

Práctica Inicial (P0)

Programación II · Curso 2021-22

Facultad de Informática



Índice

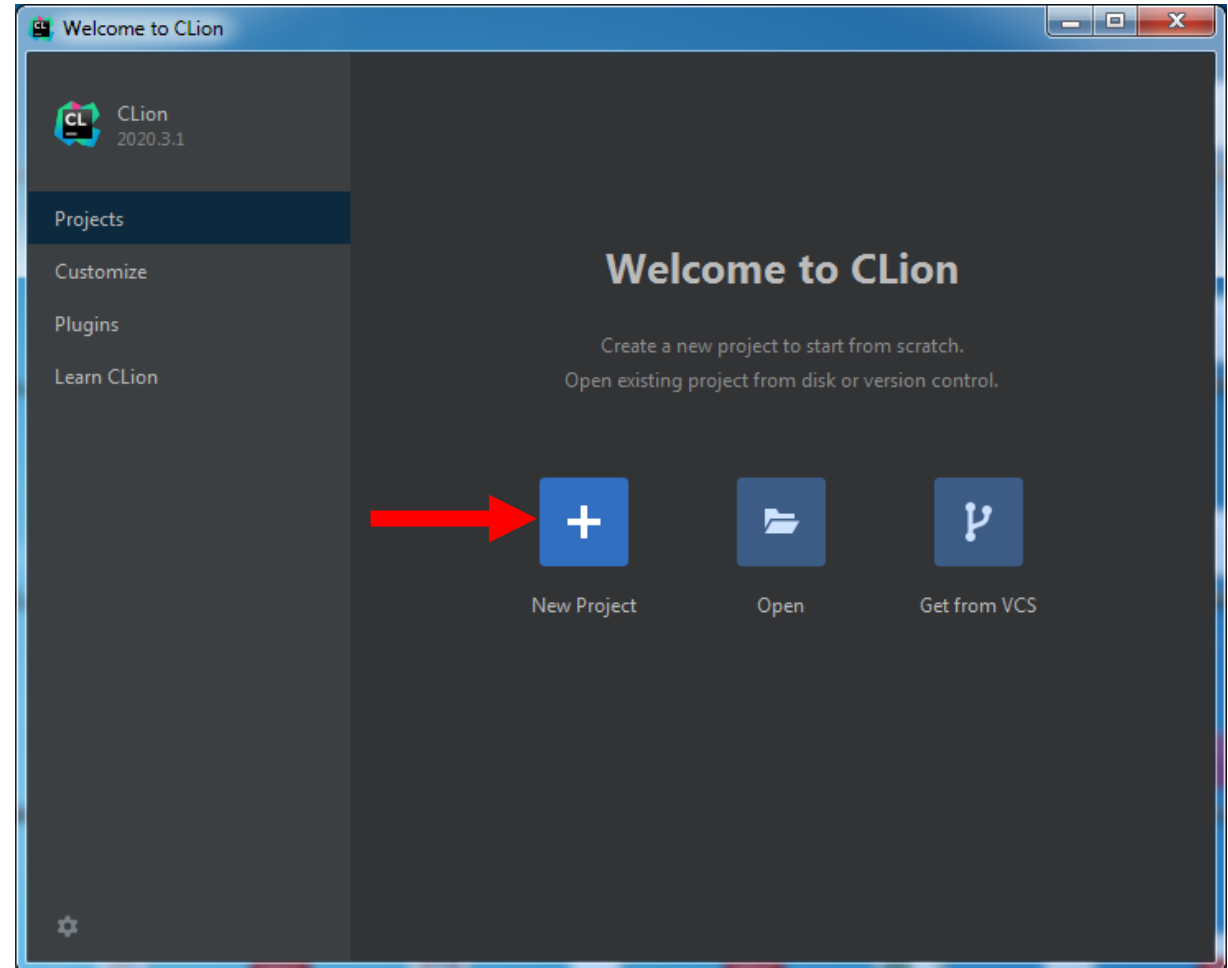
- Tutorial de Strings (cadenas.pdf)
- Trabajando con módulos: ejemplo Rational
 - Compilación en Clion
 - Depuración en Clion
 - Compilación contra la máquina de referencia con SSH

Ejemplo en CLion: Rational

- Vamos a realizar un único programa (`main.c`) que se ejecuta con dos versiones de código diferente:
 - Implementación con una estructura (`rational_struct`)
 - Implementación con punteros (`rational_pointer`)

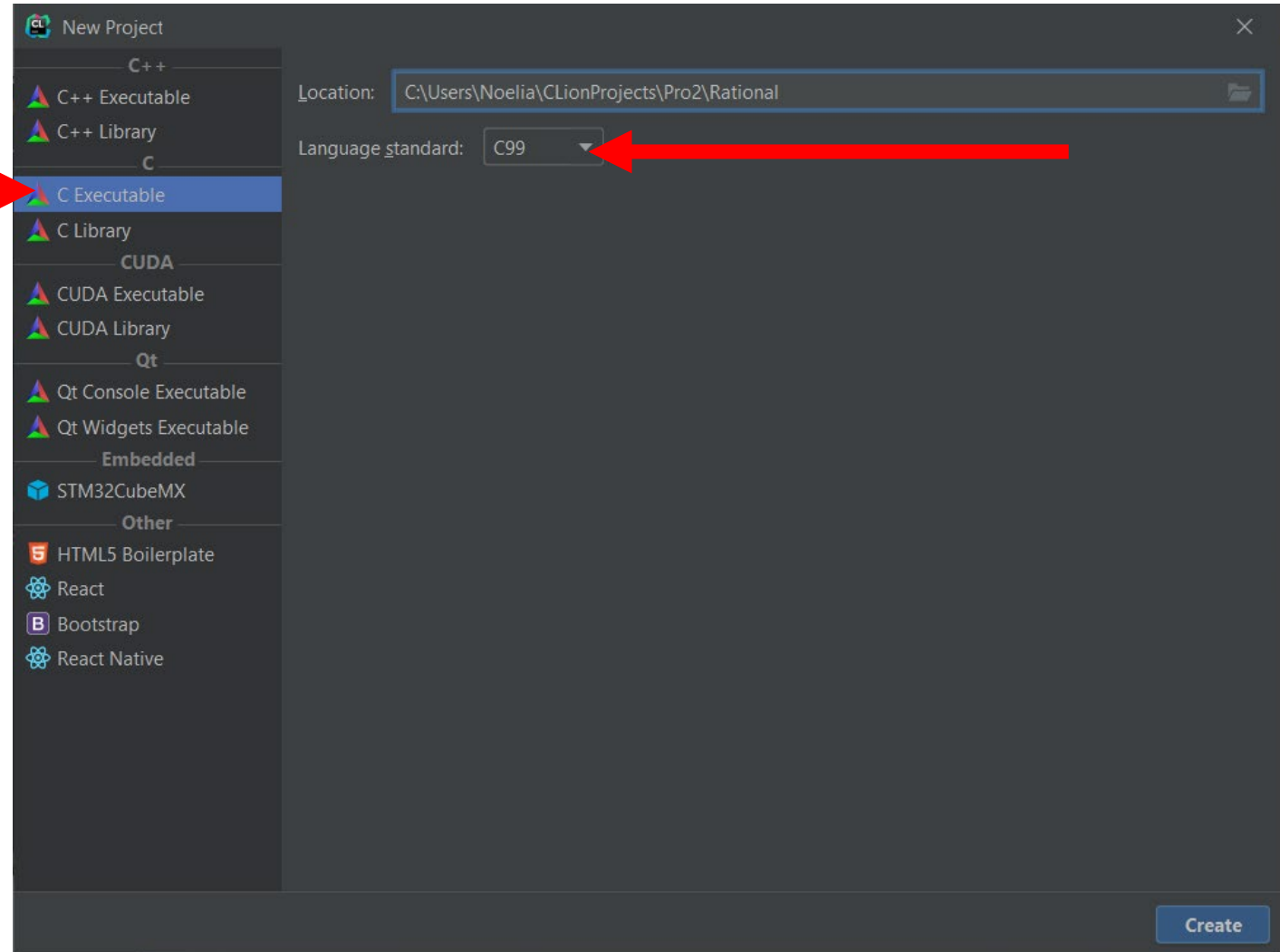
Crear un proyecto nuevo

- En Clion seleccionamos la opción de Nuevo Proyecto (*New Project*)



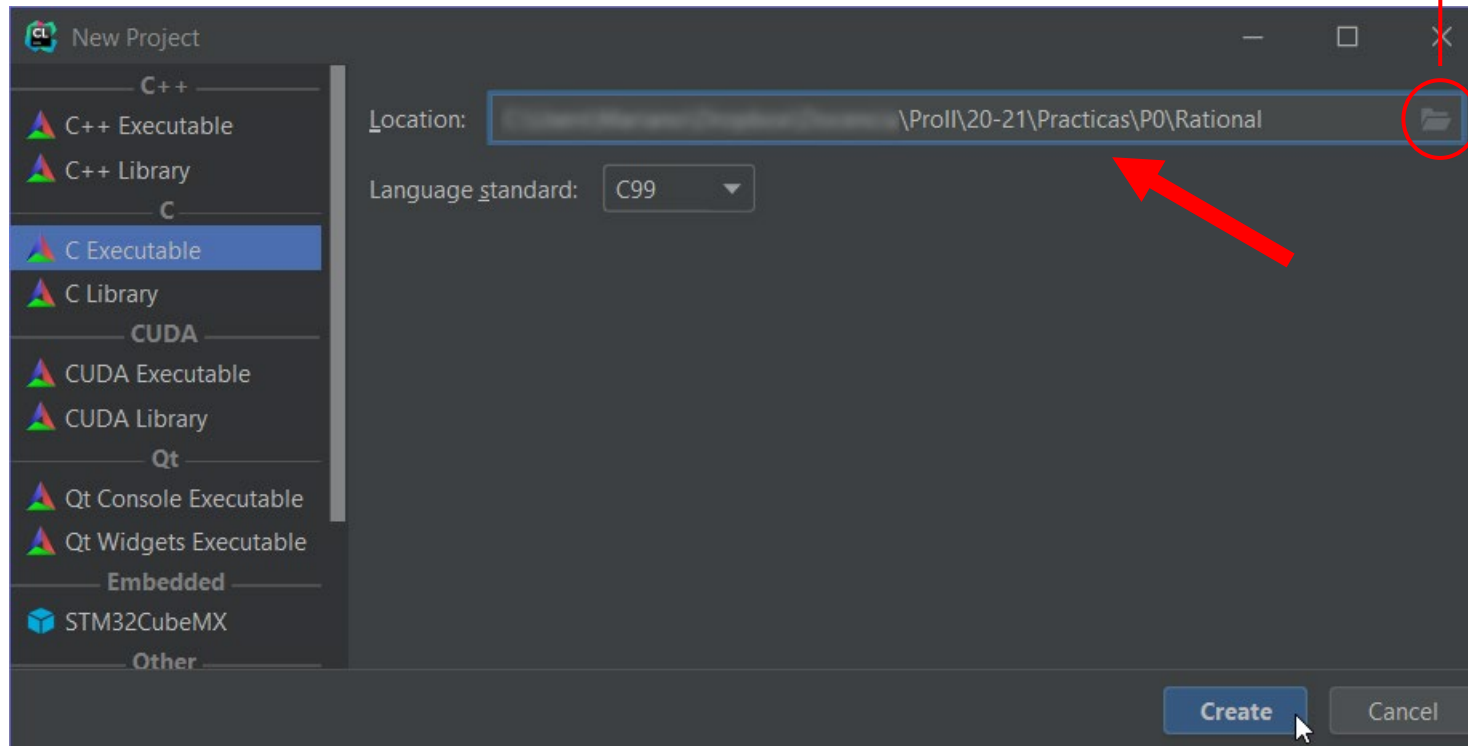
Selección del tipo de proyecto

- Proyecto ejecutable en C
(*C Executable*)
- Seleccionamos el estándar del lenguaje: C99



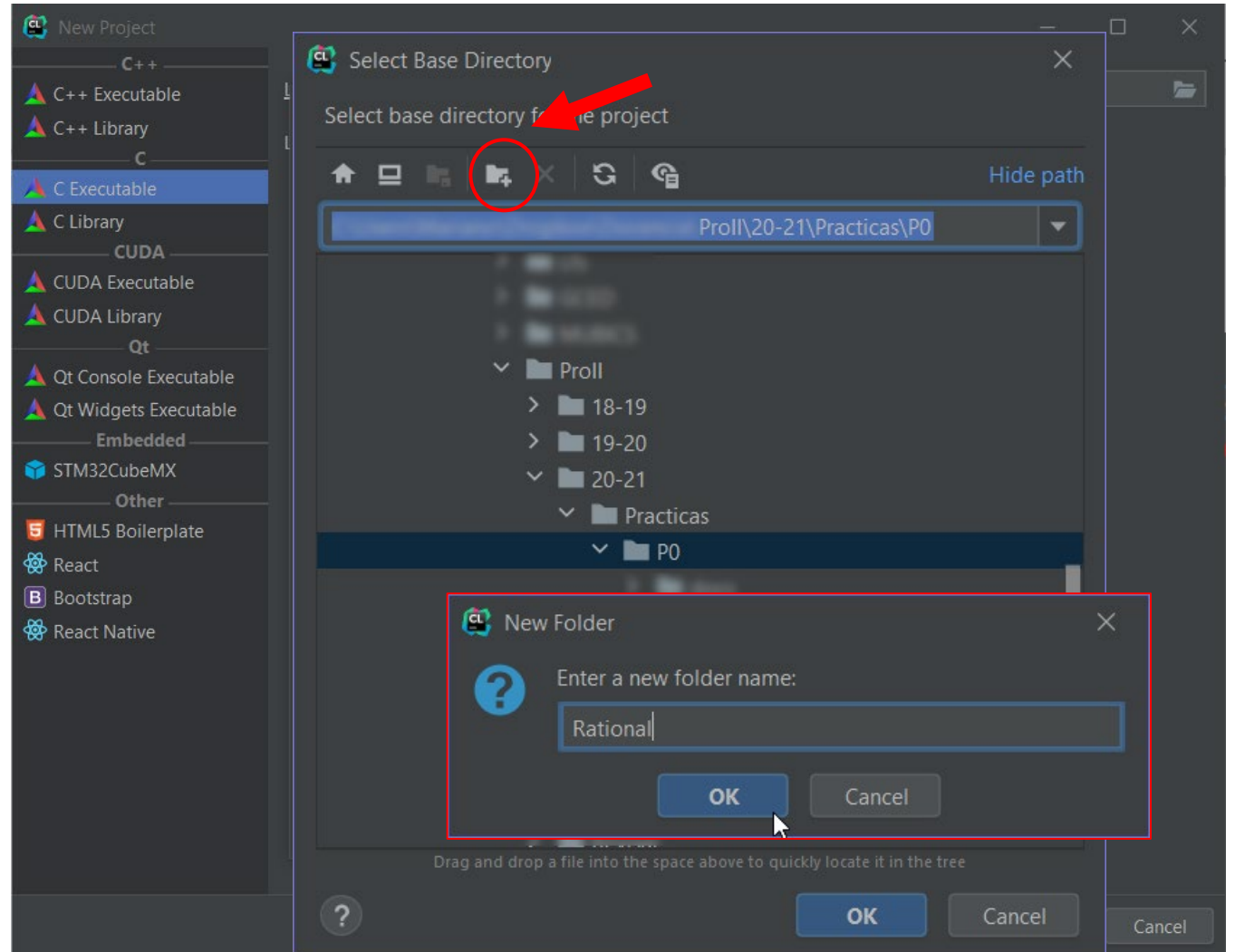
Selección de la carpeta del proyecto

- Le indicamos el directorio donde guardarlo
- Para elegir la carpeta, clic



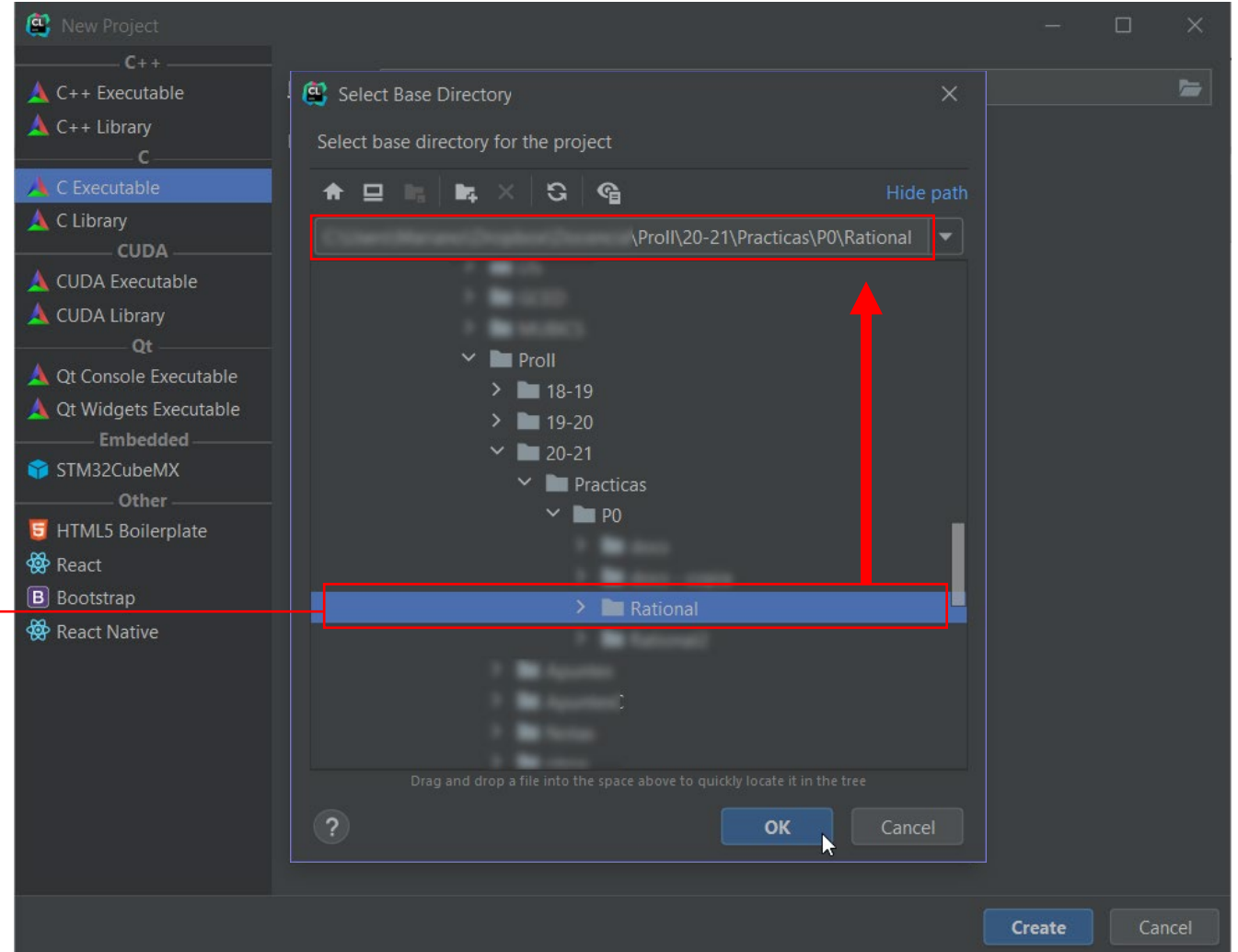
Selección de la carpeta del proyecto

- Creamos la carpeta si no existe



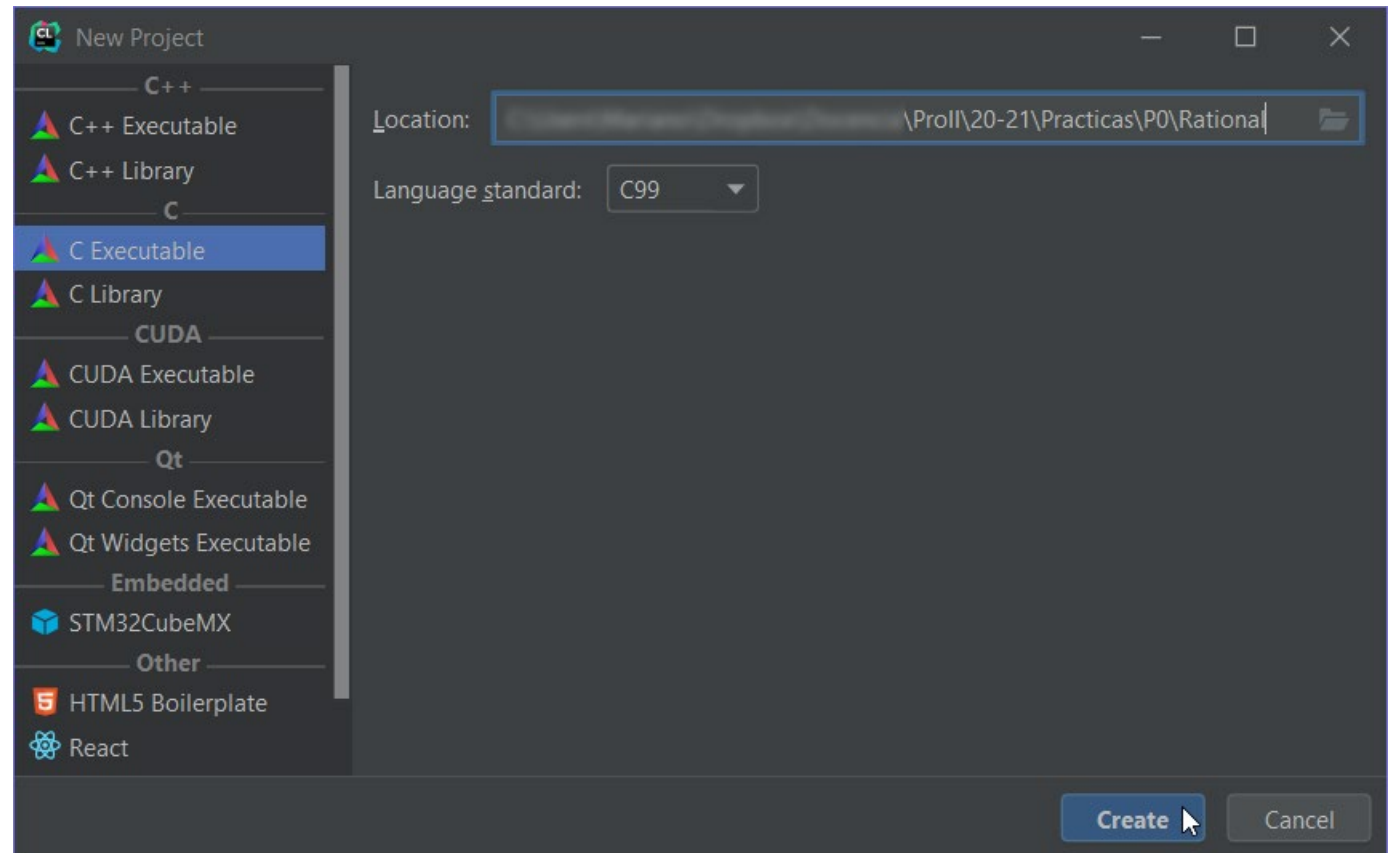
Selección de la carpeta del proyecto

- Hacer clic y la carpeta queda seleccionada



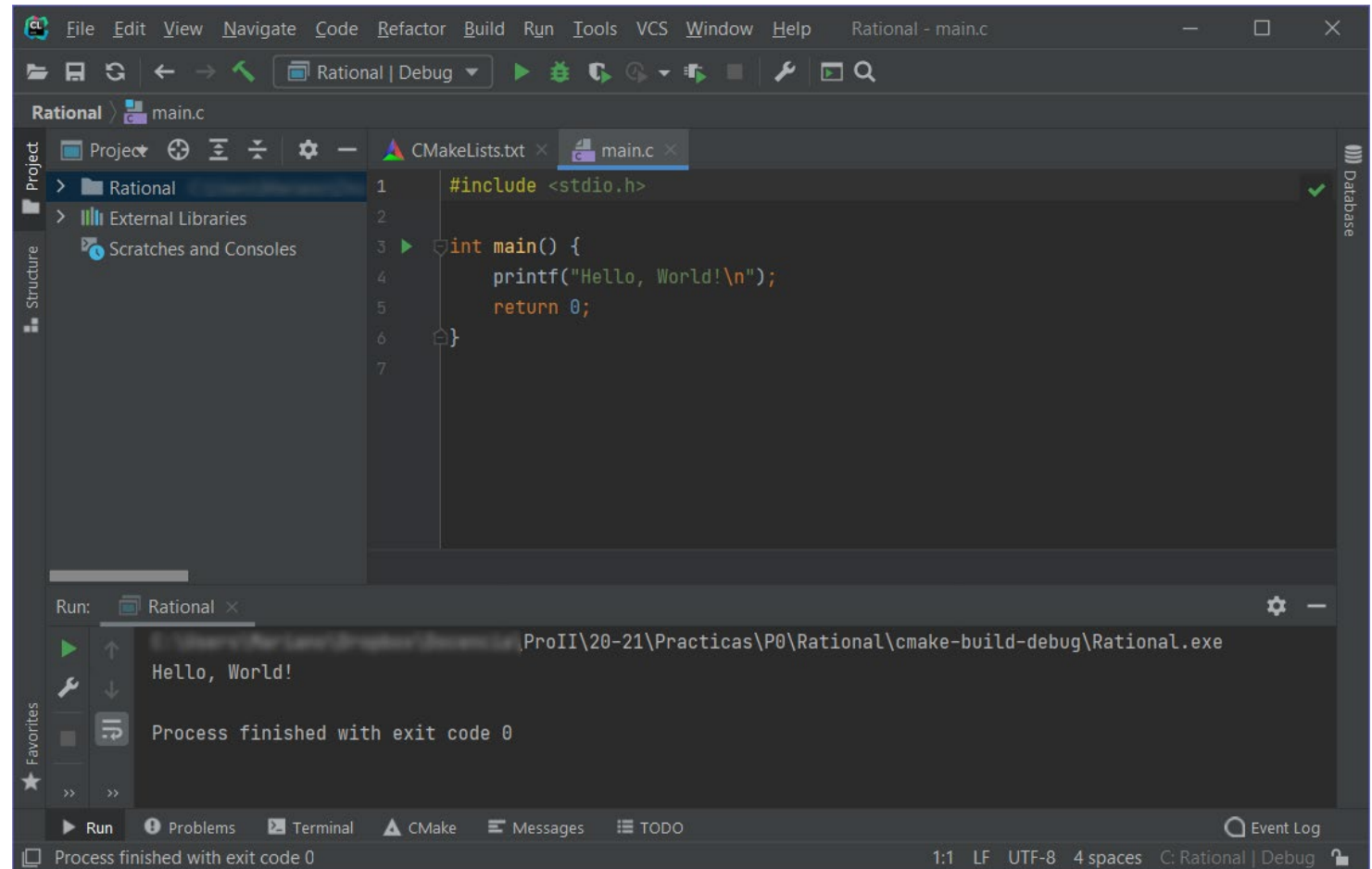
Crear un proyecto nuevo

- Pulsando *Create* se crea un nuevo proyecto



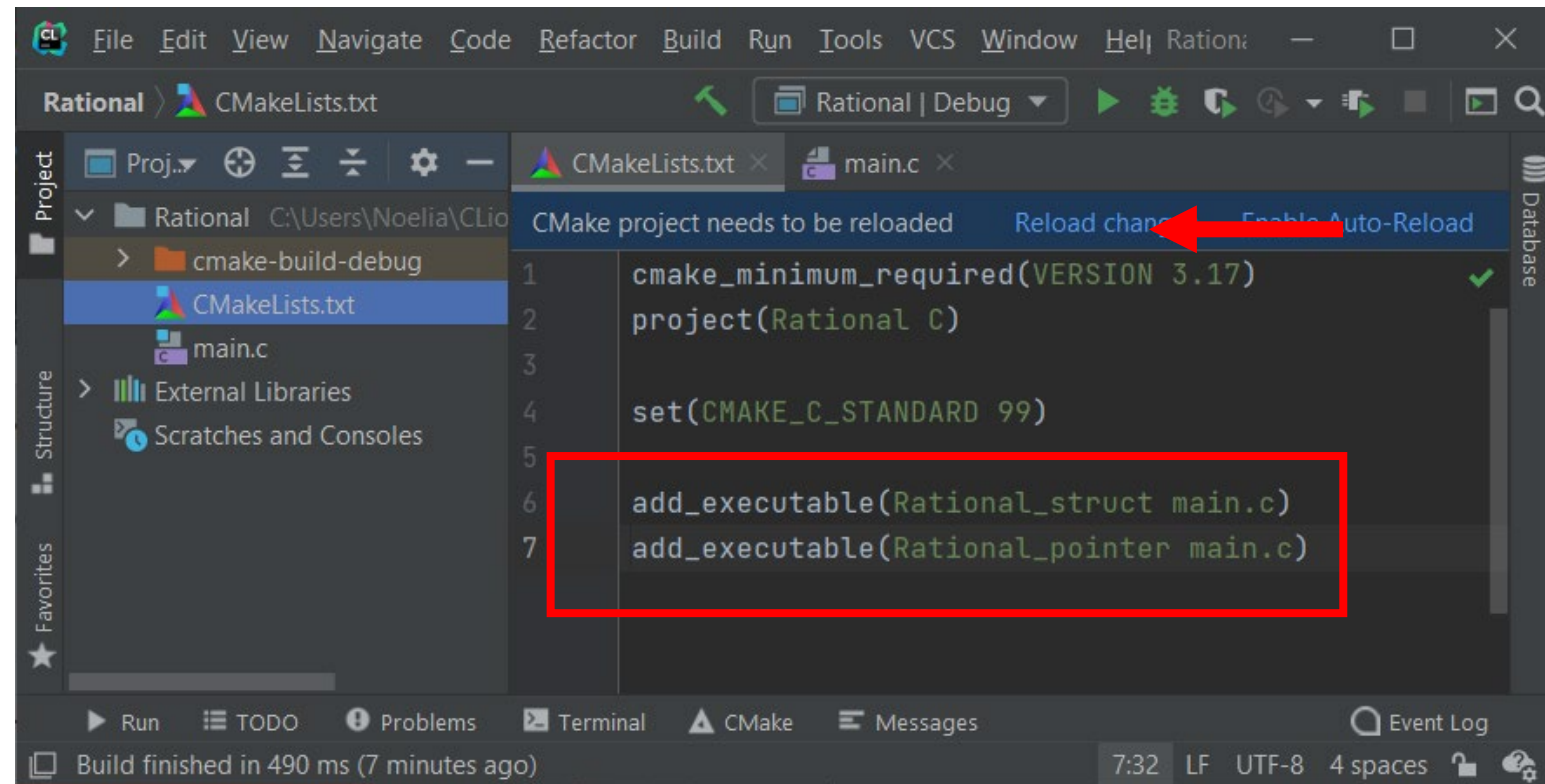
Proyecto nuevo creado

- El fichero `main.c` creado por defecto muestra el mensaje “Hello world” por pantalla



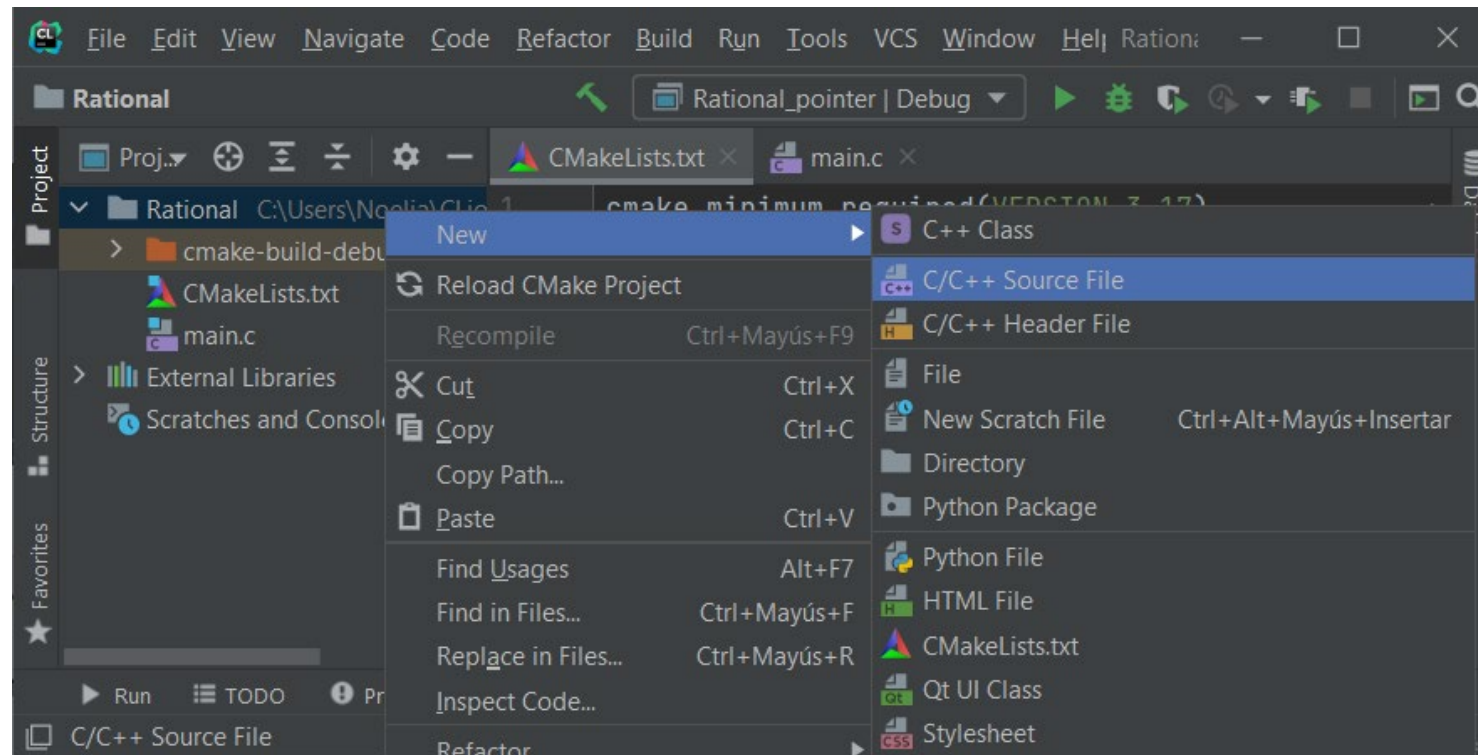
Modificamos el fichero CMakeLists.txt

- Añadimos 2 opciones de ejecución:
 - Rational_struct
 - Rational_pointer
- Ambas usan main.c
- Pulsamos *Reload_changes* para que los cambios tengan efecto



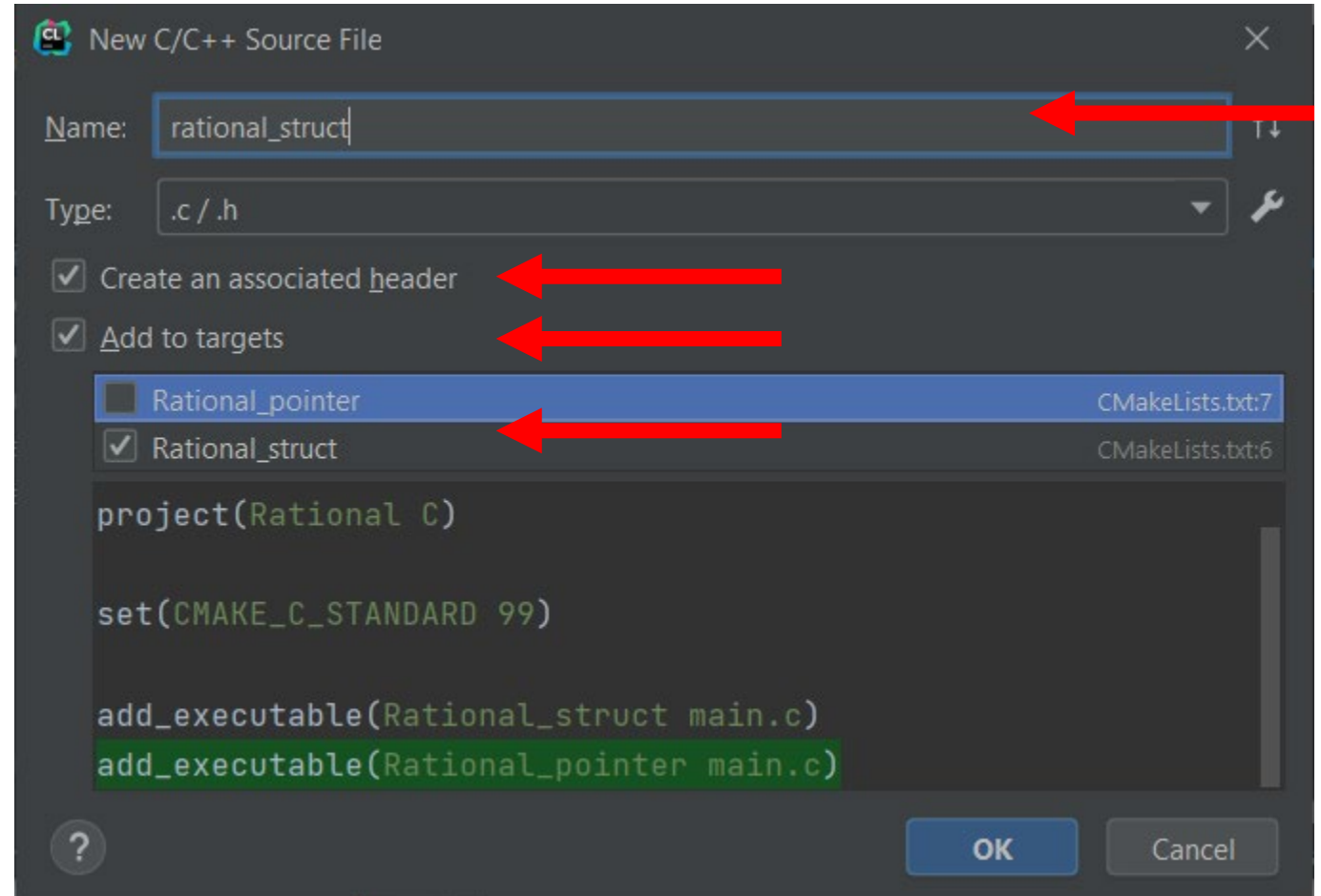
Creamos la primera implementación

- Pulsando el botón del derecho del ratón sobre el proyecto seleccionamos *New → C/C++ Source File*



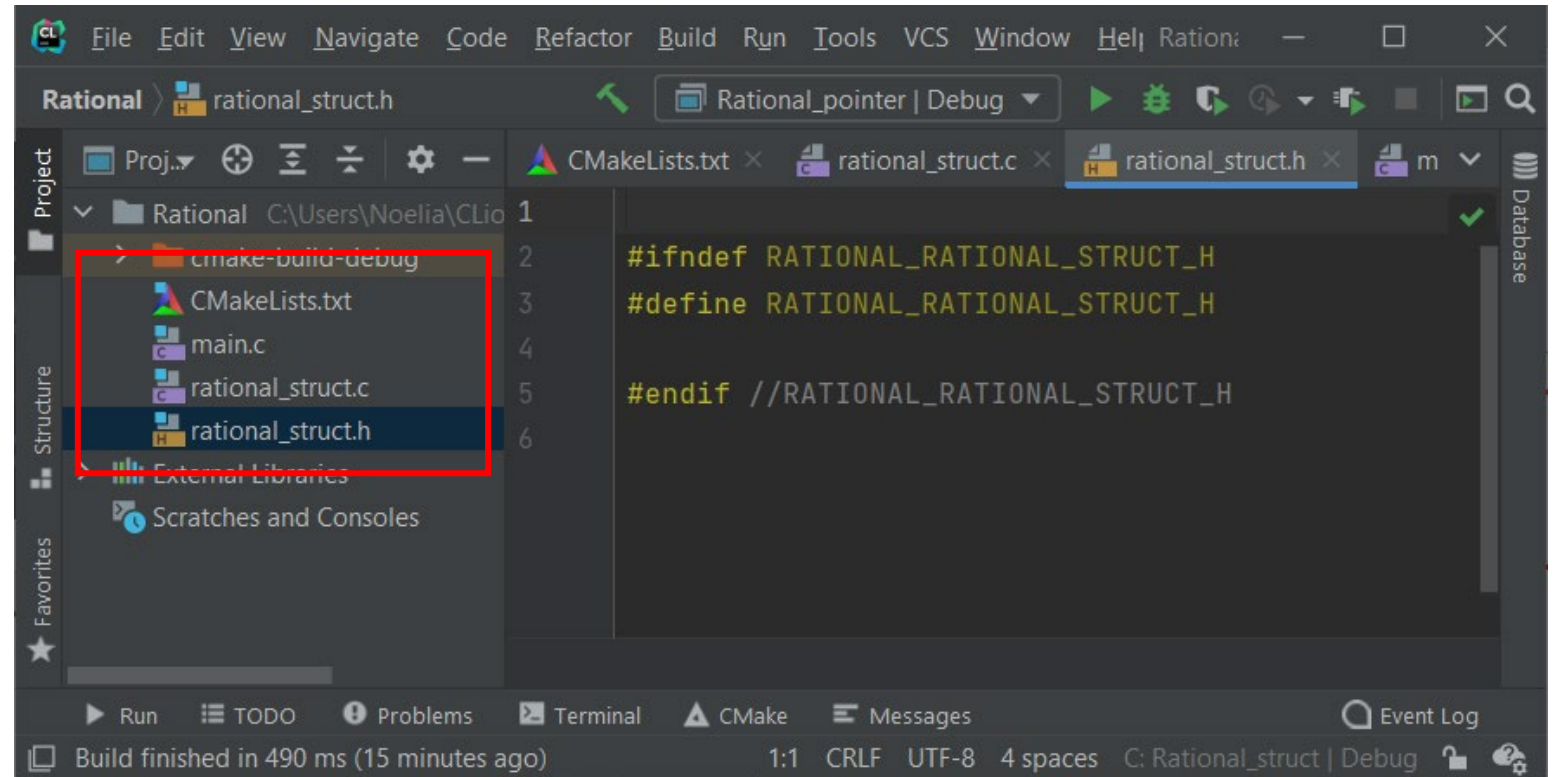
Propiedades del fichero de implementación

- Damos nombre al fichero
- Indicamos el tipo (.c)
- Activamos la opción de crear una librería asociada (se añadirá un .h al tipo)
- La incluimos solo en el *target rational_struct*



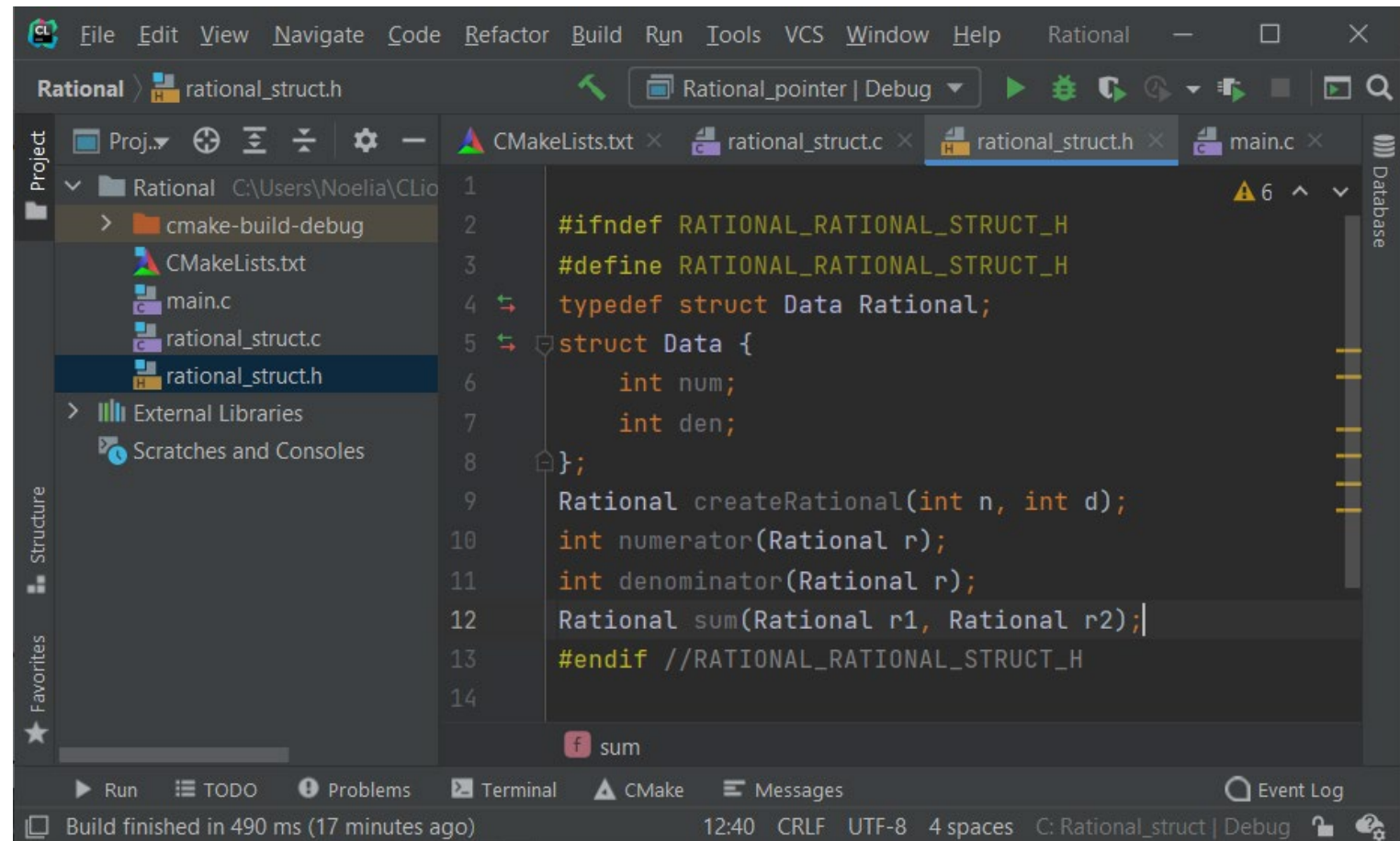
Ficheros del proyecto

- CMakeLists.txt
- main.c
- rational_struct.c
- rational_struct.h



rational_struct.h: Declaraciones de tipo y operaciones

- Incluimos en la librería (.h) las declaraciones de tipos y operaciones

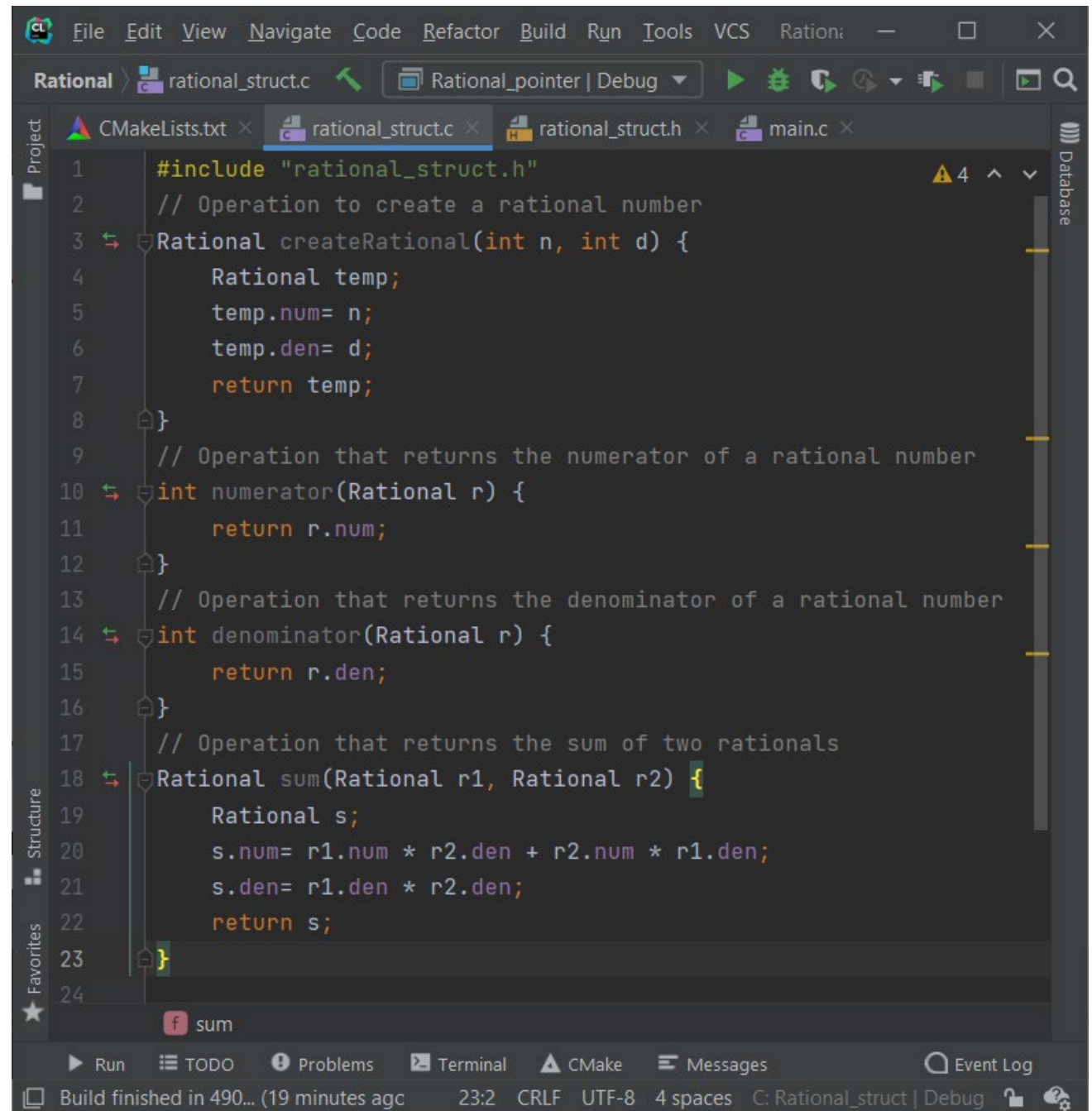


```
1
2  #ifndef RATIONAL_RATIONAL_STRUCT_H
3  #define RATIONAL_RATIONAL_STRUCT_H
4  typedef struct Data Rational;
5  struct Data {
6      int num;
7      int den;
8  };
9  Rational createRational(int n, int d);
10 int numerator(Rational r);
11 int denominator(Rational r);
12 Rational sum(Rational r1, Rational r2);
13 #endif //RATIONAL_RATIONAL_STRUCT_H
14
```

The screenshot shows a code editor window titled 'Rational' with the file 'rational_struct.h' open. The code defines a 'Rational' type as a struct with 'num' and 'den' fields, and declares functions for creating, accessing, and summing rational numbers. The IDE interface includes a project explorer on the left, a toolbar at the top, and a status bar at the bottom.

rational_struct.c: Implementación

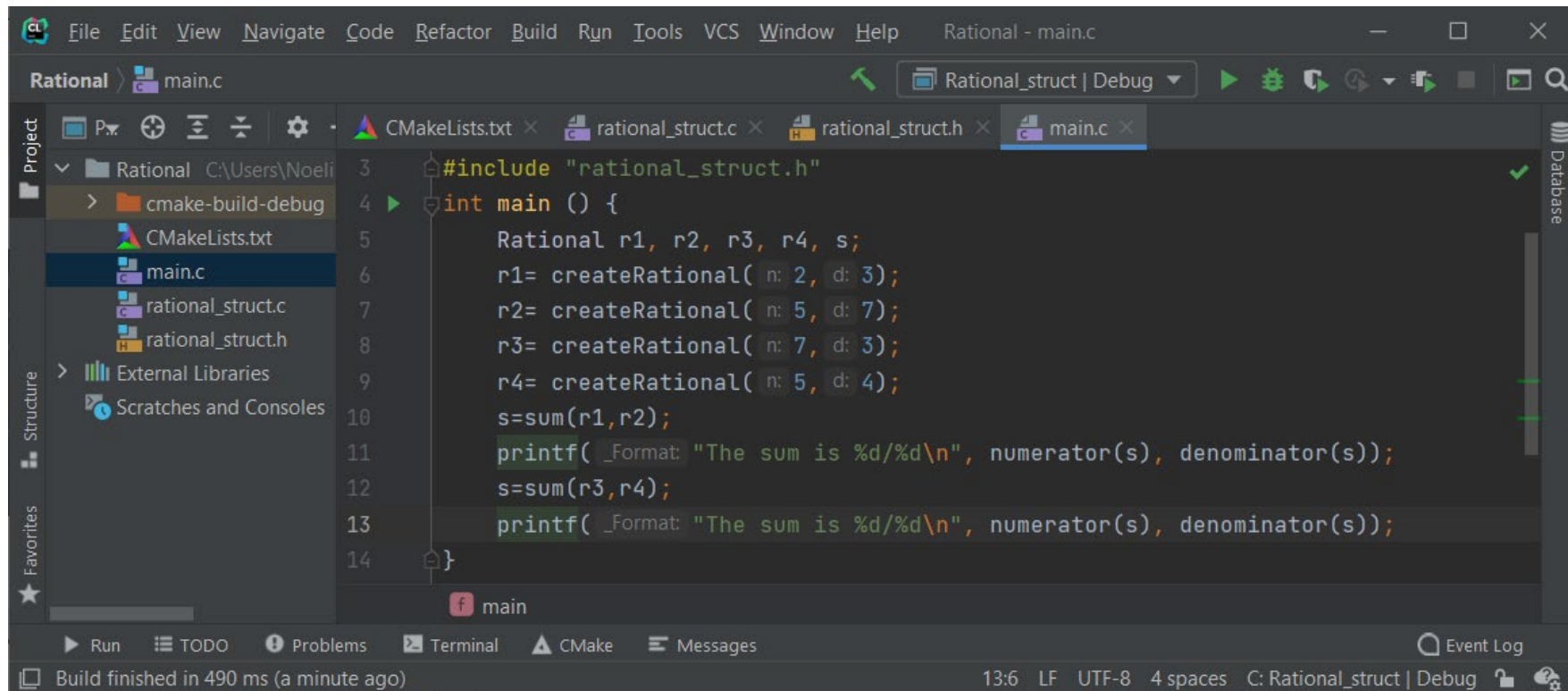
- Incluimos el código de cada operación en el fichero .c



```
1  #include "rational_struct.h"
2  // Operation to create a rational number
3  Rational createRational(int n, int d) {
4      Rational temp;
5      temp.num= n;
6      temp.den= d;
7      return temp;
8  }
9  // Operation that returns the numerator of a rational number
10 int numerator(Rational r) {
11     return r.num;
12 }
13 // Operation that returns the denominator of a rational number
14 int denominator(Rational r) {
15     return r.den;
16 }
17 // Operation that returns the sum of two rationals
18 Rational sum(Rational r1, Rational r2) {
19     Rational s;
20     s.num= r1.num * r2.den + r2.num * r1.den;
21     s.den= r1.den * r2.den;
22     return s;
23 }
```


Creación de un programa de prueba

- Incluimos código para probar `rational_struct` en `main.c`



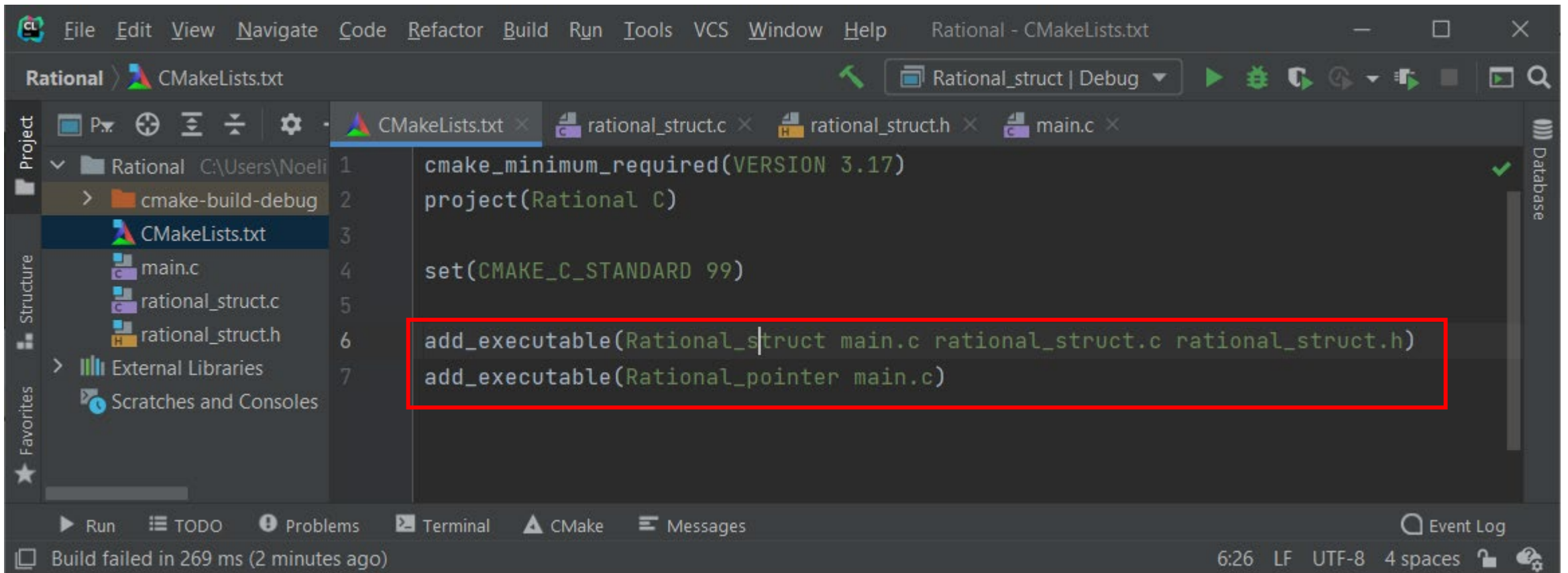
The screenshot shows an IDE window titled "Rational - main.c". The left sidebar displays the project structure with files: CMakeLists.txt, rational_struct.c, rational_struct.h, and main.c. The main editor area shows the following C code:

```
1 #include "rational_struct.h"
2
3 int main () {
4     Rational r1, r2, r3, r4, s;
5     r1= createRational( n: 2, d: 3);
6     r2= createRational( n: 5, d: 7);
7     r3= createRational( n: 7, d: 3);
8     r4= createRational( n: 5, d: 4);
9     s=sum(r1,r2);
10    printf( _Format: "The sum is %d/%d\n", numerator(s), denominator(s));
11    s=sum(r3,r4);
12    printf( _Format: "The sum is %d/%d\n", numerator(s), denominator(s));
13
14 }
```

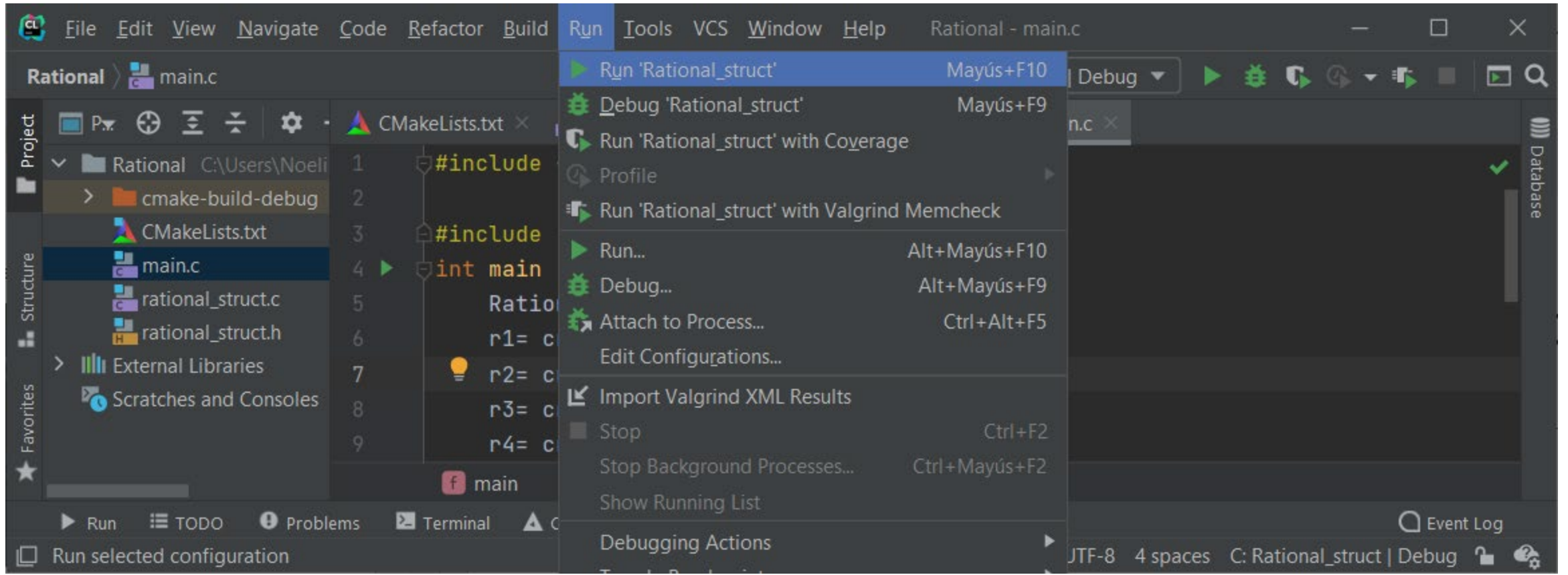
The bottom status bar indicates "Build finished in 490 ms (a minute ago)" and "13:6 LF UTF-8 4 spaces C: Rational_struct | Debug".

Cambios automáticos en CMakeLists.txt

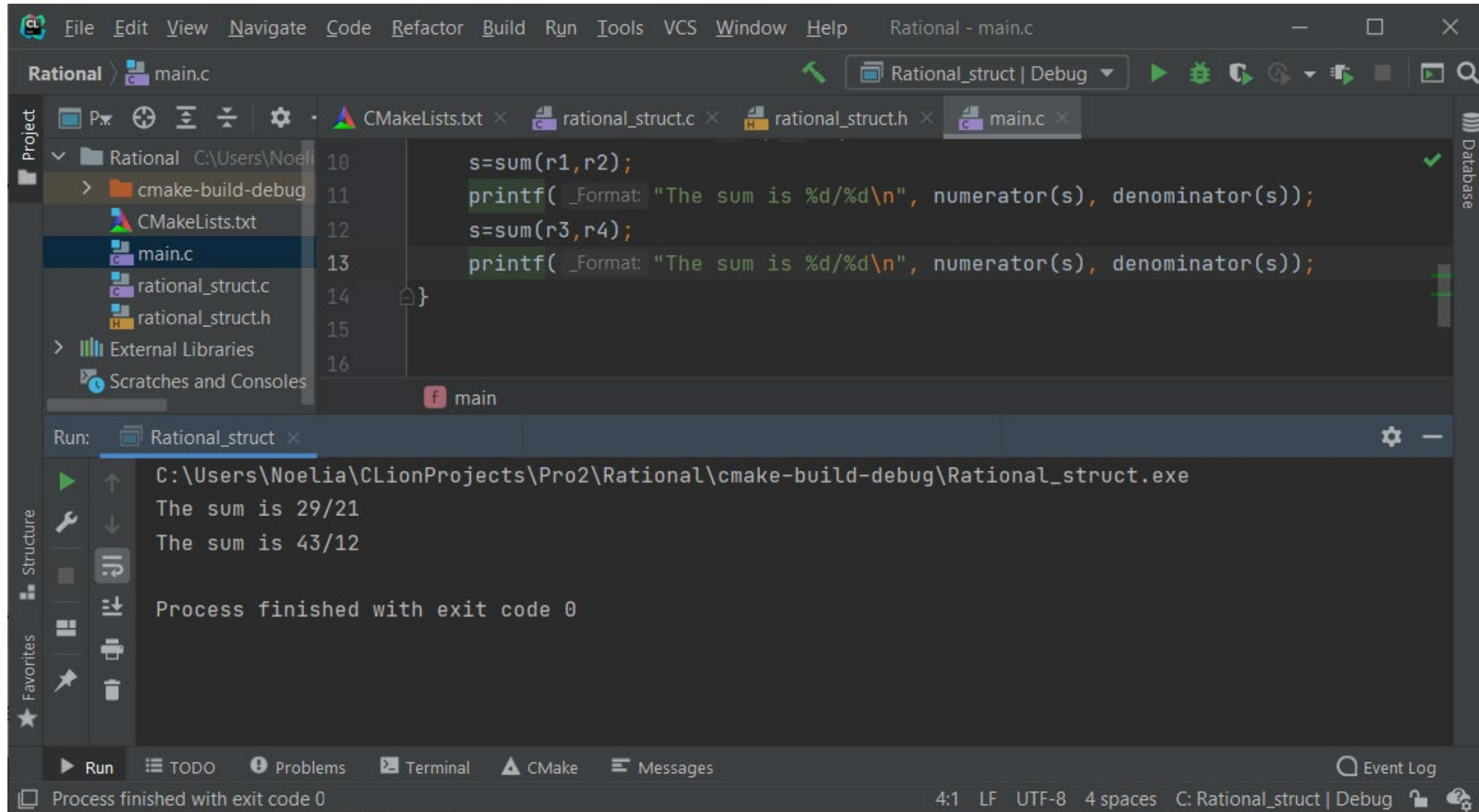
- Los dos ficheros creados (.h y .c) se han añadido al *target* rational_struct



Compilamos el proyecto



Resultado de ejecución



The screenshot shows the CLion IDE interface with the following components:

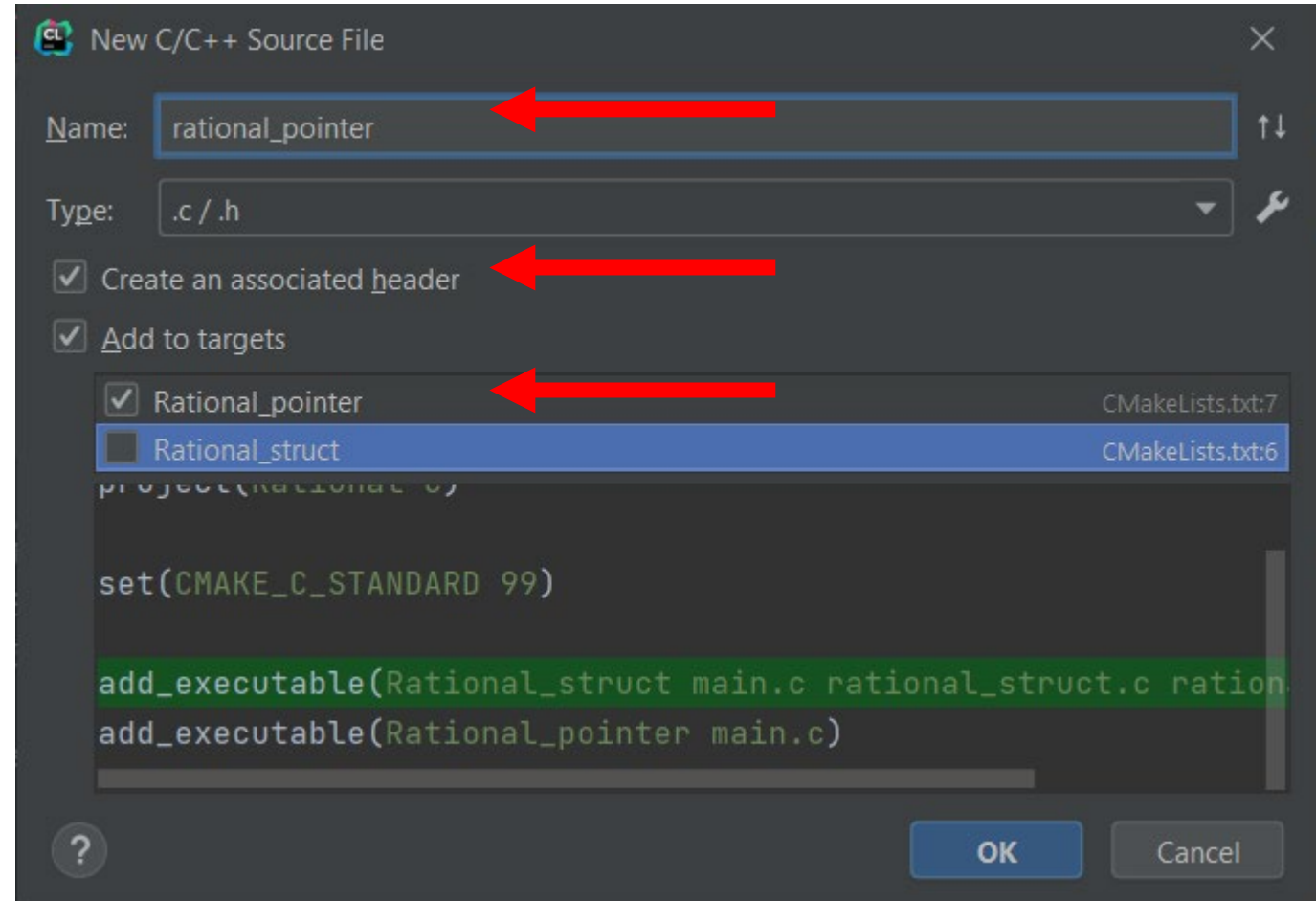
- Project Explorer:** Displays the project structure, including `Rational`, `cmake-build-debug`, `CMakeLists.txt`, `main.c`, `rational_struct.c`, and `rational_struct.h`.
- Editor:** Shows the `main.c` file with the following code:

```
10 s=sum(r1,r2);  
11 printf( _Format: "The sum is %d/%d\n", numerator(s), denominator(s));  
12 s=sum(r3,r4);  
13 printf( _Format: "The sum is %d/%d\n", numerator(s), denominator(s));  
14 }  
15  
16
```
- Run Window:** Shows the execution output for `Rational_struct`. The output is:

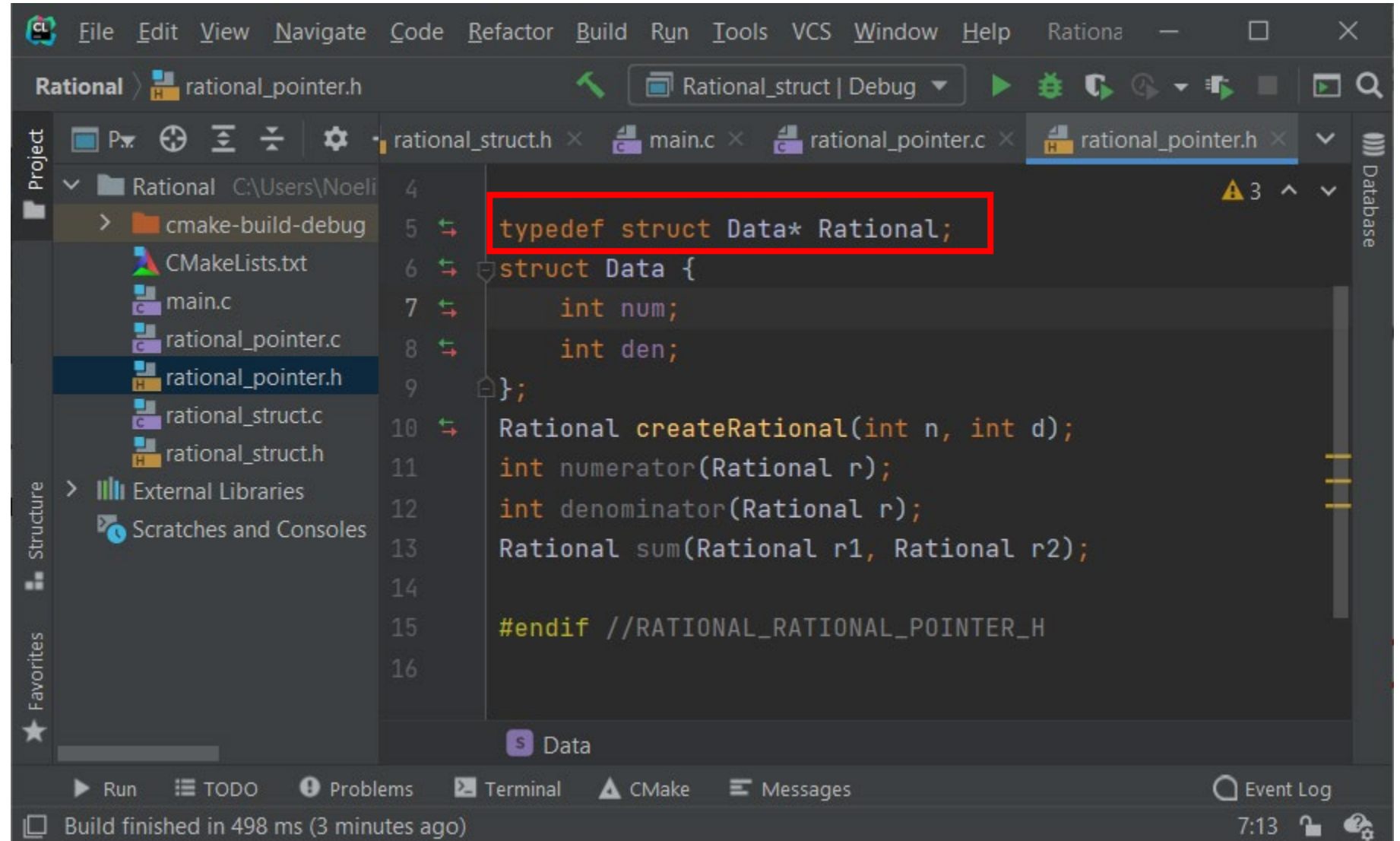
```
C:\Users\Noelia\CLionProjects\Pro2\Rational\cmake-build-debug\Rational_struct.exe  
The sum is 29/21  
The sum is 43/12  
Process finished with exit code 0
```
- Bottom Panel:** Includes tabs for `Run`, `TODO`, `Problems`, `Terminal`, `CMake`, and `Messages`. The status bar at the bottom indicates the file encoding is UTF-8 and the workspace is `C: Rational_struct | Debug`.

Incluyendo una nueva implementación

- Añadimos un nuevo fichero de código (*rational_pointer*) con su librería asociada
- Lo añadimos solo al target *rational_pointer*



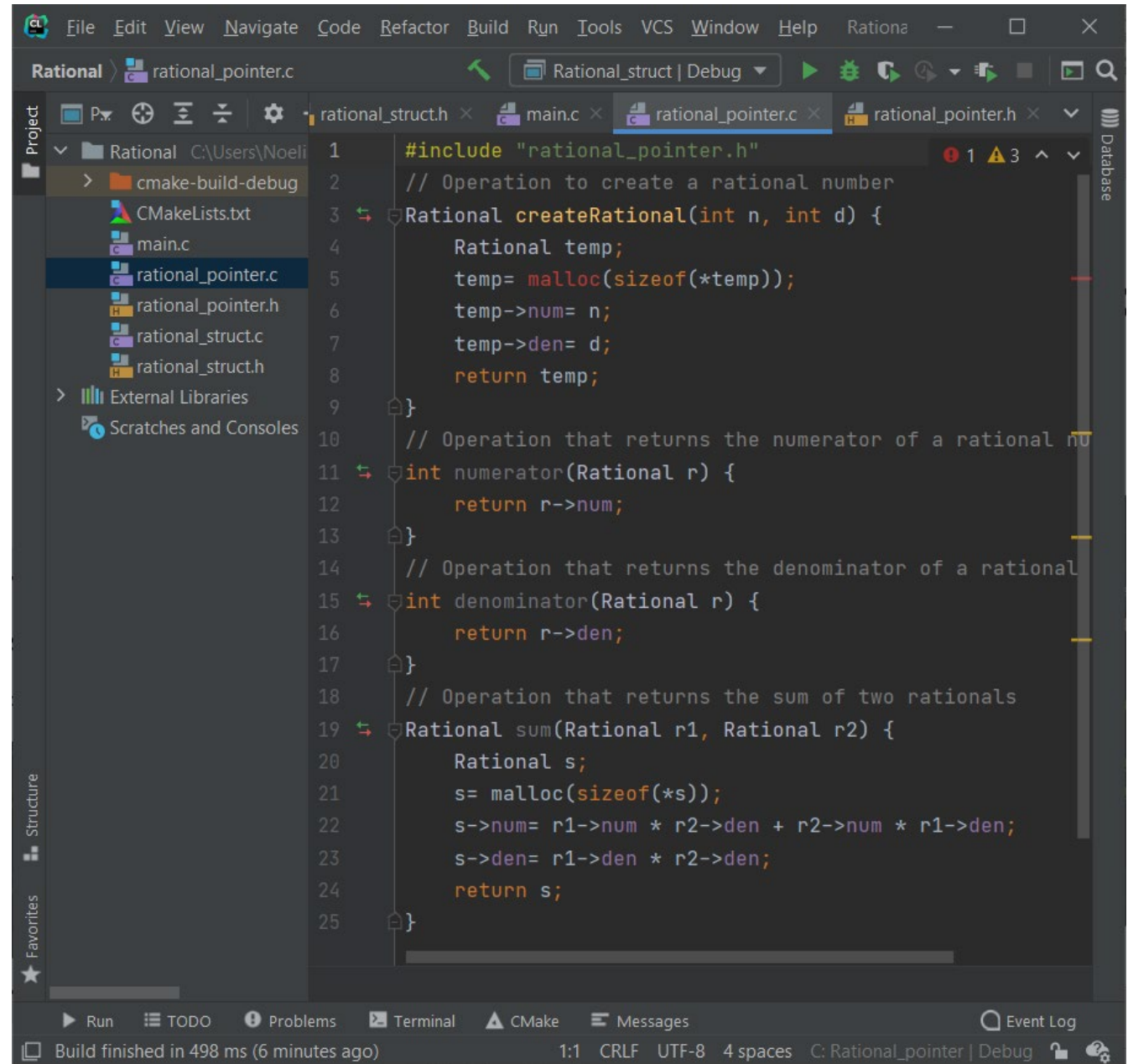
rational_pointer: Cambio en la declaración de tipos



The screenshot shows an IDE window with the file `rational_pointer.h` open. The code defines a `Rational` type as a pointer to a `Data` struct. The `typedef struct Data* Rational;` line is highlighted with a red box. Below it, the `struct Data` is defined with `int num;` and `int den;` members. Several functions are declared, including `createRational`, `numerator`, `denominator`, and `sum`. The file ends with `#endif //RATIONAL_RATIONAL_POINTER_H`. The IDE interface includes a menu bar, a toolbar, a project explorer on the left, and a status bar at the bottom.

```
4
5 typedef struct Data* Rational;
6 struct Data {
7     int num;
8     int den;
9 };
10 Rational createRational(int n, int d);
11 int numerator(Rational r);
12 int denominator(Rational r);
13 Rational sum(Rational r1, Rational r2);
14
15 #endif //RATIONAL_RATIONAL_POINTER_H
16
```

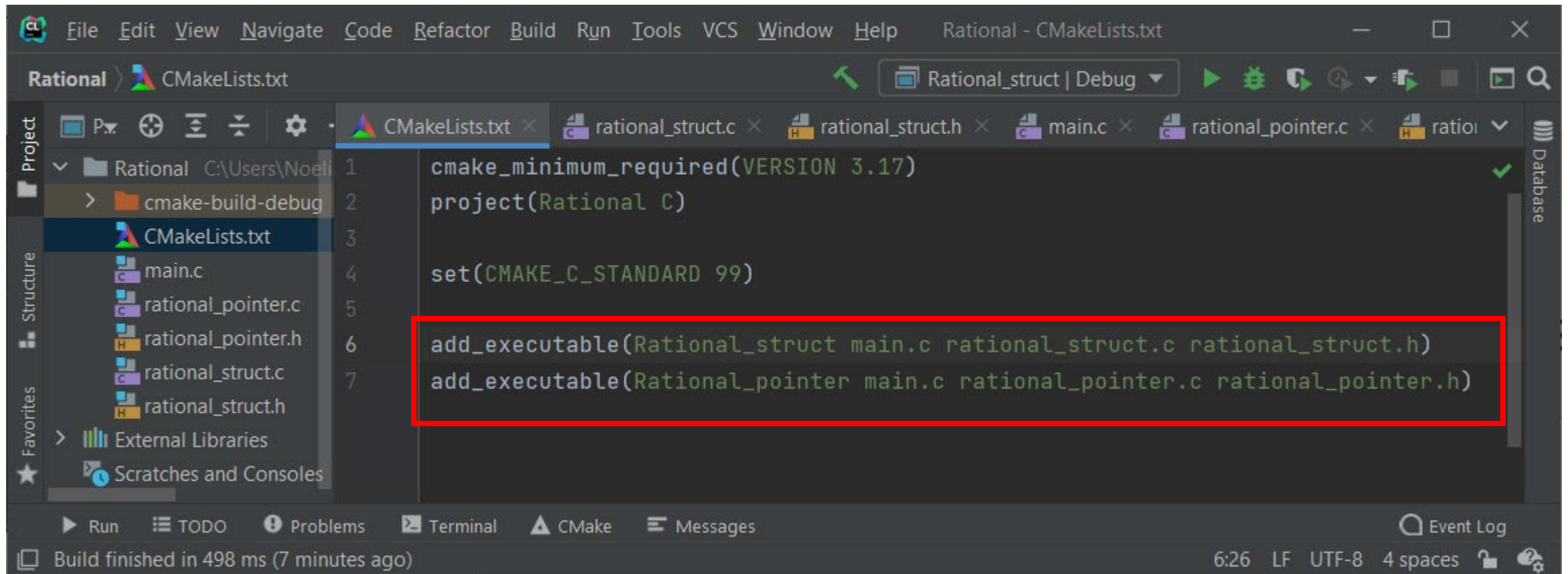
Rational_pointer: implementación



```
1  #include "rational_pointer.h"
2  // Operation to create a rational number
3  Rational createRational(int n, int d) {
4      Rational temp;
5      temp = malloc(sizeof(*temp));
6      temp->num = n;
7      temp->den = d;
8      return temp;
9  }
10 // Operation that returns the numerator of a rational number
11 int numerator(Rational r) {
12     return r->num;
13 }
14 // Operation that returns the denominator of a rational number
15 int denominator(Rational r) {
16     return r->den;
17 }
18 // Operation that returns the sum of two rationals
19 Rational sum(Rational r1, Rational r2) {
20     Rational s;
21     s = malloc(sizeof(*s));
22     s->num = r1->num * r2->den + r2->num * r1->den;
23     s->den = r1->den * r2->den;
24     return s;
25 }
```

Cambios automáticos en CMakeLists.txt

- Hay 2 posibilidades de ejecución: *rational_struct* o *rational_pointer*



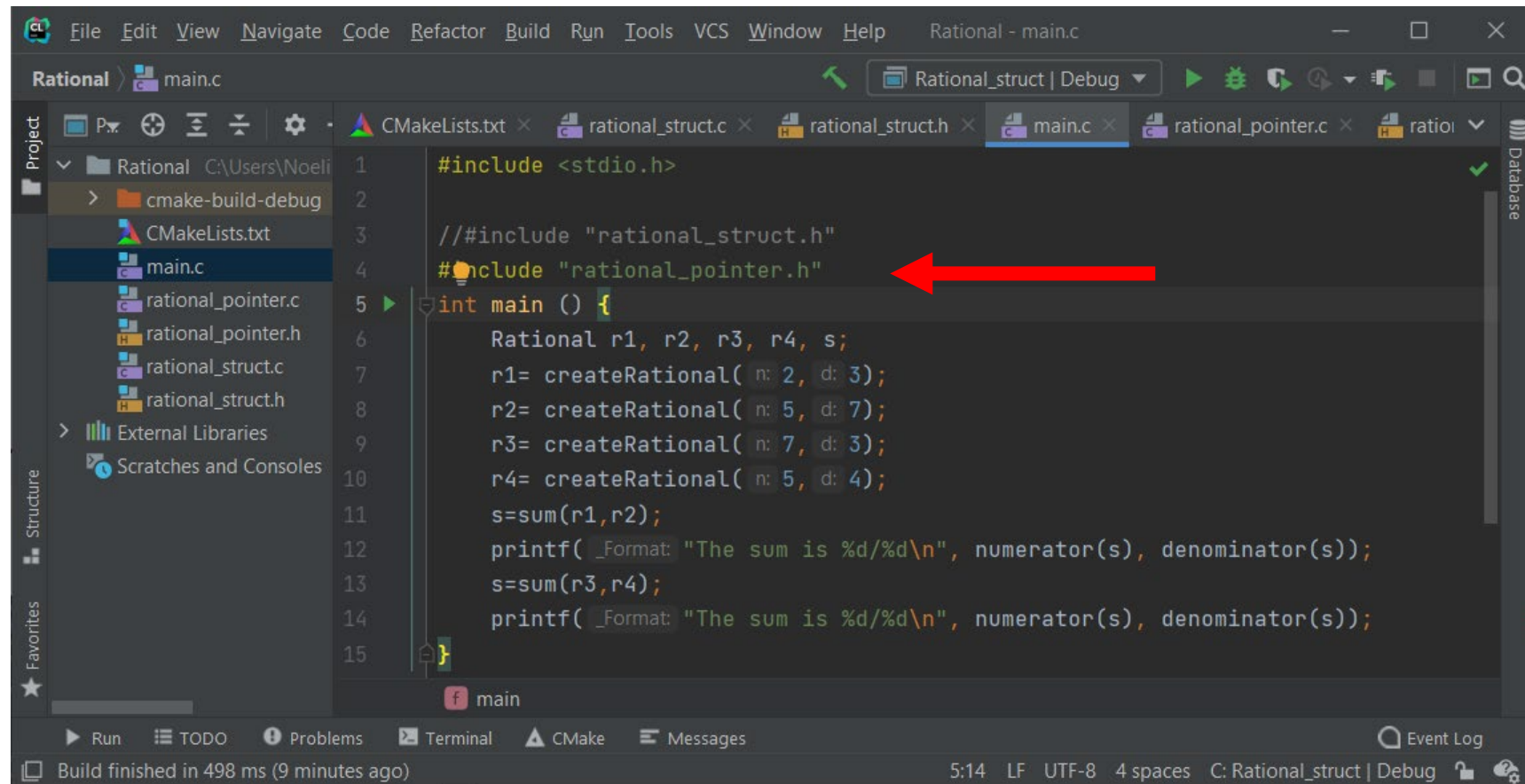
The screenshot shows an IDE window titled "Rational - CMakeLists.txt". The left sidebar displays the project structure with "CMakeLists.txt" selected. The main editor area shows the following CMake code:

```
1 cmake_minimum_required(VERSION 3.17)
2 project(Rational C)
3
4 set(CMAKE_C_STANDARD 99)
5
6 add_executable(Rational_struct main.c rational_struct.c rational_struct.h)
7 add_executable(Rational_pointer main.c rational_pointer.c rational_pointer.h)
```

The last two lines, which define the executables, are enclosed in a red rectangular box. The status bar at the bottom indicates "Build finished in 498 ms (7 minutes ago)".

Cambiamos la librería en el main.c

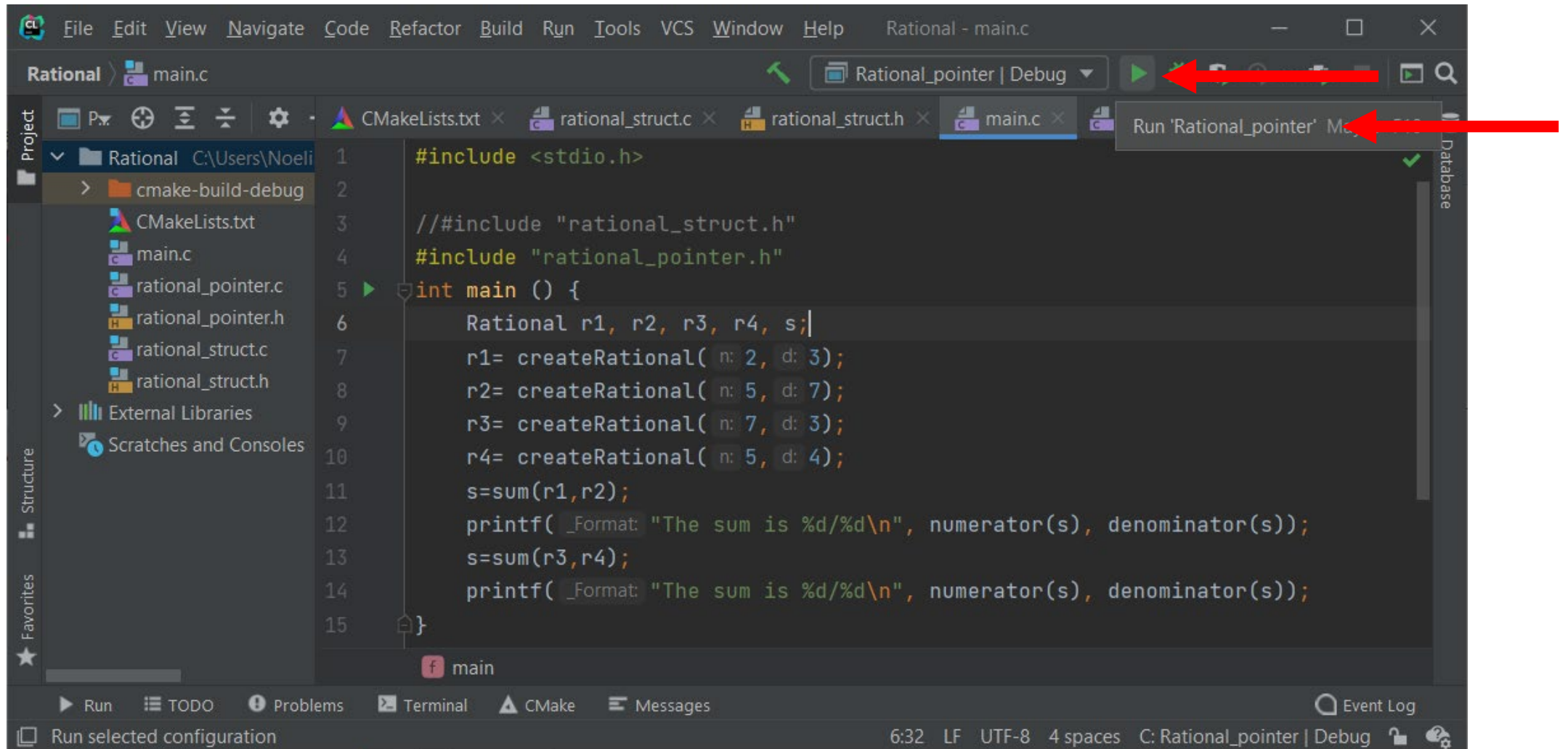
- ÚNICO cambio que hay que hacer en el main



```
1  #include <stdio.h>
2
3  // #include "rational_struct.h"
4  #include "rational_pointer.h"
5
6  int main () {
7      Rational r1, r2, r3, r4, s;
8      r1= createRational( n: 2, d: 3);
9      r2= createRational( n: 5, d: 7);
10     r3= createRational( n: 7, d: 3);
11     r4= createRational( n: 5, d: 4);
12     s=sum(r1,r2);
13     printf( _Format: "The sum is %d/%d\n", numerator(s), denominator(s));
14     s=sum(r3,r4);
15     printf( _Format: "The sum is %d/%d\n", numerator(s), denominator(s));
16 }
```

Build finished in 498 ms (9 minutes ago) 5:14 LF UTF-8 4 spaces C: Rational_struct | Debug

Ejecutamos el proyecto

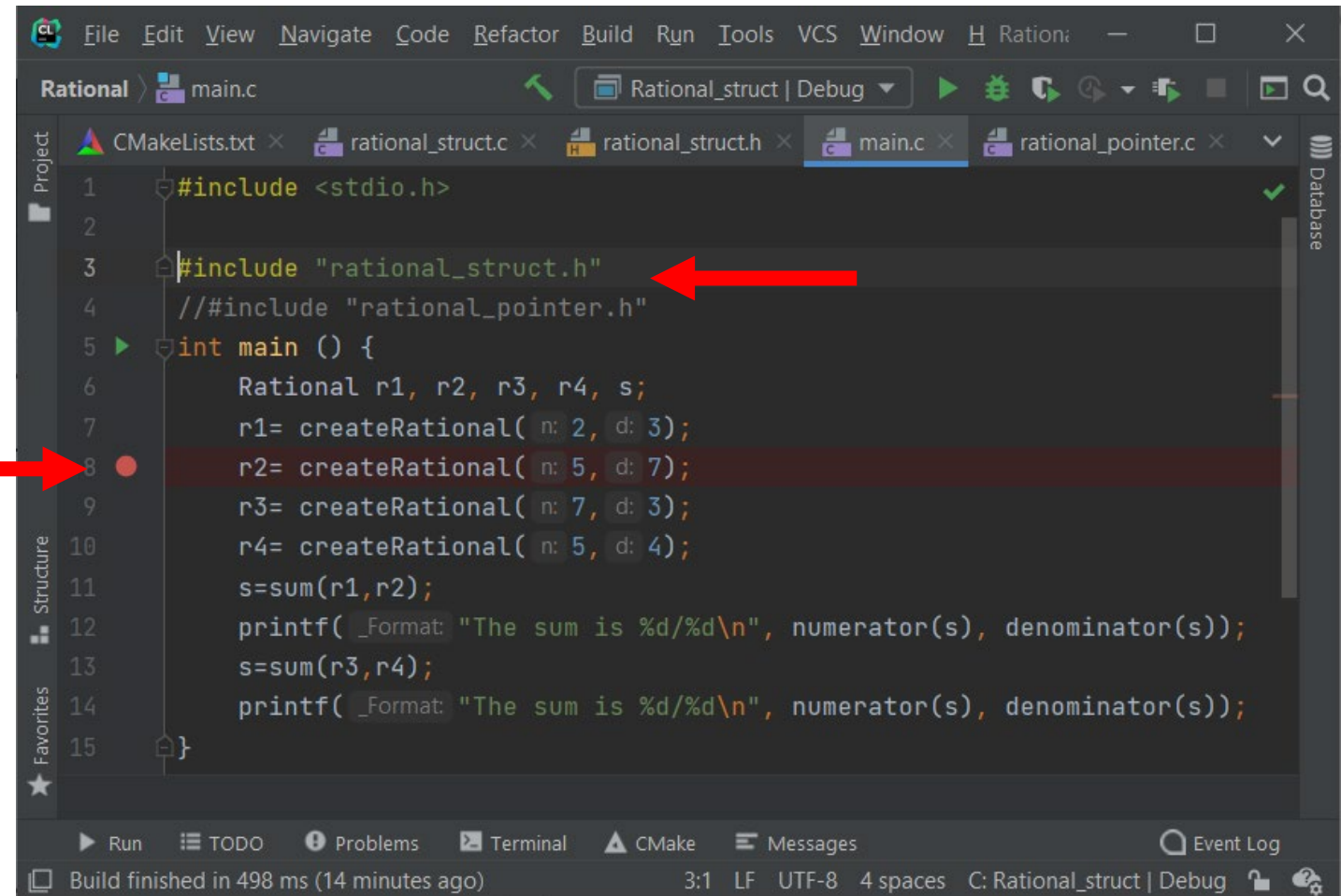


Depuración

- El depurador nos ayuda a detectar fallos de ejecución en el código
- Vamos a ilustrar algunas funcionalidades del depurador de CLion usando *rational_struct*

Depurando el código: puntos de ruptura

- Pulsando con el botón izquierdo al lado de los números de línea añadimos un punto de ruptura
- Usando el depurador, el programa se ejecutará con normalidad hasta ese punto



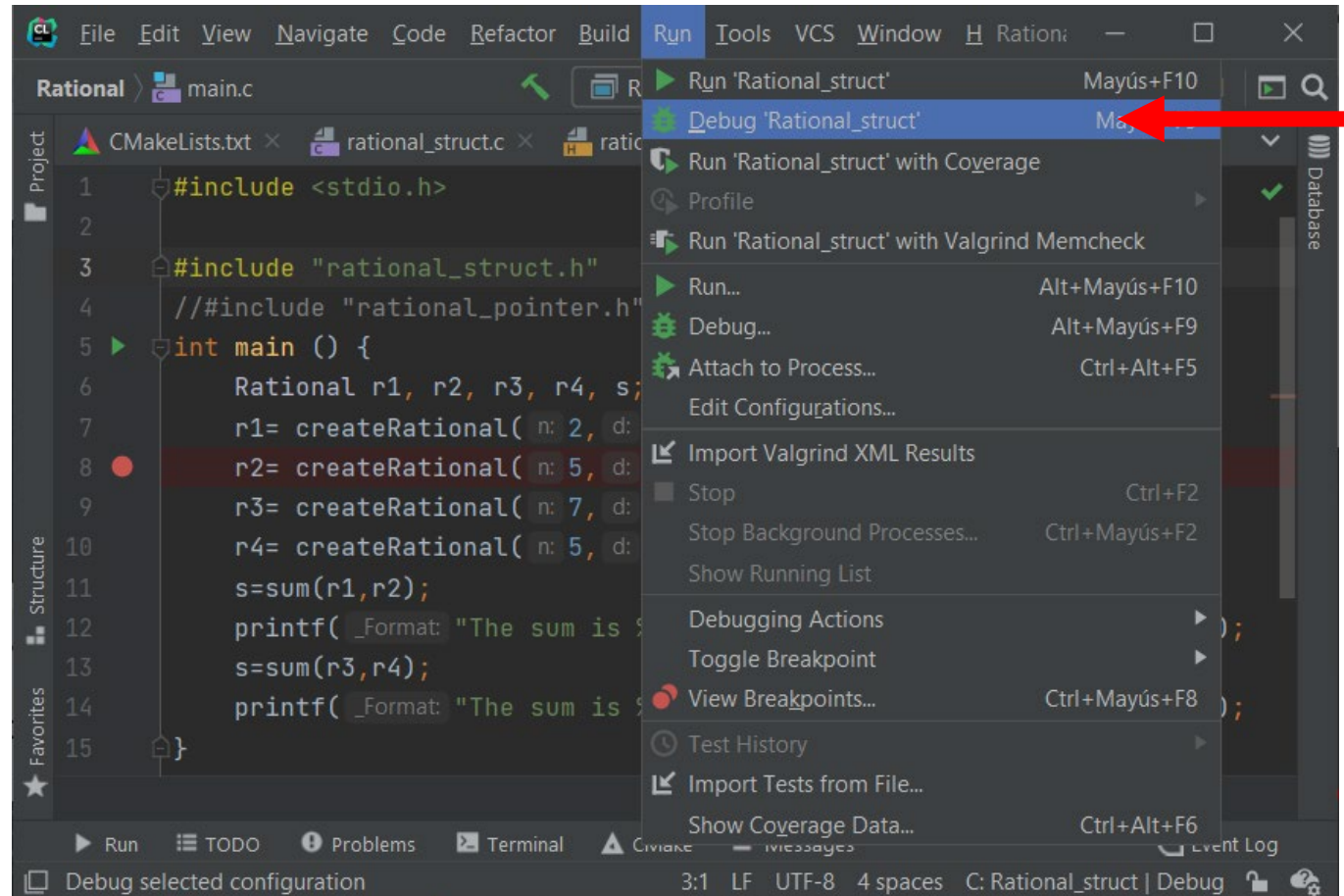
The screenshot shows an IDE window titled 'Rational' with a file named 'main.c' open. The code is as follows:

```
1 #include <stdio.h>
2
3 #include "rational_struct.h"
4 //include "rational_pointer.h"
5 int main () {
6     Rational r1, r2, r3, r4, s;
7     r1= createRational( n: 2, d: 3);
8     r2= createRational( n: 5, d: 7);
9     r3= createRational( n: 7, d: 3);
10    r4= createRational( n: 5, d: 4);
11    s=sum(r1,r2);
12    printf( _Format: "The sum is %d/%d\n", numerator(s), denominator(s));
13    s=sum(r3,r4);
14    printf( _Format: "The sum is %d/%d\n", numerator(s), denominator(s));
15 }
```

A red arrow points to the left margin next to line 8, where a red dot (breakpoint) has been set. Another red arrow points to the line number '3' in the code, indicating where a breakpoint can also be set. The IDE interface includes a menu bar, a toolbar with icons for running and debugging, and a status bar at the bottom showing 'Build finished in 498 ms (14 minutes ago)' and '3:1 LF UTF-8 4 spaces C: Rational_struct | Debug'.

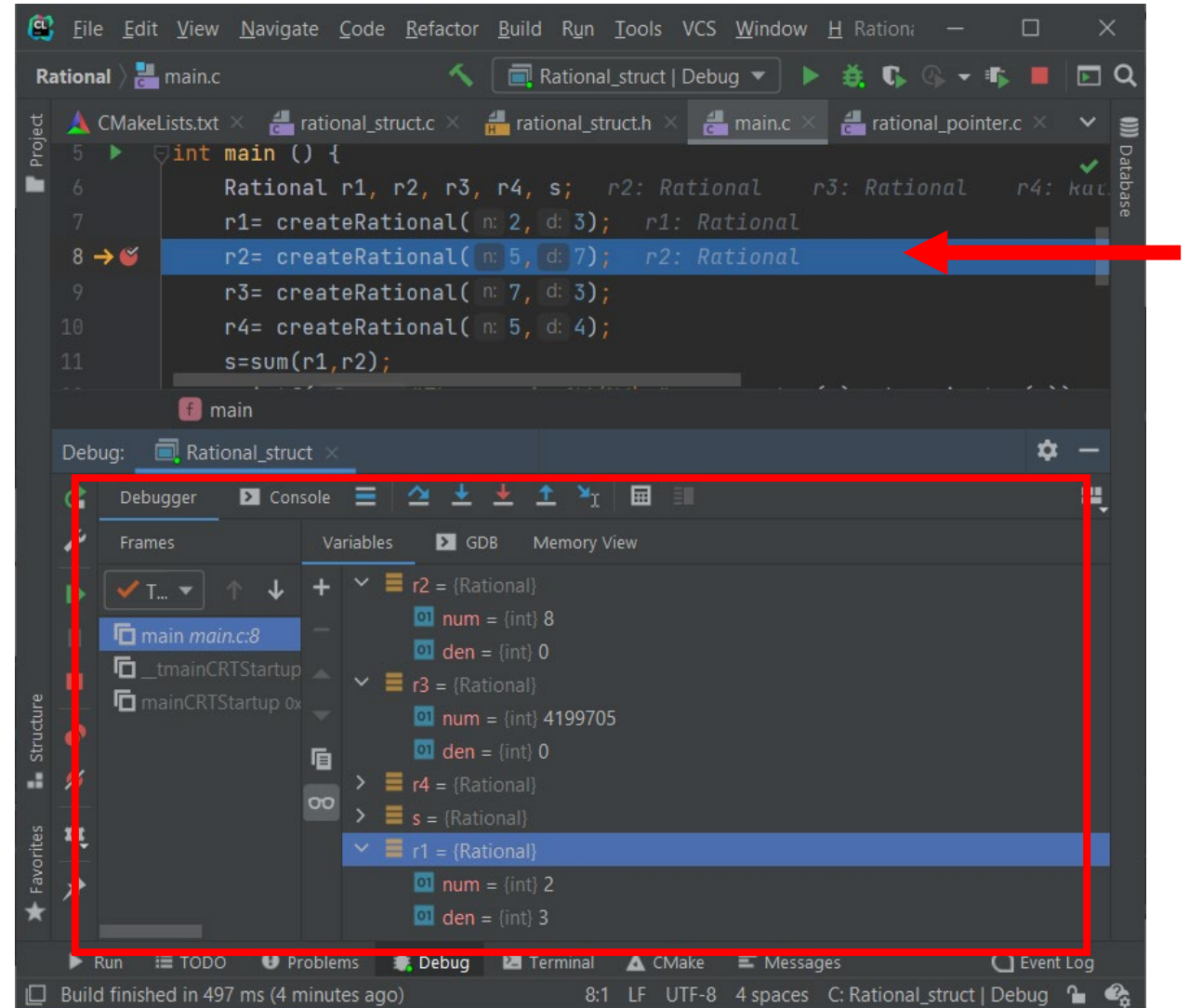
Depurando el código

- Seleccionamos la opción de depuración



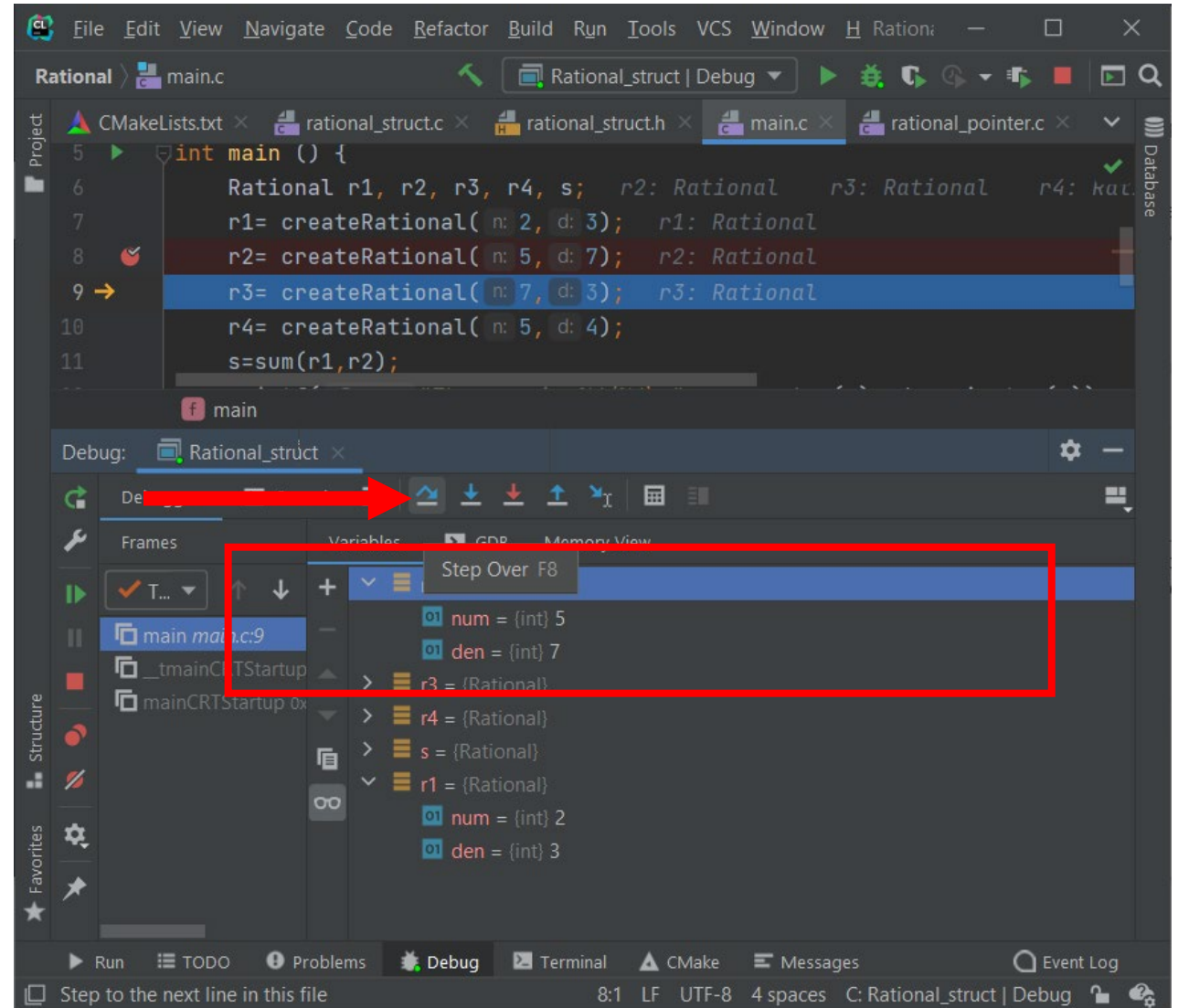
Depurando el código

- La ejecución se para en el punto de ruptura
- Podemos comprobar el estado de las variables
 - Solo `r1` inicializado



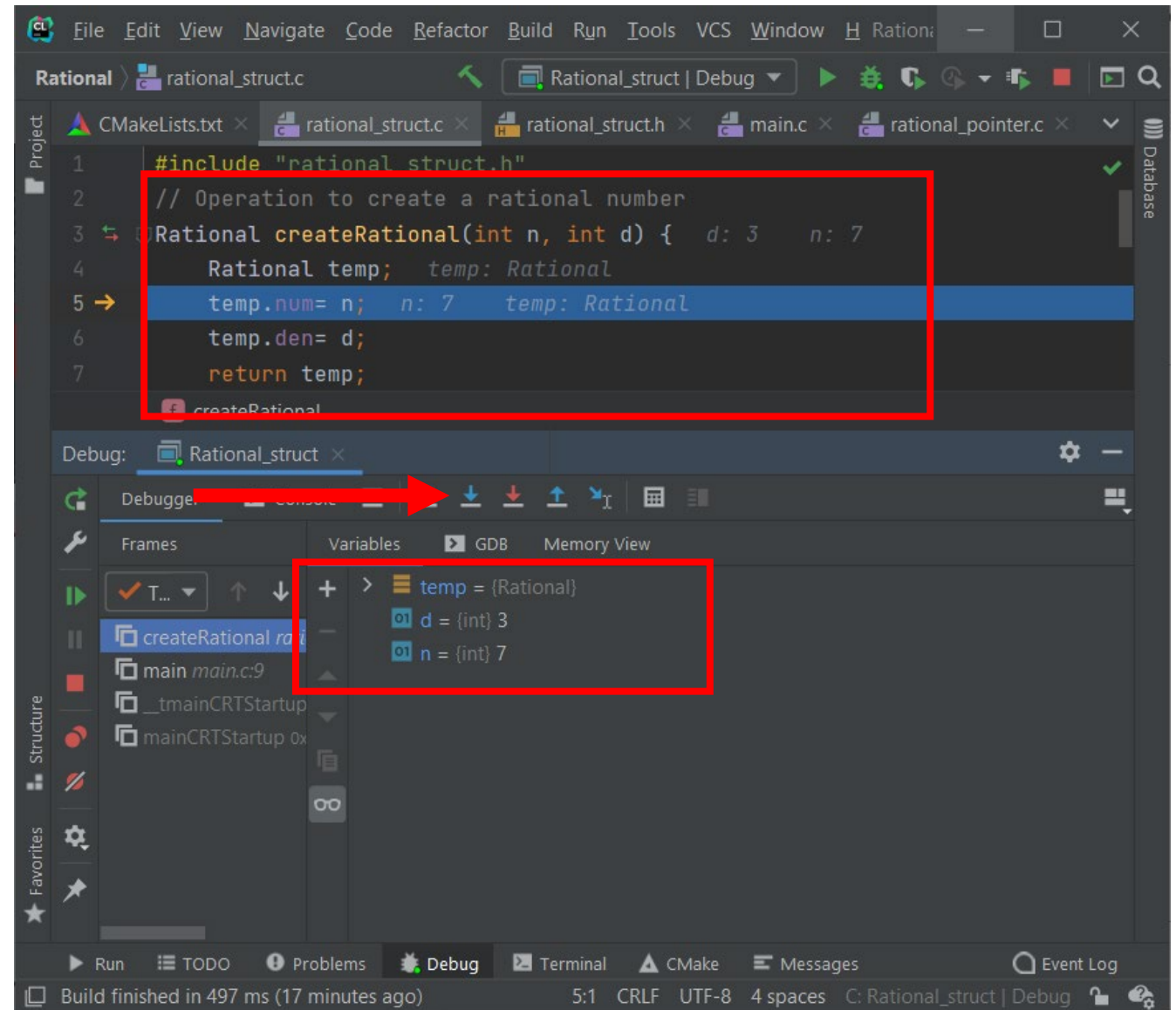
Depurando el código

- Pulsando *Step Over* (F8) se ejecuta la línea de código (sin entrar en la función)
- La variable `r2` ya está inicializada



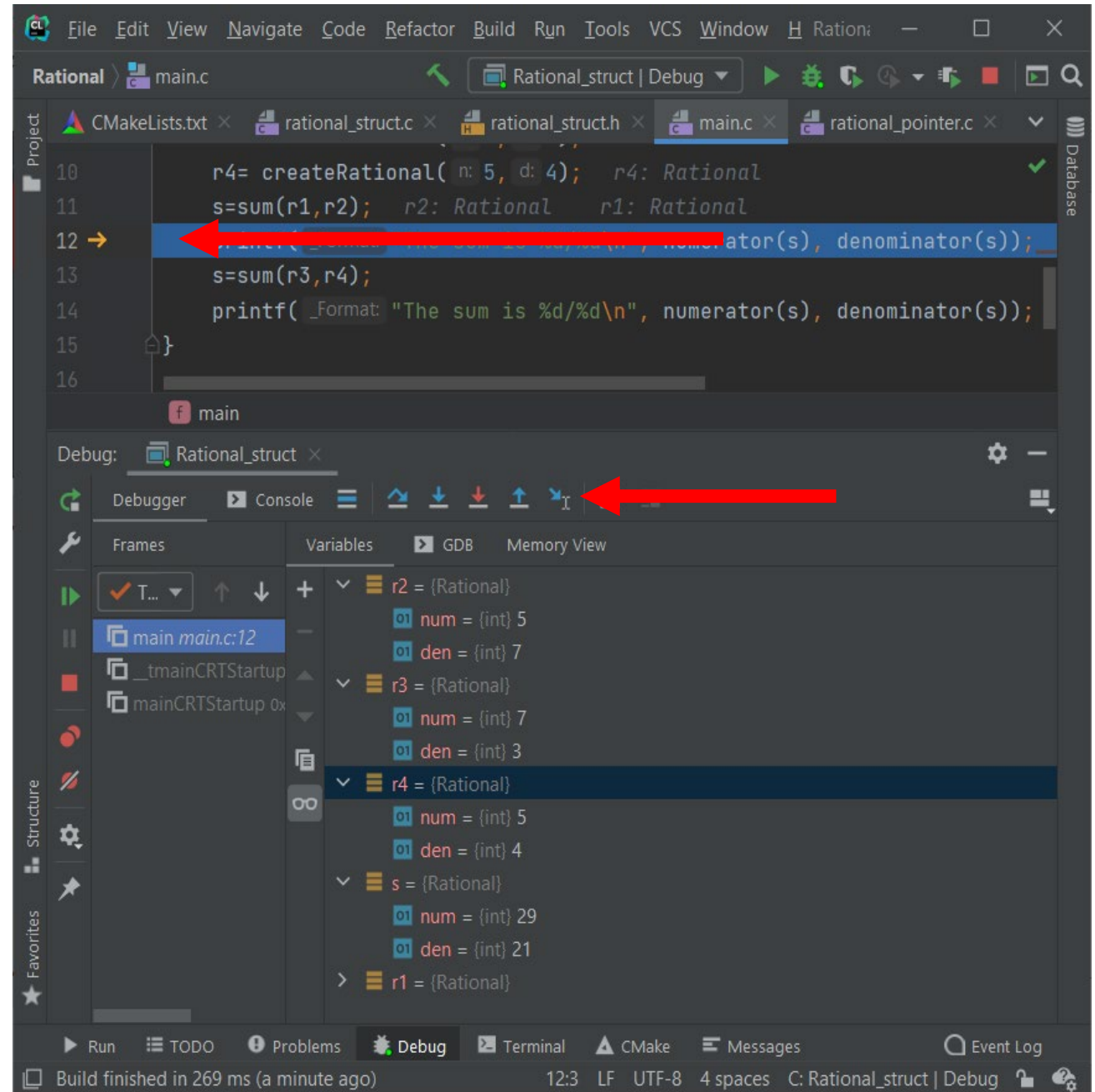
Depurando el código

- Pulsando *Step into* (F7) ejecutamos la siguiente línea entrando en la función
- El depurador entró en *createRational* y vemos las variables asociadas



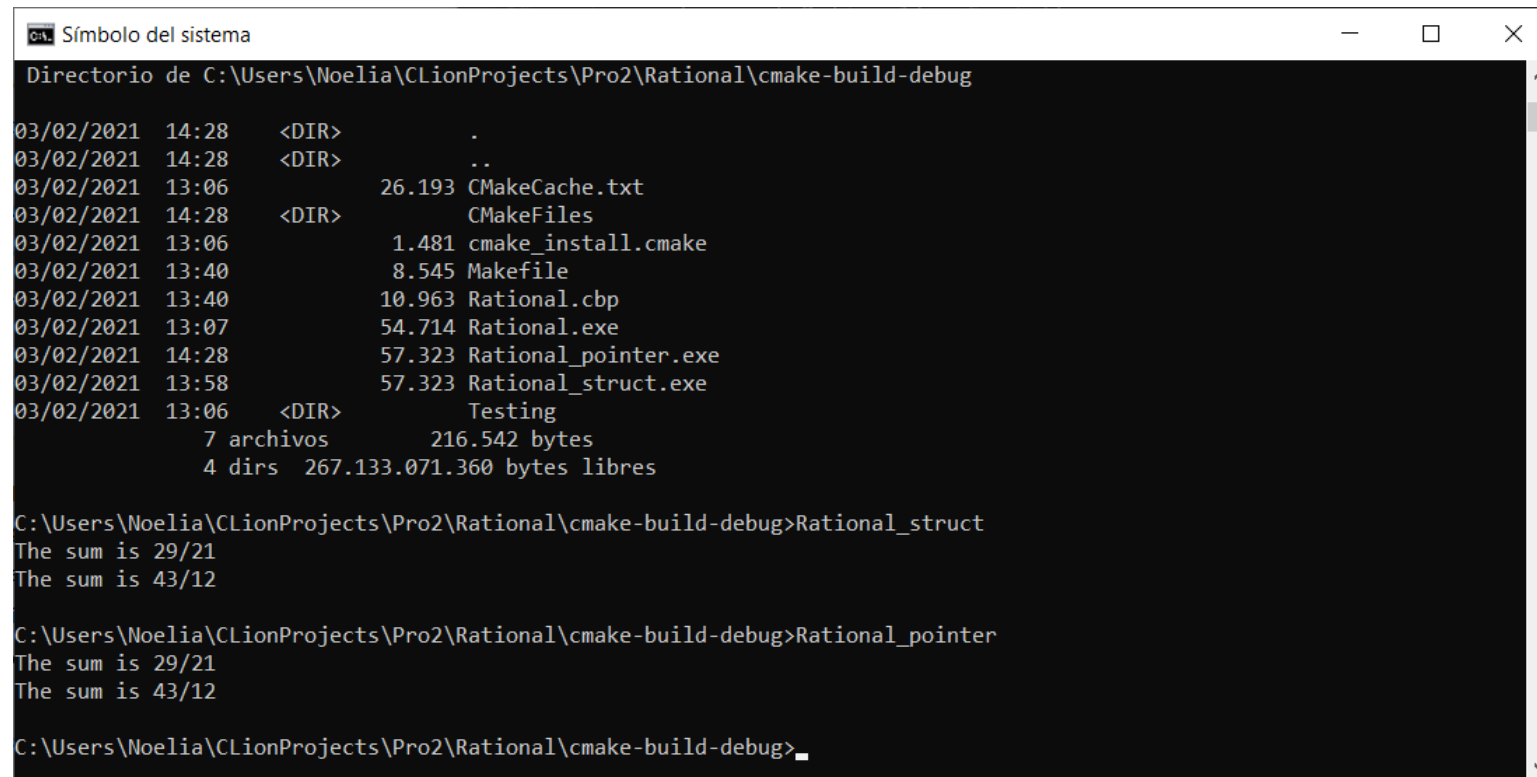
Depurando el código

- Posiciona el cursor en la línea a la que deseas ir
- Pulsa *Run to Cursor* (*Alt+F9*)
- El depurador ejecutó todas las líneas hasta el cursor (*s* tiene el primer valor)



Ejecución en consola

- También es posible ejecutar el programa en consola a partir de los ficheros binarios compilados



```
Símbolo del sistema
Directorio de C:\Users\Noelia\CLionProjects\Pro2\Rational\cmake-build-debug
03/02/2021 14:28 <DIR> .
03/02/2021 14:28 <DIR> ..
03/02/2021 13:06      26.193 CMakeCache.txt
03/02/2021 14:28 <DIR> CMakeFiles
03/02/2021 13:06      1.481 cmake_install.cmake
03/02/2021 13:40      8.545 Makefile
03/02/2021 13:40     10.963 Rational.cbp
03/02/2021 13:07     54.714 Rational.exe
03/02/2021 14:28     57.323 Rational_pointer.exe
03/02/2021 13:58     57.323 Rational_struct.exe
03/02/2021 13:06 <DIR> Testing
                7 archivos      216.542 bytes
                4 dirs 267.133.071.360 bytes libres

C:\Users\Noelia\CLionProjects\Pro2\Rational\cmake-build-debug>Rational_struct
The sum is 29/21
The sum is 43/12

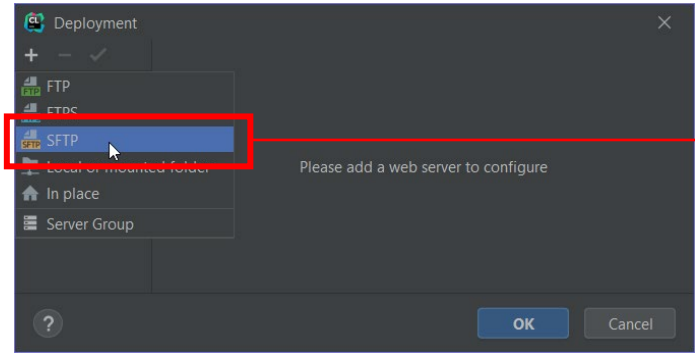
C:\Users\Noelia\CLionProjects\Pro2\Rational\cmake-build-debug>Rational_pointer
The sum is 29/21
The sum is 43/12

C:\Users\Noelia\CLionProjects\Pro2\Rational\cmake-build-debug>
```

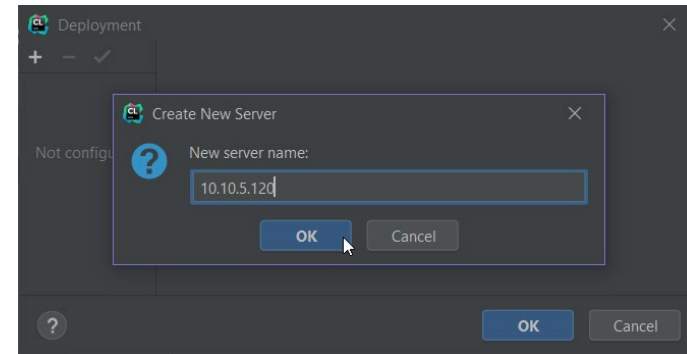
Configuración de CLion para sincronización en la máquina de referencia

- La práctica deberá funcionar correctamente en la máquina de referencia (**10.11.28.50**). Para poder acceder a dicha máquina desde el exterior de la red de docencia es necesario configurar la VPN en nuestro equipo: [Recursos informáticos \(sharepoint.com\)](https://sharepoint.com)
- Es posible configurar el CLion para habilitar la opción de sincronizar el código en la máquina de referencia y facilitar la prueba posterior del mismo en ésta.
- Para ello es necesario configurar la opción de Deployment (**Tools -> Deployment -> Configuration**)
 - Configurar el apartado **Connection**.
 - Configurar el directorio de sincronización en **Mappings**.
- Una vez configurado, la opción **Tools->Deployment->Upload to** subirá los ficheros a la máquina de referencia.
- *En las siguientes transparencias se ilustra este proceso.*

Configuración de CLion para sincronización en la máquina de referencia

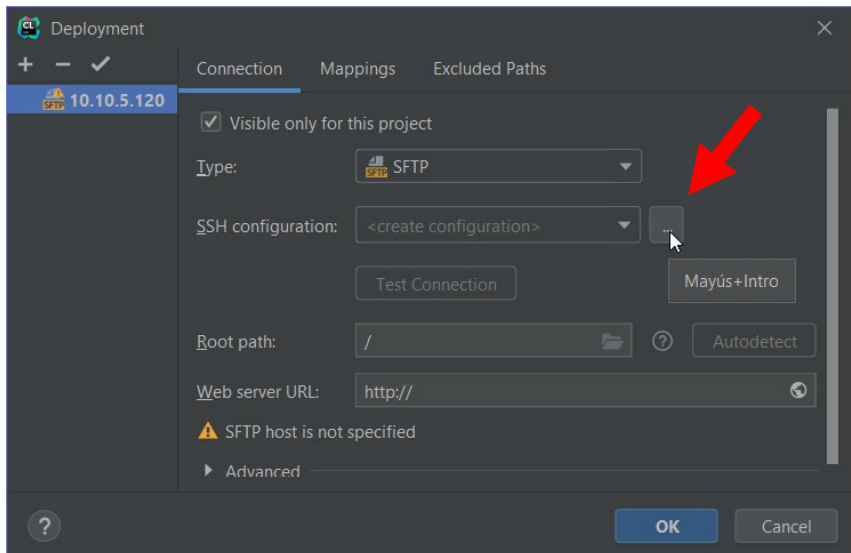


1

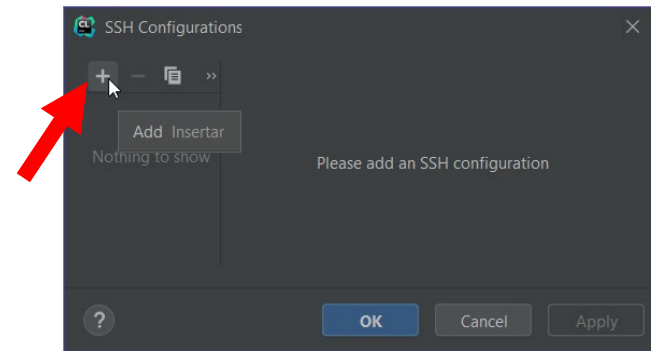


2

IP: 10.11.28.50



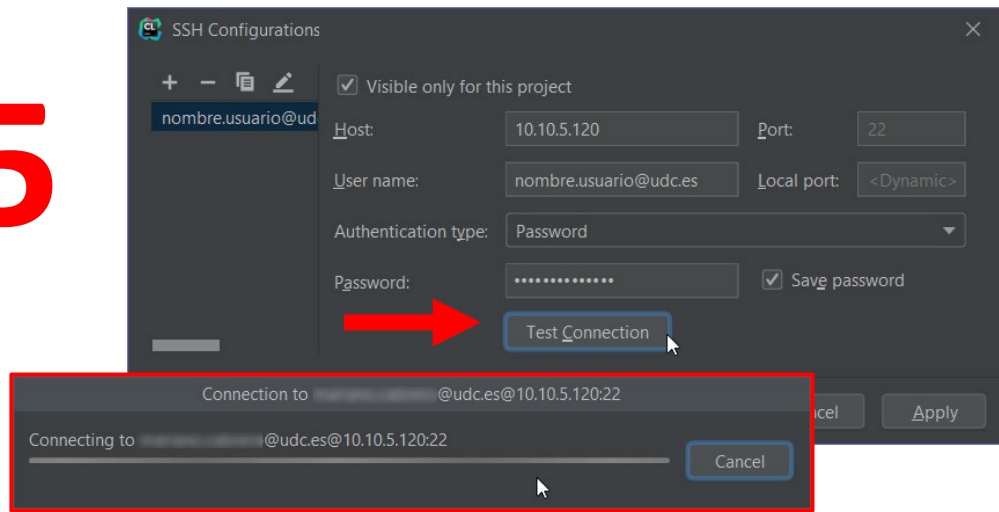
3



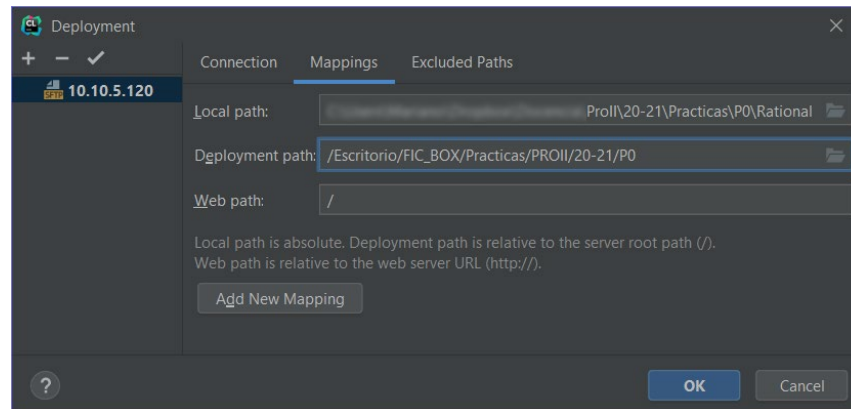
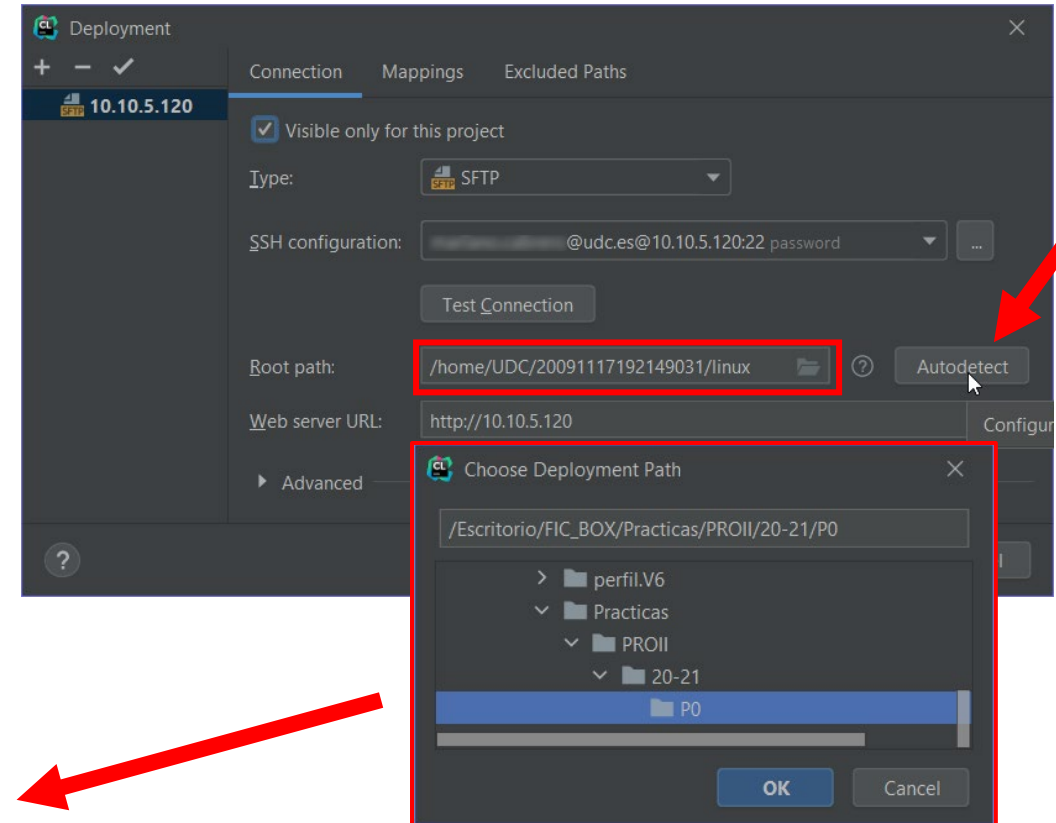
4

Configuración de CLion para sincronización en la máquina de referencia

5



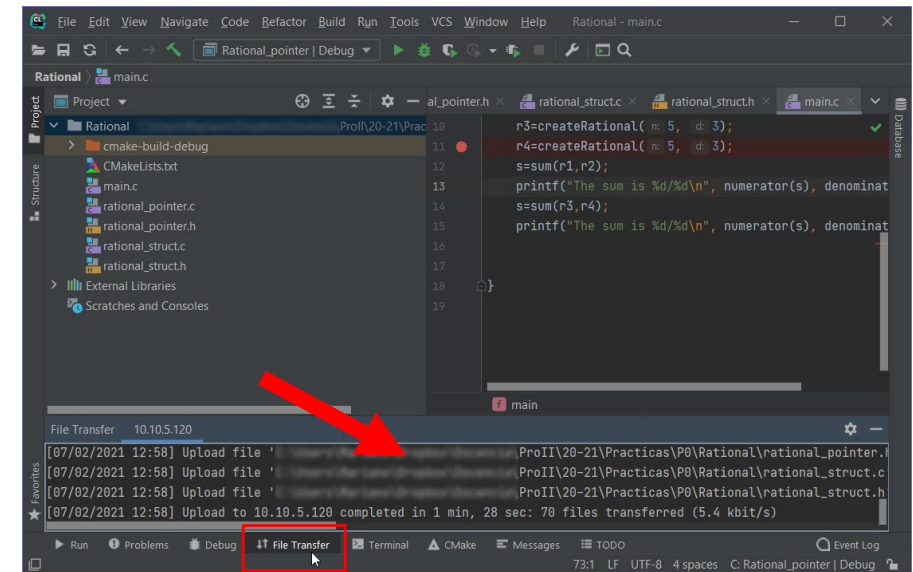
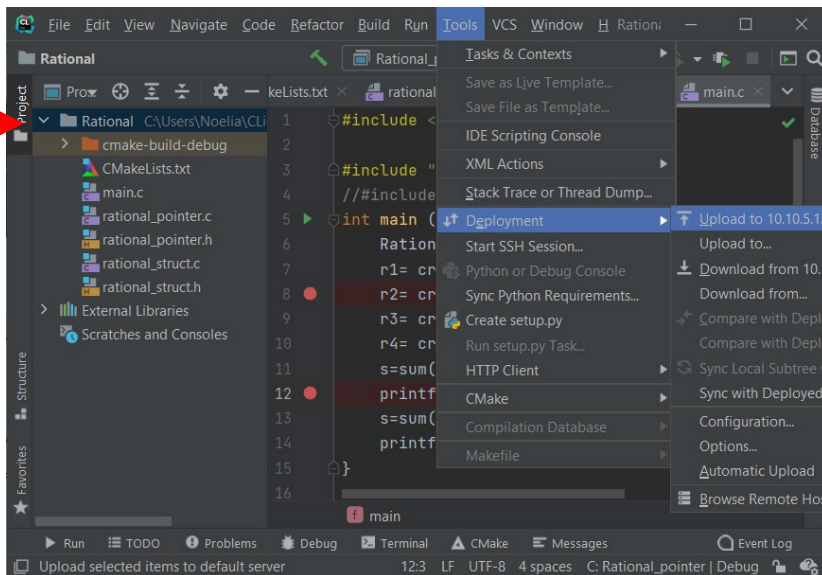
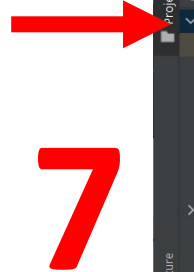
6



Configuración de CLion para sincronización en la máquina de referencia

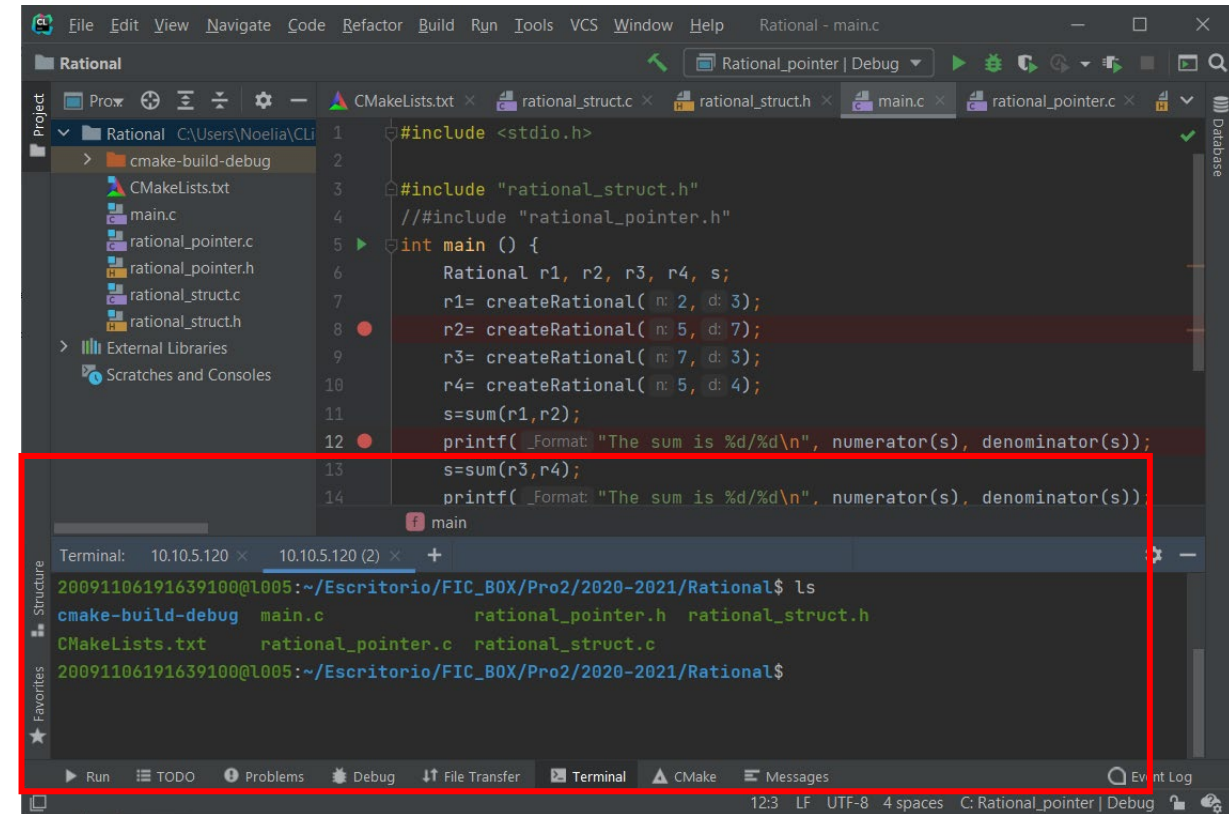
- Cada vez que se ejecute la opción del menú **Tools -> Deployment -> Upload to...** se sincronizarán los cambios.
 - En el árbol de archivos de CLion debe seleccionarse la carpeta del proyecto para que se suban **todos** los ficheros.
- Alternativamente, es posible utilizar un programa como **Filezilla** para transmitir los ficheros de la práctica a la máquina de referencia.

⚠
Seleccionar
carpeta del
proyecto



Sesión ssh remota con la máquina de referencia

- Es posible abrir una conexión **ssh** con la máquina de referencia desde el propio CLion, utilizando la opción del menú:
Tools -> Start SSH Session...
- Esta opción no está disponible en algunas versiones de CLion.
- Alternativamente, es posible utilizar cualquier otra aplicación disponible para abrir una terminal remota en la máquina de referencia, por ejemplo, la aplicación PuTTY.



The screenshot displays the CLion IDE interface. The top toolbar includes menus like File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, and Help. The project 'Rational' is open, with files like CMakeLists.txt, main.c, rational_pointer.c, rational_pointer.h, rational_struct.c, and rational_struct.h visible in the Project tool window. The main editor shows the code for main.c, which includes headers and defines a main function that creates rational numbers and calculates their sum. Below the code editor, a terminal window is open, showing an SSH session to a remote machine (10.10.5.120). The terminal output shows the user running 'ls' and listing the files in the project directory. The terminal window is highlighted with a red border.

```
#include <stdio.h>
#include "rational_struct.h"
// #include "rational_pointer.h"
int main () {
    Rational r1, r2, r3, r4, s;
    r1= createRational( n: 2, d: 3);
    r2= createRational( n: 5, d: 7);
    r3= createRational( n: 7, d: 3);
    r4= createRational( n: 5, d: 4);
    s=sum(r1,r2);
    printf( _Format: "The sum is %d/%d\n", numerator(s), denominator(s));
    s=sum(r3,r4);
    printf( _Format: "The sum is %d/%d\n", numerator(s), denominator(s));
}
```

```
Terminal: 10.10.5.120 x 10.10.5.120 (2) x +
20091106191639100@l005:~/Escritorio/FIC_BOX/Pro2/2020-2021/Rational$ ls
cmake-build-debug  main.c          rational_pointer.h  rational_struct.h
CMakeLists.txt    rational_pointer.c  rational_struct.c
20091106191639100@l005:~/Escritorio/FIC_BOX/Pro2/2020-2021/Rational$
```