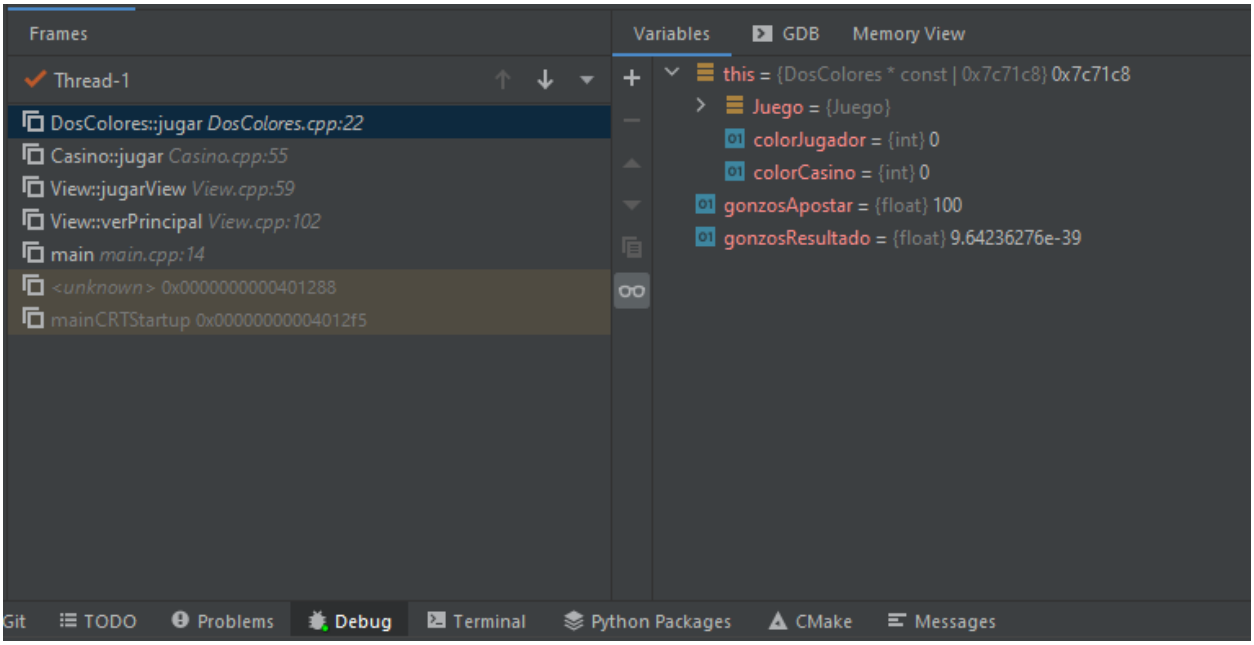
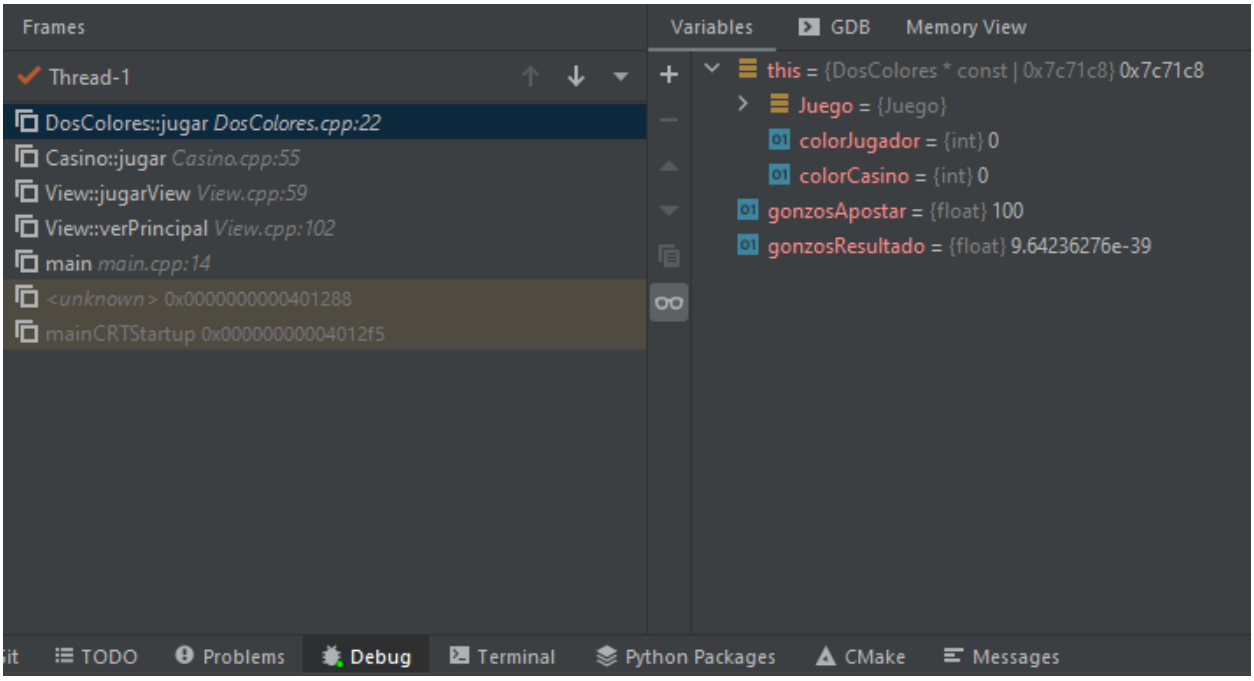
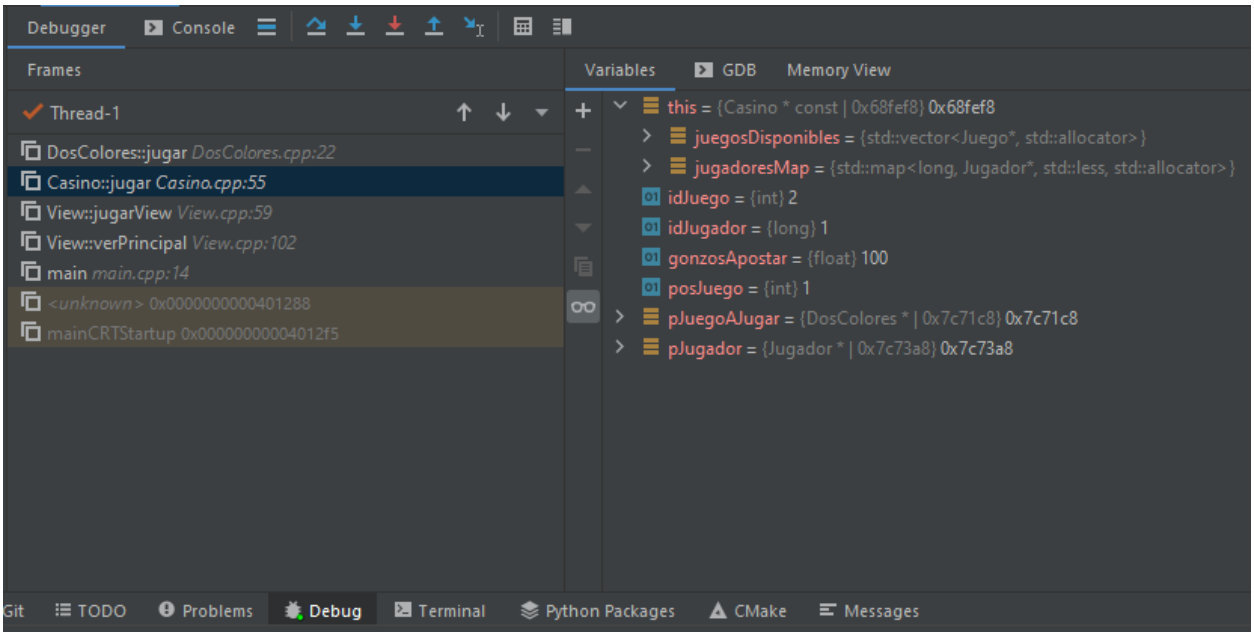
1:

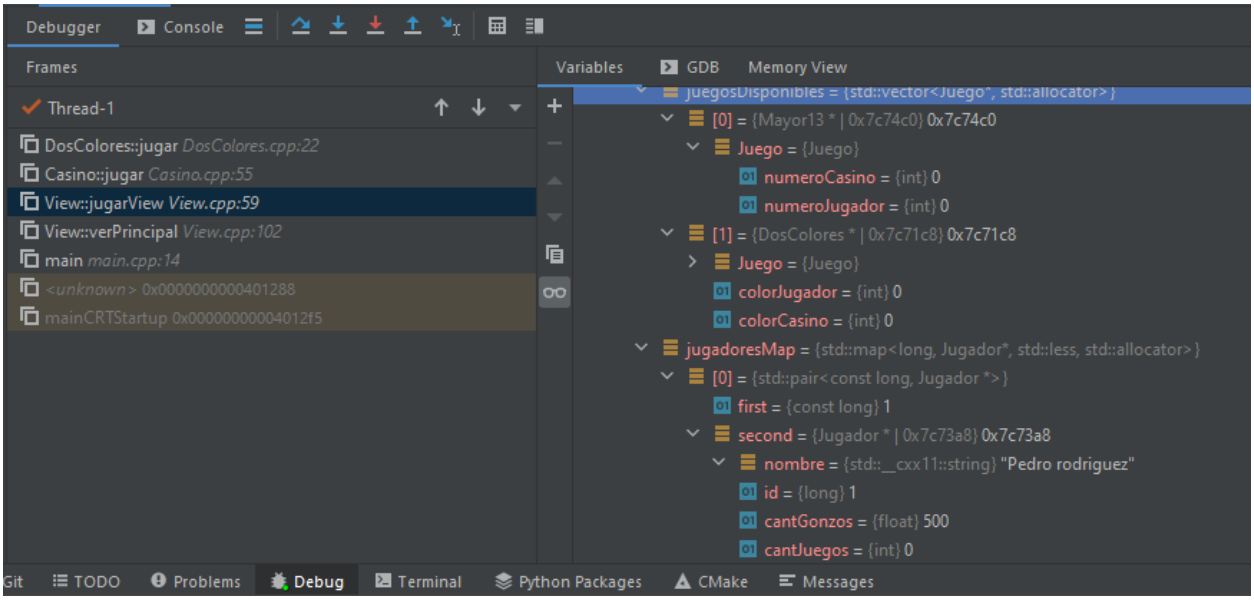


A: El método que inició la ejecución es el mainCRTStartup.

B:

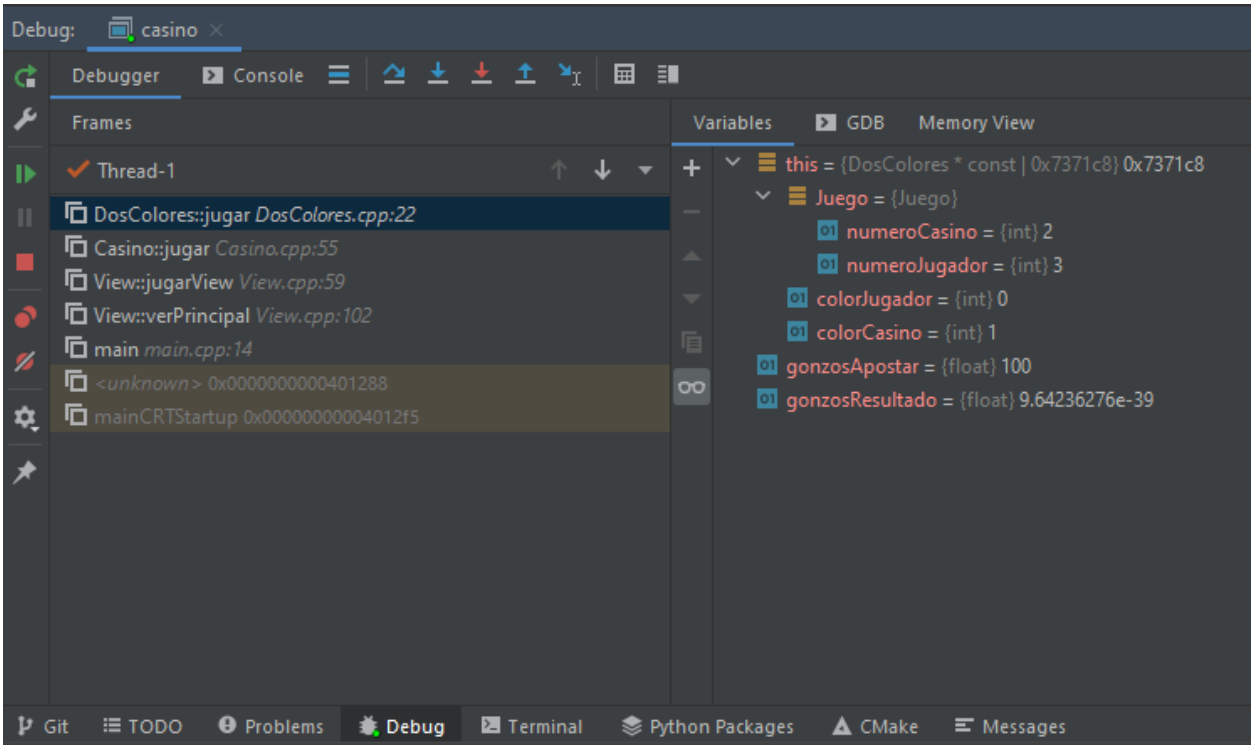



Debugger
Console
Frames
Thread-1
DosColores::jugar DosColores.cpp:22
Casino::jugar Casino.cpp:55
View::jugarView View.cpp:59
View::verPrincipal View.cpp:102
main main.cpp:14
<unknown> 0x0000000000401288
mainCRTStartup 0x00000000004012f5
Variables
GDB
Memory View
this = {Casino * const | 0x68fef8} 0x68fef8
juegosDisponibles = {std::vector<Juego*, std::allocator>}
jugadoresMap = {std::map<long, Jugador*, std::less, std::allocator>}
idJuego = {int} 2
idJugador = {long} 1
gonzosApostar = {float} 100
posJuego = {int} 1
pJuegoAJugar = {DosColores * | 0x7c71c8} 0x7c71c8
pJugador = {Jugador * | 0x7c73a8} 0x7c73a8
Git
TODO
Problems
Debug
Terminal
Python Packages
CMake
Messages

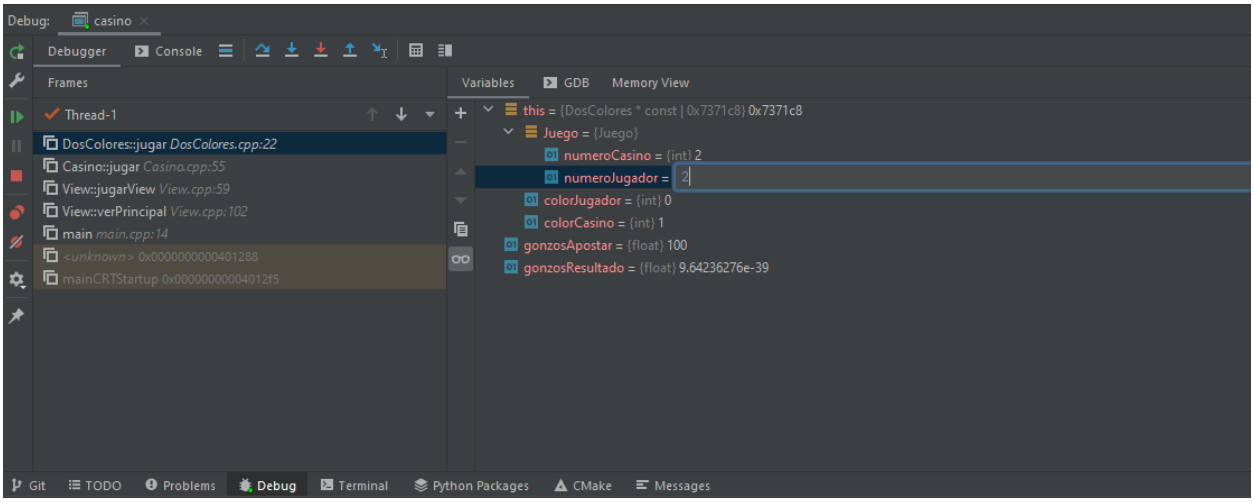

Debugger
Console
Frames
Thread-1
DosColores::jugar DosColores.cpp:22
Casino::jugar Casino.cpp:55
View::jugarView View.cpp:59
View::verPrincipal View.cpp:102
main main.cpp:14
<unknown> 0x0000000000401288
mainCRTStartup 0x00000000004012f5
Variables
GDB
Memory View
juegosDisponibles = {std::vector<Juego*, std::allocator>}
[0] = {Mayor13 * | 0x7c74c0} 0x7c74c0
Juego = {Juego}
numeroCasino = {int} 0
numeroJugador = {int} 0
[1] = {DosColores * | 0x7c71c8} 0x7c71c8
Juego = {Juego}
colorJugador = {int} 0
colorCasino = {int} 0
jugadoresMap = {std::map<long, Jugador*, std::less, std::allocator>}
[0] = {std::pair<const long, Jugador *>}
first = {const long} 1
second = {Jugador * | 0x7c73a8} 0x7c73a8
nombre = {std::__cxx11::string} "Pedro rodriguez"
id = {long} 1
cantGonzos = {float} 500
cantJuegos = {int} 0
Git
TODO
Problems
Debug
Terminal
Python Packages
CMake
Messages

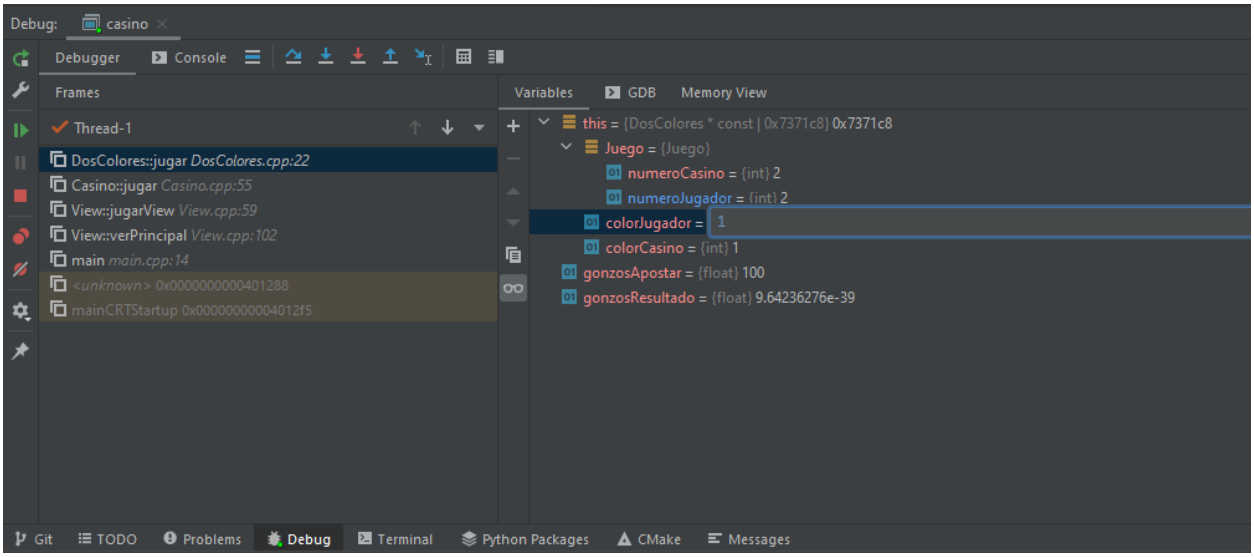
2:

En este caso el jugador pierde:

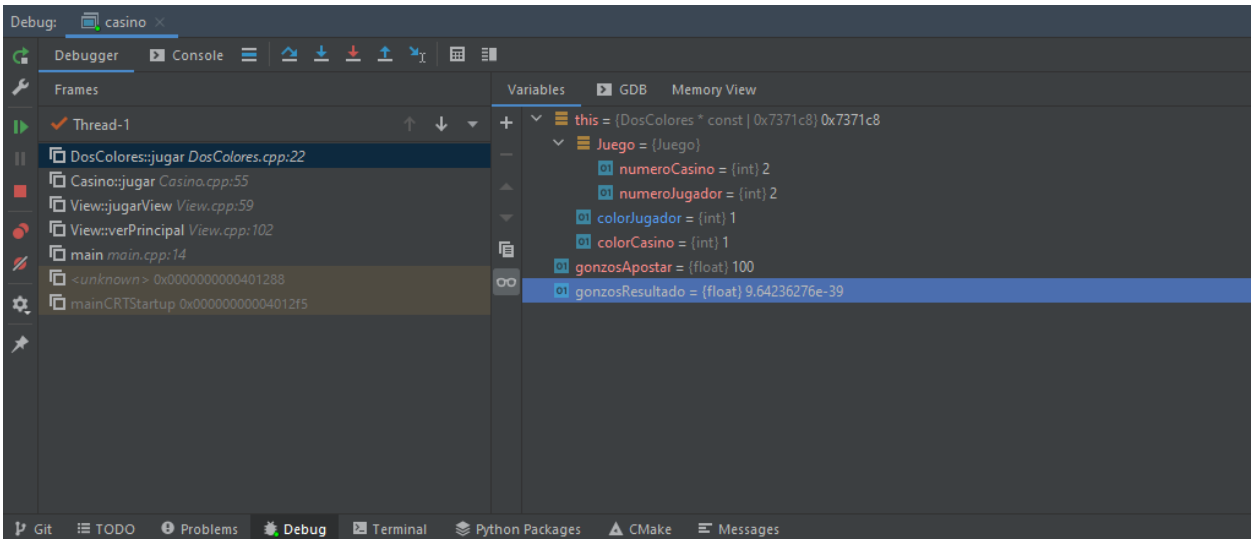


Modificamos las variables para que el jugador gane:



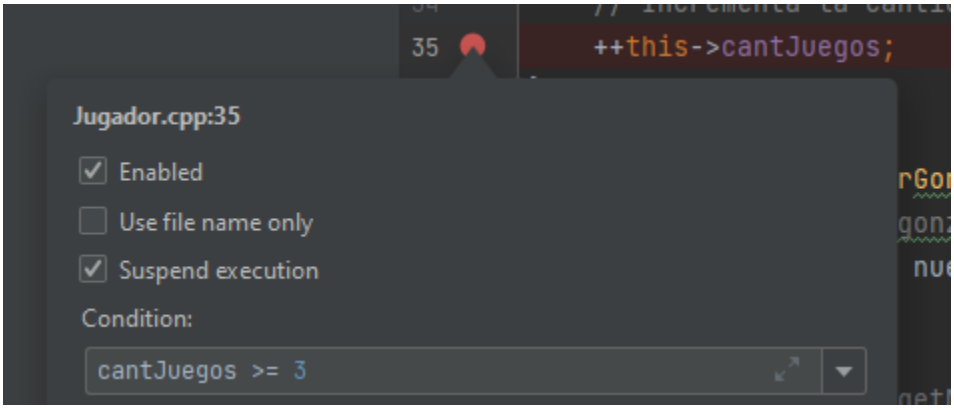


Habiendo hecho esto, los datos quedarían así:

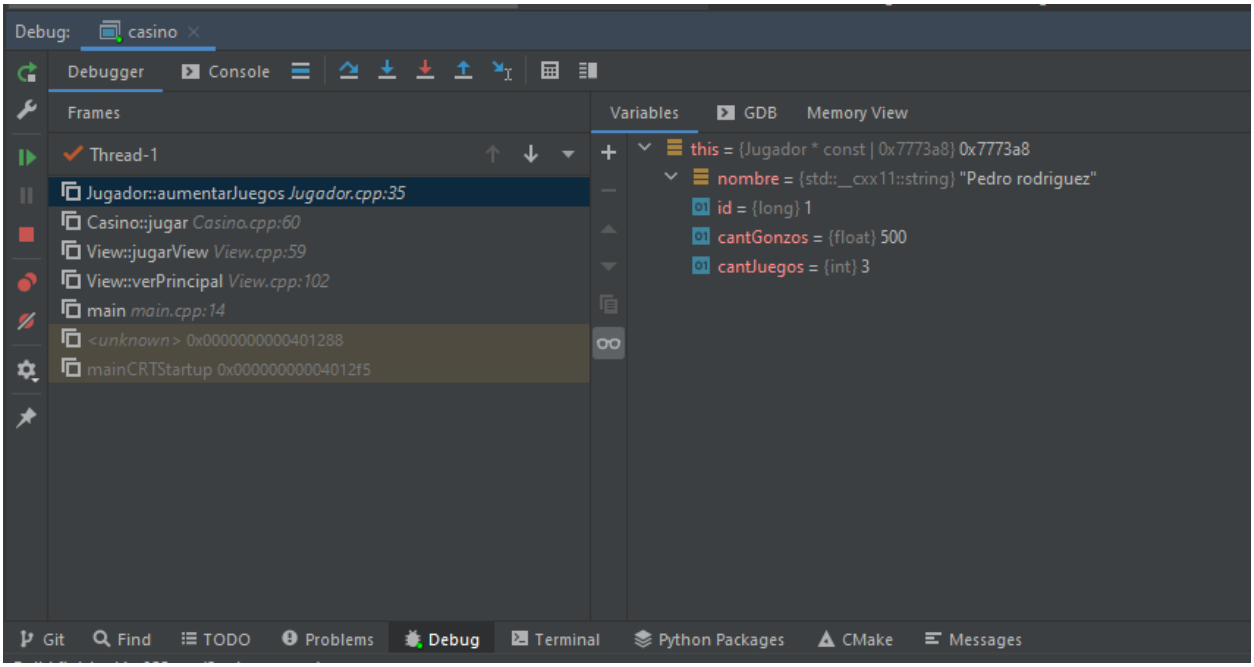


En este caso el jugador ganaría.

3:

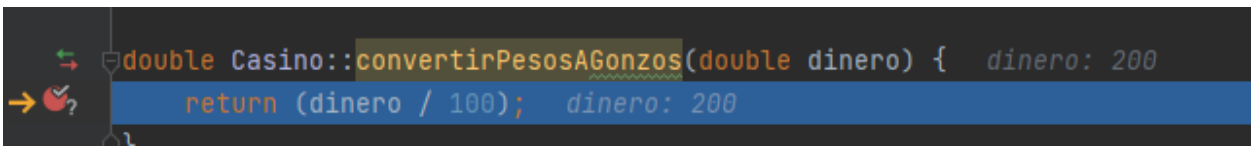


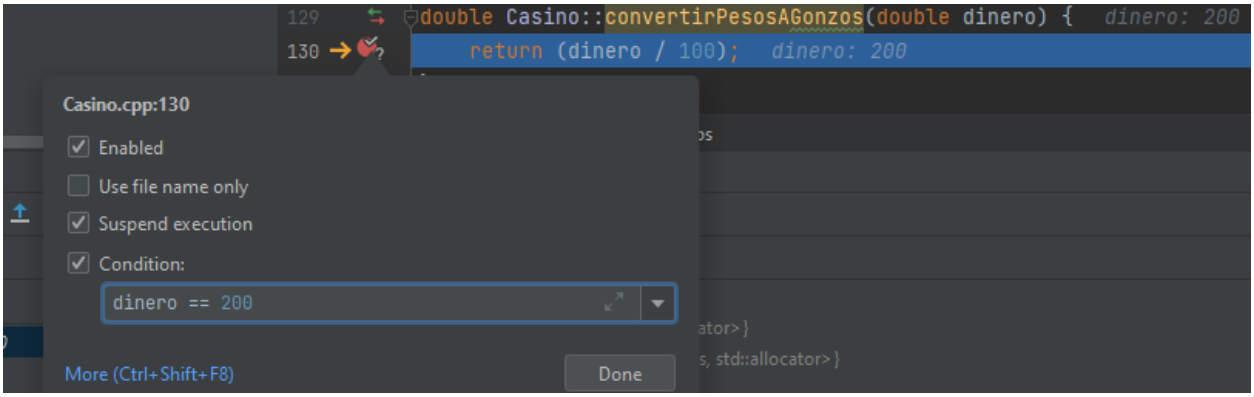
En este caso, tuve que jugar 3 veces hasta que el breakpoint se lanzó:



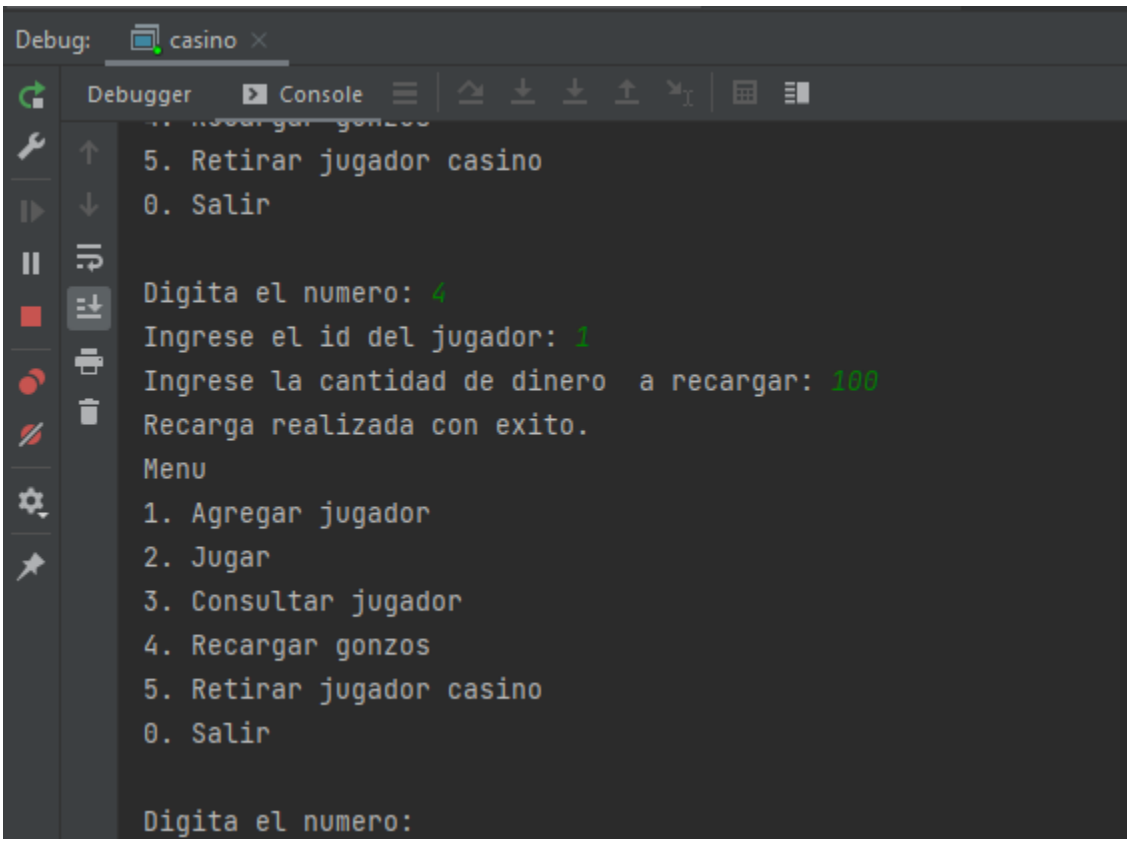
4:

El breakpoint se puso en el método de recargar gonzos:

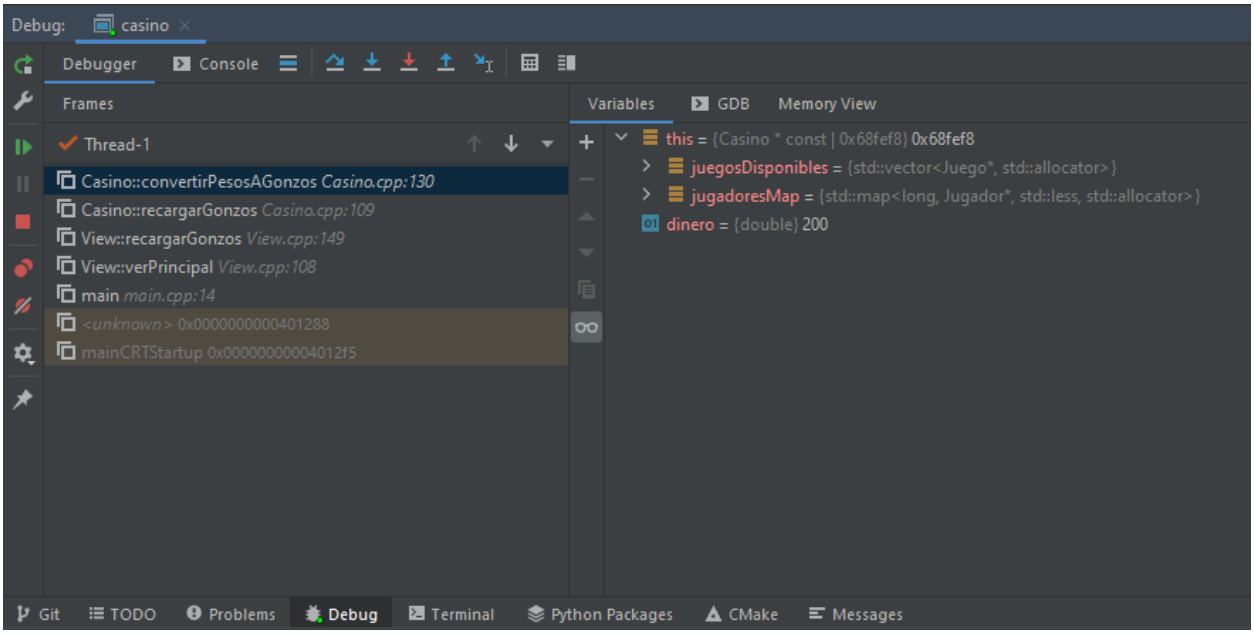




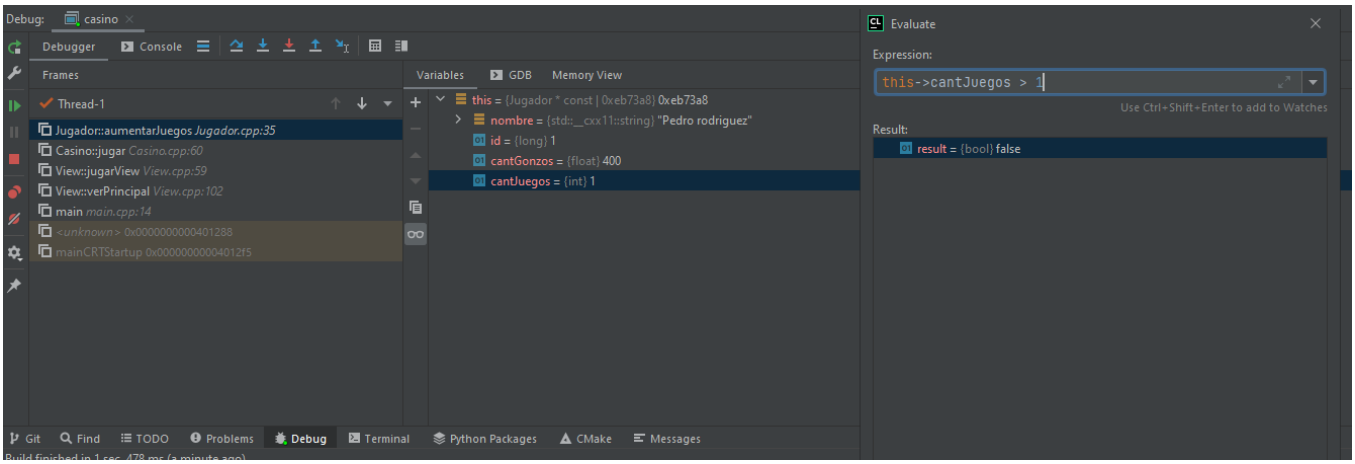
Como podemos ver, aquí no se detuvo ya que no cumple la condición que se le puso al breakpoint:



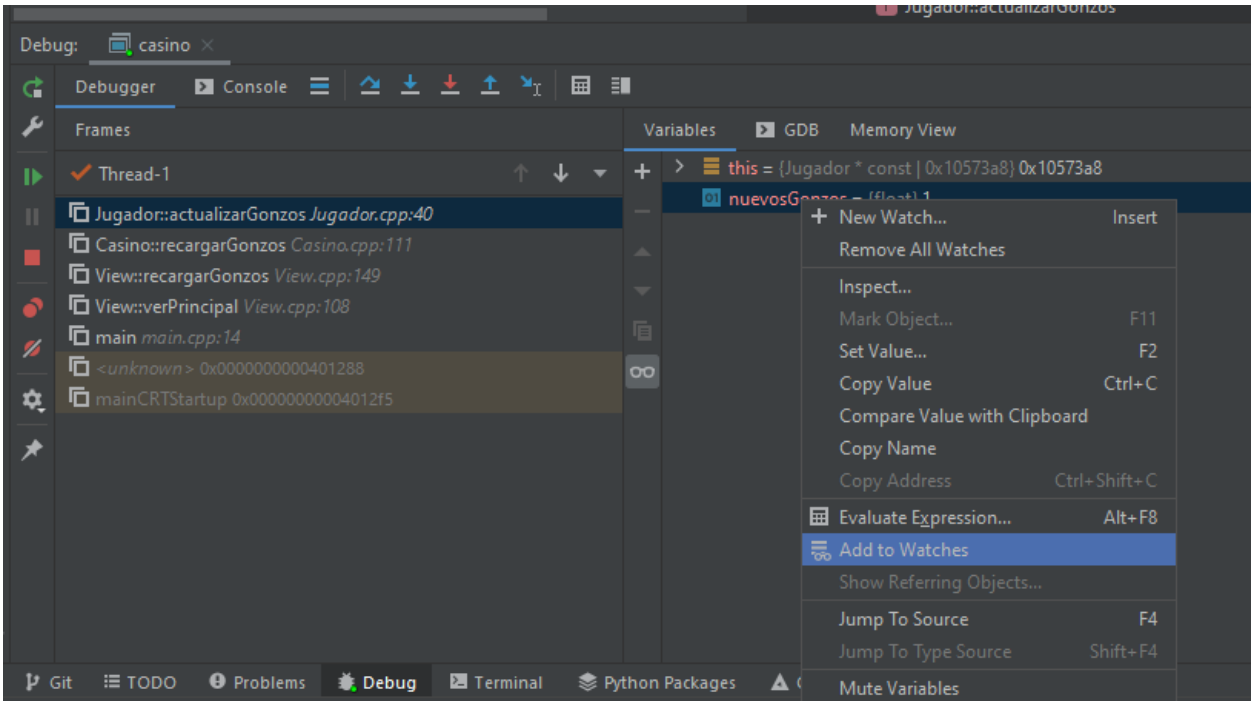
Al cumplirse la condición, se lanza el breakpoint:

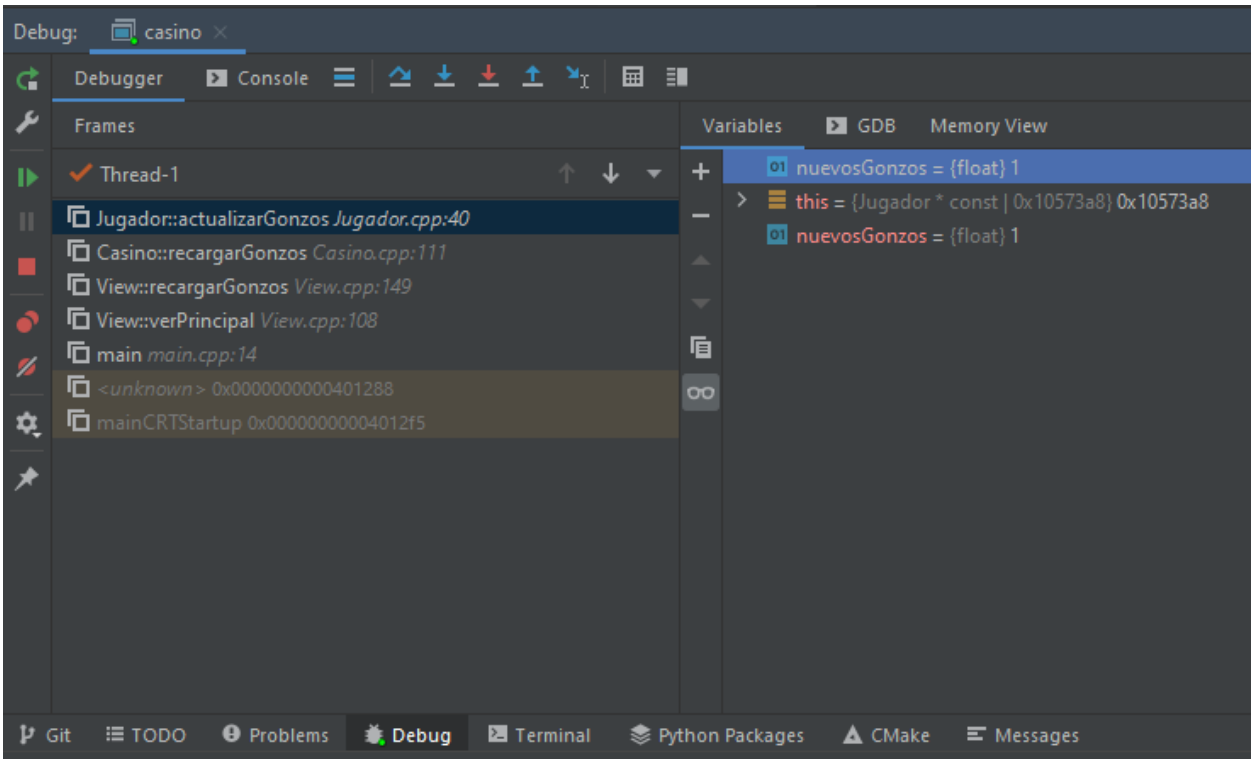


5:



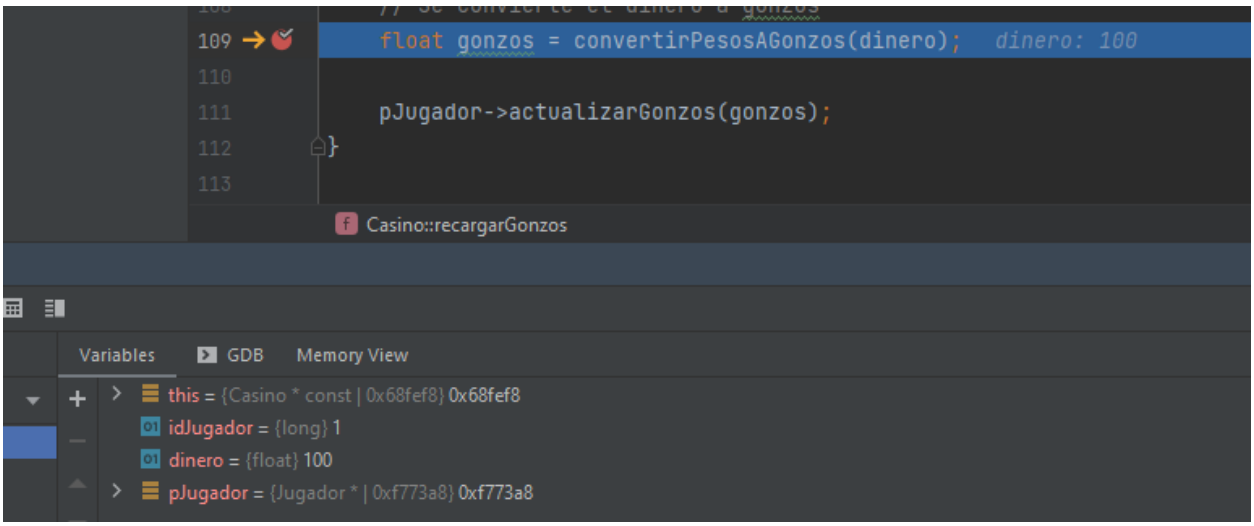
6:


Debug: casino
Debugger
Console
Frames
Thread-1
Jugador::actualizarGonzos Jugador.cpp:40
Casino::recargarGonzos Casino.cpp:111
View::recargarGonzos View.cpp:149
View::verPrincipal View.cpp:108
main main.cpp:14
<unknown> 0x0000000000401288
mainCRTStartup 0x00000000004012f5
Variables
GDB
Memory View
this = {Jugador * const | 0x10573a8} 0x10573a8
New Watch... Insert
Remove All Watches
Inspect...
Mark Object... F11
Set Value... F2
Copy Value Ctrl+C
Compare Value with Clipboard
Copy Name
Copy Address Ctrl+Shift+C
Evaluate Expression... Alt+F8
Add to Watches
Show Referring Objects...
Jump To Source F4
Jump To Type Source Shift+F4
Mute Variables
Git
TODO
Problems
Debug
Terminal
Python Packages

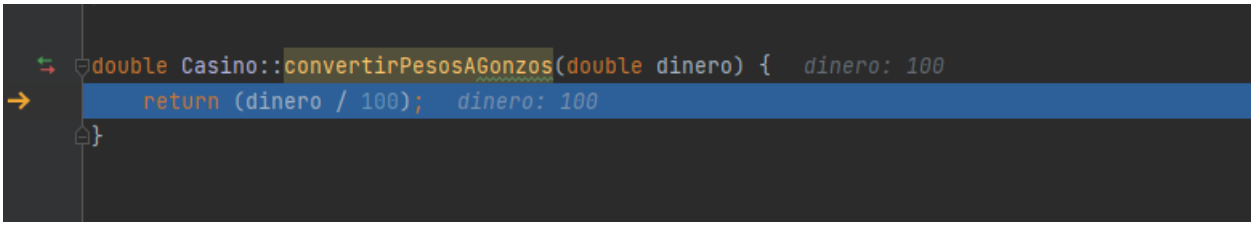


7: El step over (F8) sirve para pasar entre las líneas de código incluso si hay por ejemplo, una función siendo invocada en esa siguiente línea, mientras que el step into (F7) sí se mete a esa función que está siendo llamada. Ejemplo:

Aquí inicia el breakpoint:



Esto pasa al moverse con F7 hasta la línea 111: me lleva directamente a convertir pesos a gonzos.



Pero si utilizo el F8, vemos que me lleva directamente a la otra línea, así haya antes una función invocada: