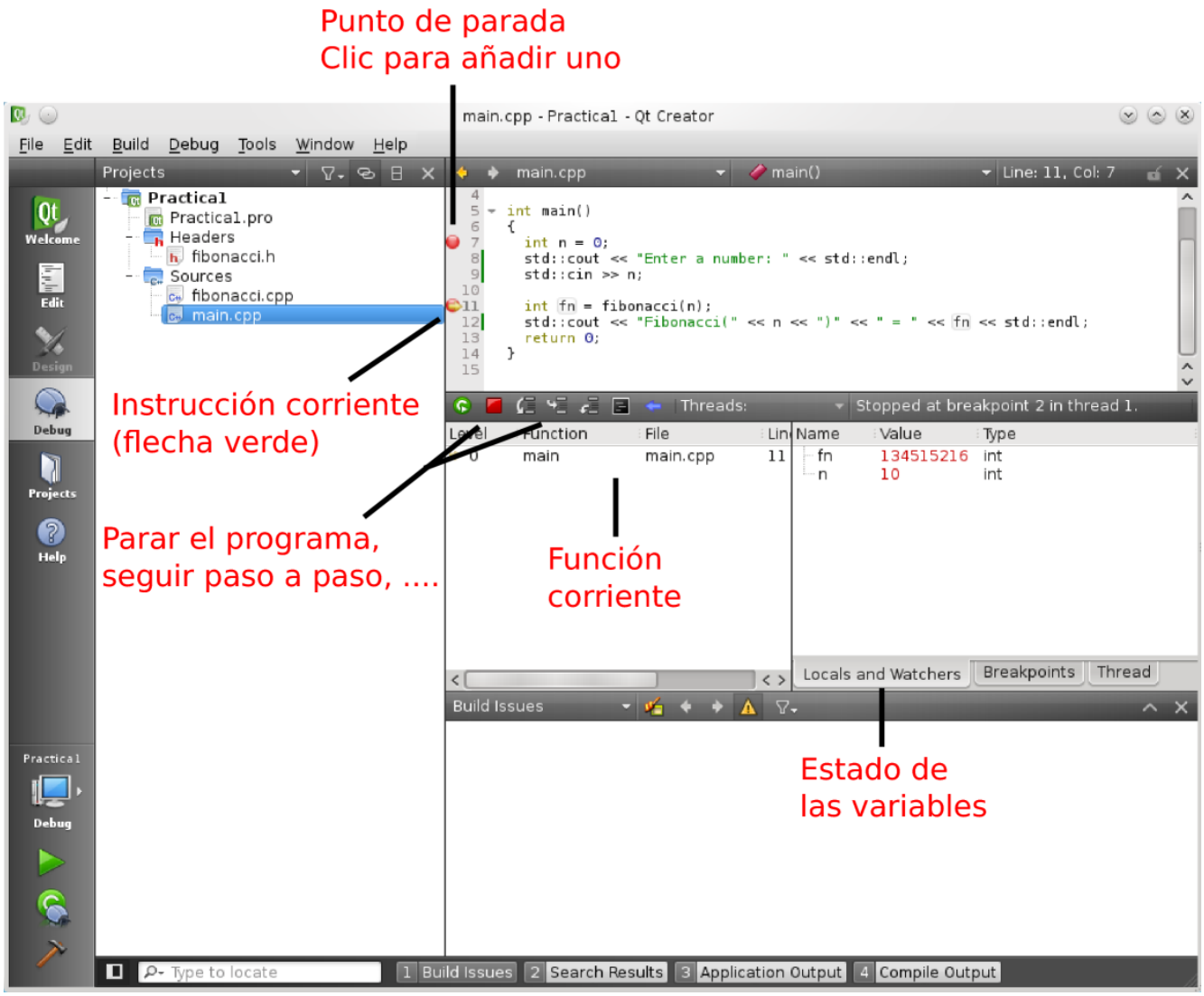
En caso de que el programa tenga interacciones con el teclado, será necesario lanzarlo a parte en una terminal. Para ello, habrá que configurar los parámetros del proyecto en el menú de la izquierda, ***Projects→Run Settings***, y seleccionar (run in terminal)



6. Para avanzados. Depuración de código

QtCreator dispone de una interfaz integrada del depurador **GDB**. Se puede activar el modo de depuración haciendo clic en el triángulo verde que tiene un *icono de escarabajo* (símbolo del depurador), mediante el atajo de teclado ***F5*** o a través del menú ***Debug→Start debugging→Start debugging***.

Para abrir la ventana de depuración, seleccionar la pestaña *Debug* (abajo izquierda) y aparecerá el entorno similar a como se muestra en la siguiente figura.



7. Ejercicios

Hasta ahora hemos programado en C++ utilizando las mismas funciones que en C. Esto es posible porque C++ es compatible con C. Sin embargo, C++ viene con su propia librería estándar, más potente y completa. En particular, las entradas/salidas se manejan de manera diferente utilizando los operadores de flujo "<<" y ">>".

7.1. Ejercicio 1

El siguiente programa es un ejemplo de "Hola mundo" en C++ (izquierda) y su equivalente en C (derecha):

```
#include <iostream> // std::cout

int main() {
      std::cout << "hola mundo\n";
}
```

```
#include <stdio.h> /* printf */

int main() {
      printf("hola mundo\n");
}
```

7.2. Ejercicio 2

Este programa muestra un mensaje en el terminal mediante ***std::cout*** que representa la salida estándar de programa. Después se utiliza el operador "<<" para enviar datos hacia el flujo de salida. Los datos se pueden enviar en cadena y sin importar el tipo.

```
#include <iostream> // cabecero para std::cout

int main() {
      int i = 42;
      float f = 42.5;

      std::cout << "Entero: " << i << " Float: " << f << std::endl;
      // endl es fin de línea
}
```

La lectura de datos desde la entrada estándar se hace con ***std::cin***:

```
#include <iostream> // cabecera para std::cout, std::cin

int main() {
      int n;

      std::cout << "Introduce un numero:" << std::endl;
      std::cin >> n;
      std::cout << "El numero es " << n << "\n";
}
```

Para leer se utiliza el operador de flujo inverso "**>>**". También se puede utilizar en cadena y puede leer varios tipos de datos:

```
#include <iostream> // cabecera para std::cout, std::cin

int main() {
      int n;
      float f;

      std::cout << "Introduce un numero entero y uno flotante:" << std::endl;
      std::cin >> n >> f;
      std::cout << "Entero: " << n << " Float: " << f << std::endl;
}
```

7.3. Ejercicio 3

Transformar el programa de fibonacci para que use las funciones estándar de C++ en

caso de que no lo hiciese ya.

7.4. Ejercicio 4. Espacio de nombres

El prefijo ***std::*** que precede a ***cout*** y ***cin*** viene de una funcionalidad de C++ que se llama *espacio de nombres*. Esto permite definir varias funciones con el mismo nombre, poniéndolas en espacios diferentes.

Todas la funciones de la librería estándar están en el espacio de nombres ***std***. Para evitar teclear siempre su prefijo, se puede añadir la siguiente línea para que el compilador busque automáticamente dichas funciones.

```
#include <iostream>

using namespace std; // esta línea permite omitir el prefijo std::

int main() {
      cout << "Hola sin prefijo" << endl;
}
```

7.5. Ejercicio 5. Propuesto

Escribir un programa que lea 10 valores enteros por teclado, los guarde en un vector y los muestre por pantalla. Los enteros deben cumplir la condición de ser un número par positivo. No se admitirá el valor 0.