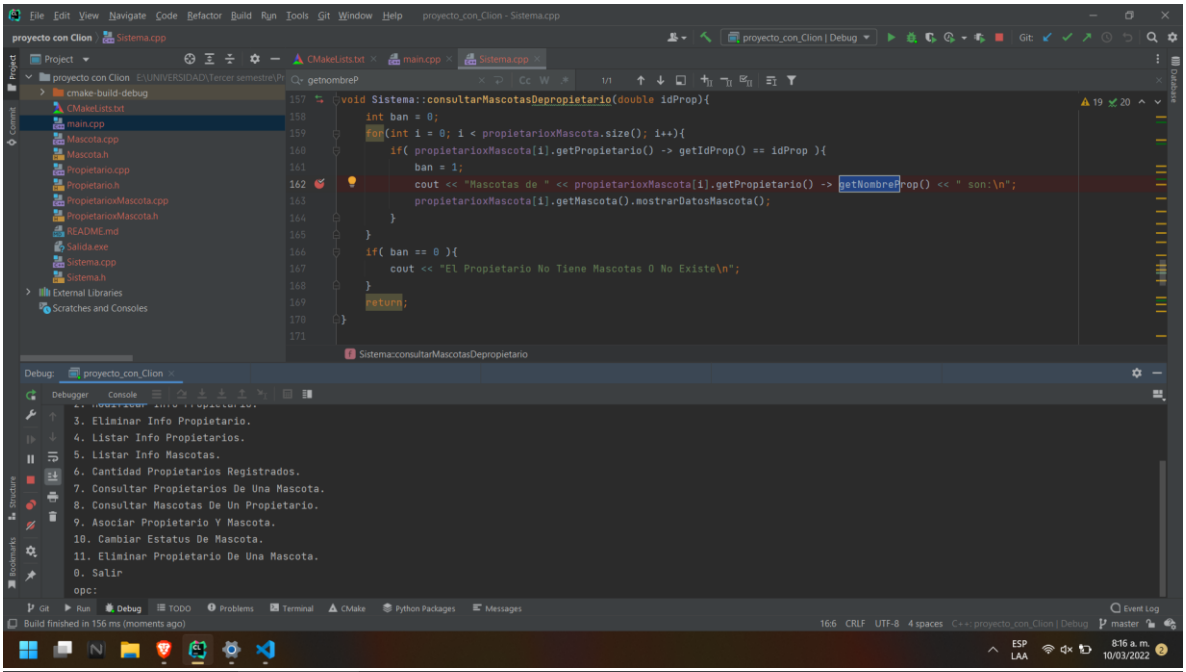
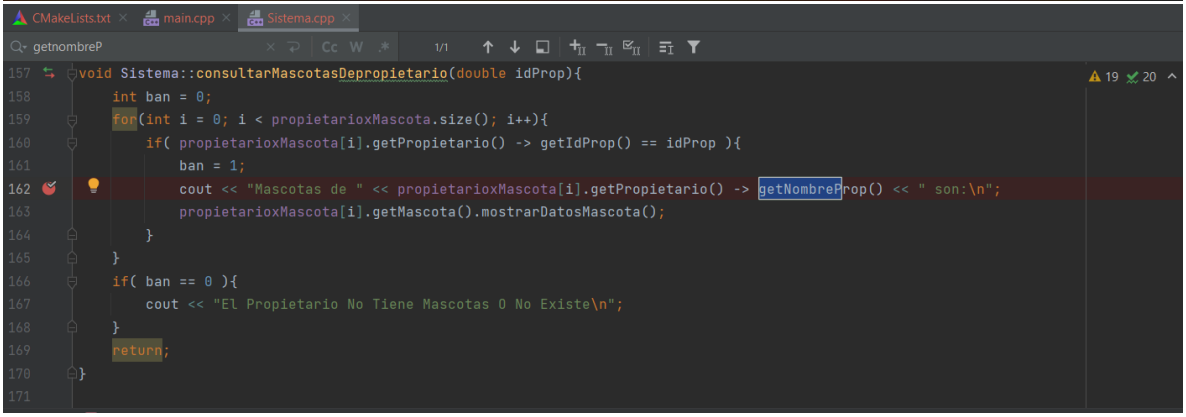
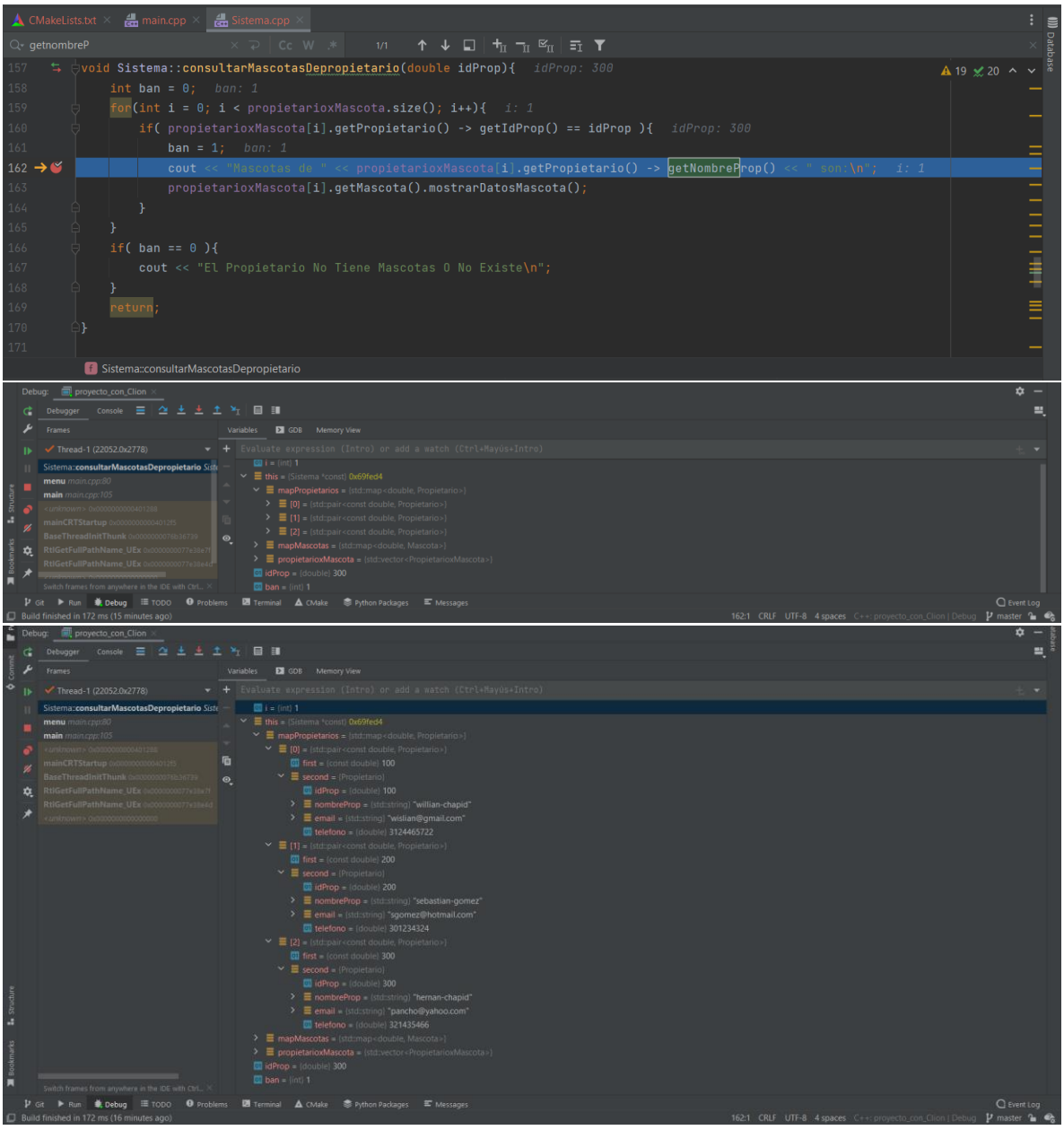
ACTIVIDAD DEBUG

Willian David Chapid Tobar
Programación Orientada a Objetos
2022

1) Breakpoint en getNombre e información hasta ese momento



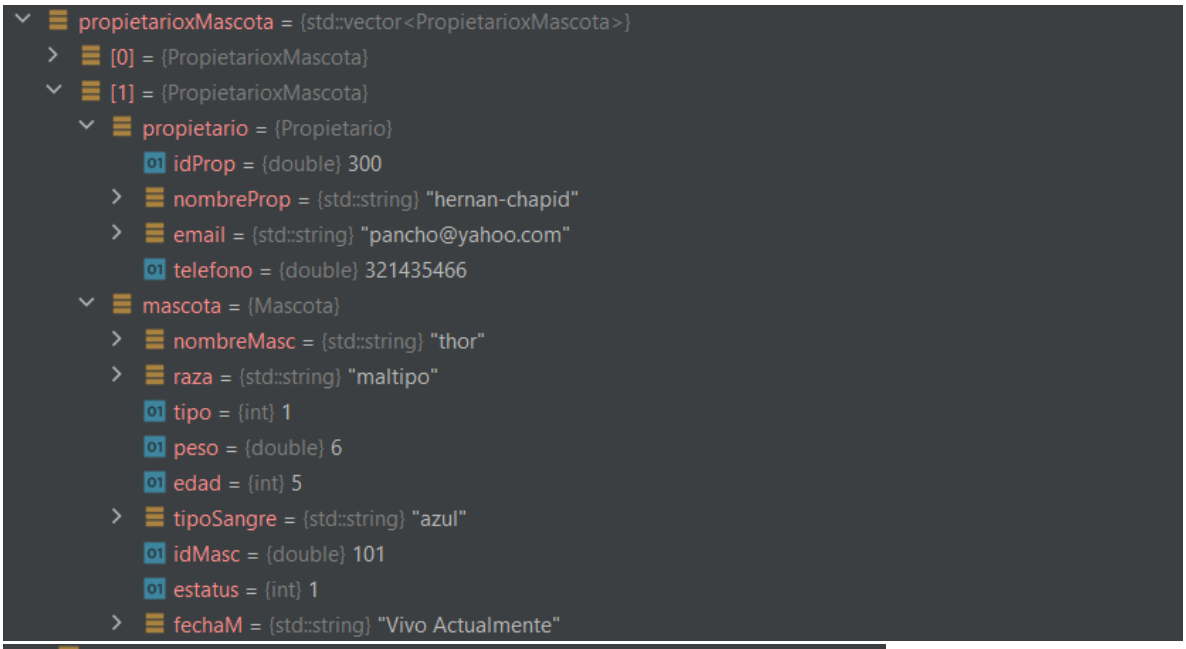


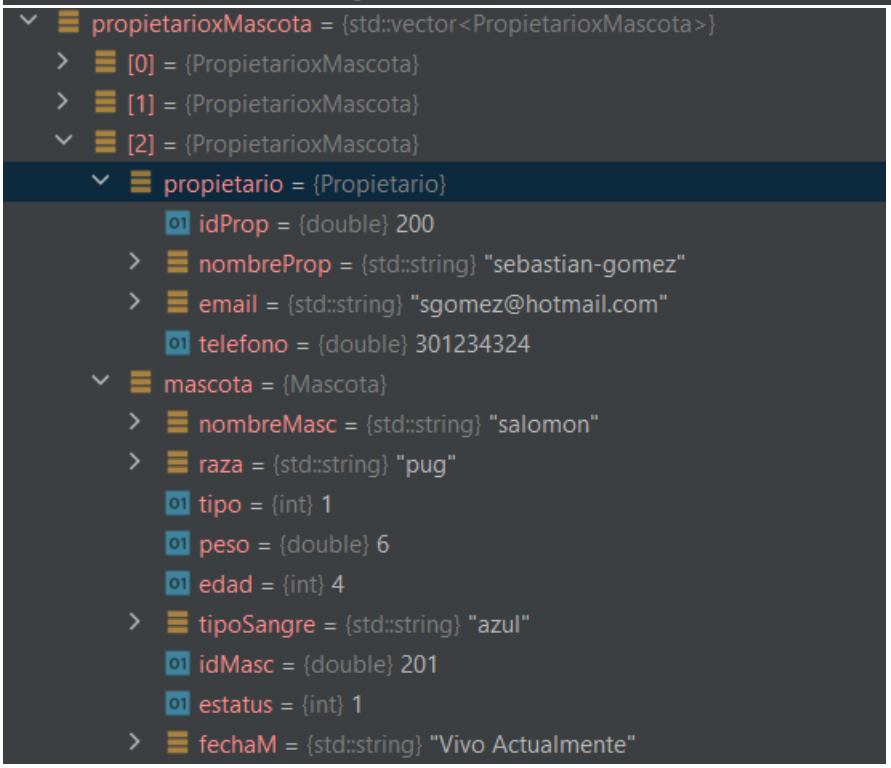

CMakeLists.txt
main.cpp
Sistema.cpp
getnombreP
1/1
19
20
157 void Sistema::consultarMascotasDepropietario(double idProp){ idProp: 300
158 int ban = 0; ban: 1
159 for(int i = 0; i < propietarioxMascota.size(); i++){ i: 1
160 if(propietarioxMascota[i].getPropietario() -> getIdProp() == idProp){ idProp: 300
161 ban = 1; ban: 1
162 cout << "Mascotas de " << propietarioxMascota[i].getPropietario() -> getNombreProp() << " son:\n"; i: 1
163 propietarioxMascota[i].getMascota().mostrarDatosMascota();
164 }
165 }
166 if(ban == 0){
167 cout << "El Propietario No Tiene Mascotas O No Existe\n";
168 }
169 return;
170 }
171
Sistema::consultarMascotasDepropietario
Database

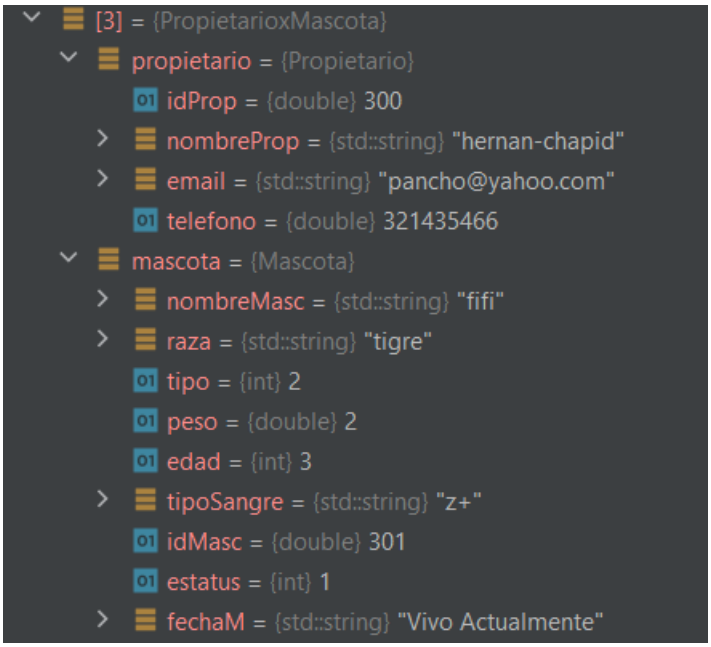
Debug: proyecto_con_Clion
Debugger
Console
Frames
Variables
GDB
Memory View
Thread-1 (22052.0x2778)
Evaluate expression (Intro) or add a watch (Ctrl+Mayús+Intro)
Sistema::consultarMascotasDepropietario
menu main.cpp:80
main main.cpp:105
i = {int} 1
this = {Sistema *const} 0x69fed4
mapPropietarios = {std::map<double, Propietario>}
[0] = {std::pair<const double, Propietario>}
[1] = {std::pair<const double, Propietario>}
[2] = {std::pair<const double, Propietario>}
mapMascotas = {std::map<double, Mascota>}
propietarioxMascota = {std::vector<PropietarioxMascota>}
idProp = {double} 300
ban = {int} 1
Git
Run
Debug
TODO
Problems
Terminal
CMake
Python Packages
Messages
Event Log
Build finished in 172 ms (15 minutes ago)
162:1 CRLF UTF-8 4 spaces
master

Debug: proyecto_con_Clion
Debugger
Console
Frames
Variables
GDB
Memory View
Thread-1 (22052.0x2778)
Evaluate expression (Intro) or add a watch (Ctrl+Mayús+Intro)
i = {int} 1
this = {Sistema *const} 0x69fed4
mapPropietarios = {std::map<double, Propietario>}
[0] = {std::pair<const double, Propietario>}
first = {const double} 100
second = {Propietario}
idProp = {double} 100
nombreProp = {std::string} "willian-chapid"
email = {std::string} "wislian@gmail.com"
telefono = {double} 3124465722
[1] = {std::pair<const double, Propietario>}
first = {const double} 200
second = {Propietario}
idProp = {double} 200
nombreProp = {std::string} "sebastian-gomez"
email = {std::string} "sgomez@hotmail.com"
telefono = {double} 301234324
[2] = {std::pair<const double, Propietario>}
first = {const double} 300
second = {Propietario}
idProp = {double} 300
nombreProp = {std::string} "hernan-chapid"
email = {std::string} "pancho@yahoo.com"
telefono = {double} 321435466
mapMascotas = {std::map<double, Mascota>}
propietarioxMascota = {std::vector<PropietarioxMascota>}
idProp = {double} 300
ban = {int} 1
Git
Run
Debug
TODO
Problems
Terminal
CMake
Python Packages
Messages
Event Log
Build finished in 172 ms (16 minutes ago)
162:1 CRLF UTF-8 4 spaces
master

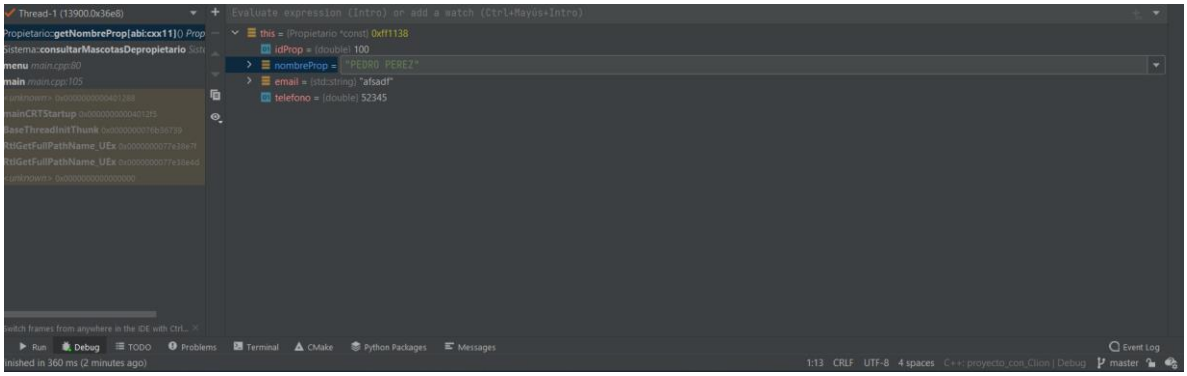

Evaluate expression (Intro) or add a watch (Ctrl+Mayus+Intro)
mapMascotas = {std::map<double, Mascota>}
[0] = {std::pair<const double, Mascota>}
first = {const double} 101
second = {Mascota}
nombreMasc = {std::string} "thor"
raza = {std::string} "maltipo"
tipo = {int} 1
peso = {double} 6
edad = {int} 5
tipoSangre = {std::string} "azul"
idMasc = {double} 101
estatus = {int} 1
fechaM = {std::string} "Vivo Actualmente"
[1] = {std::pair<const double, Mascota>}
first = {const double} 201
second = {Mascota}
nombreMasc = {std::string} "salomon"
raza = {std::string} "pug"
tipo = {int} 1
peso = {double} 6
edad = {int} 4
tipoSangre = {std::string} "azul"
idMasc = {double} 201
estatus = {int} 1
fechaM = {std::string} "Vivo Actualmente"
[2] = {std::pair<const double, Mascota>}
first = {const double} 301
second = {Mascota}
nombreMasc = {std::string} "fifi"
raza = {std::string} "tigre"
tipo = {int} 2
peso = {double} 2
edad = {int} 3
tipoSangre = {std::string} "z+"
idMasc = {double} 301
estatus = {int} 1
fechaM = {std::string} "Vivo Actualmente"
propietarioxMascota = {std::vector<PropietarioxMascota>}
idProp = {double} 300
ban = {int} 1
propietarioxMascota = {std::vector<PropietarioxMascota>}
[0] = {PropietarioxMascota}
propietario = {Propietario}
idProp = {double} 100
nombreProp = {std::string} "willian-chapid"
email = {std::string} "wislian@gmail.com"
telefono = {double} 3124465722
mascota = {Mascota}
nombreMasc = {std::string} "thor"
raza = {std::string} "maltipo"
tipo = {int} 1
peso = {double} 6
edad = {int} 5
tipoSangre = {std::string} "azul"
idMasc = {double} 101
estatus = {int} 1
fechaM = {std::string} "Vivo Actualmente"


propietarioxMascota = {std::vector<PropietarioxMascota>}
[0] = {PropietarioxMascota}
[1] = {PropietarioxMascota}
propietario = {Propietario}
idProp = {double} 300
nombreProp = {std::string} "hernan-chapid"
email = {std::string} "pancho@yahoo.com"
telefono = {double} 321435466
mascota = {Mascota}
nombreMasc = {std::string} "thor"
raza = {std::string} "maltipo"
tipo = {int} 1
peso = {double} 6
edad = {int} 5
tipoSangre = {std::string} "azul"
idMasc = {double} 101
estatus = {int} 1
fechaM = {std::string} "Vivo Actualmente"

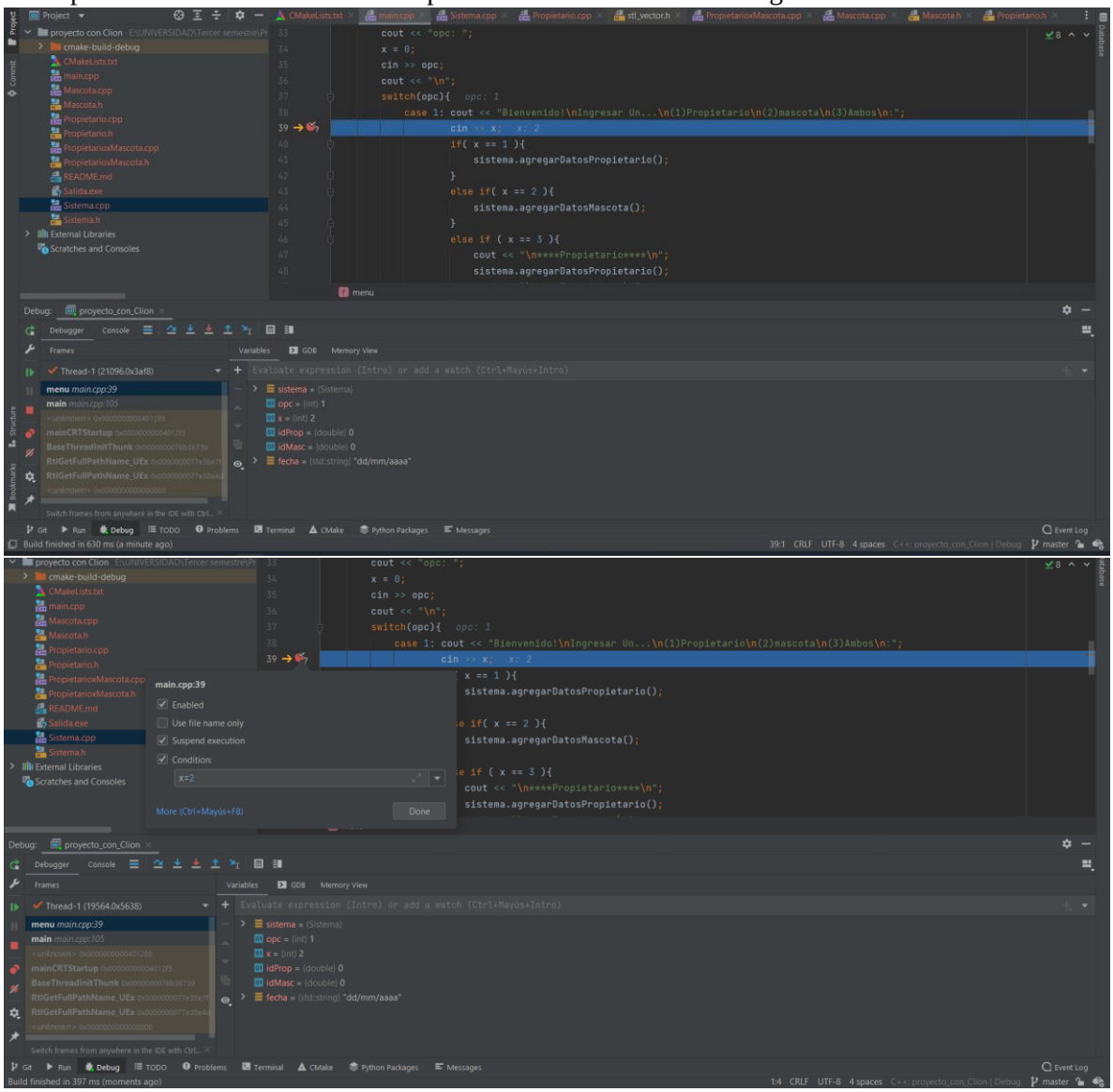

propietarioxMascota = {std::vector<PropietarioxMascota>}
[0] = {PropietarioxMascota}
[1] = {PropietarioxMascota}
[2] = {PropietarioxMascota}
propietario = {Propietario}
idProp = {double} 200
nombreProp = {std::string} "sebastian-gomez"
email = {std::string} "sgomez@hotmail.com"
telefono = {double} 301234324
mascota = {Mascota}
nombreMasc = {std::string} "salomon"
raza = {std::string} "pug"
tipo = {int} 1
peso = {double} 6
edad = {int} 4
tipoSangre = {std::string} "azul"
idMasc = {double} 201
estatus = {int} 1
fechaM = {std::string} "Vivo Actualmente"



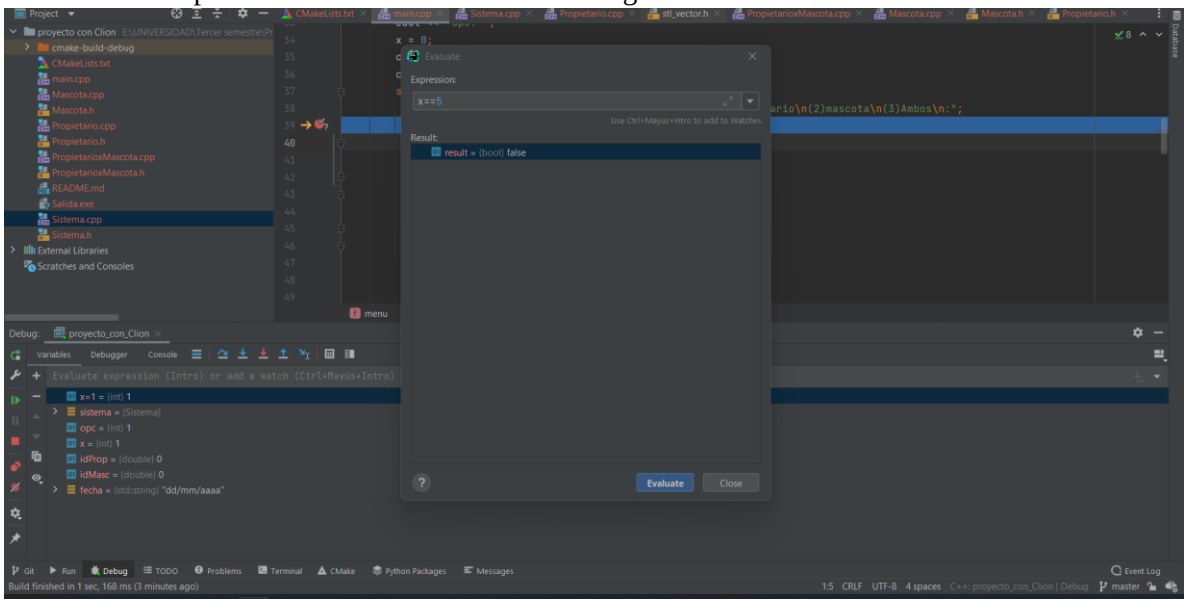
2) Set variable values: cambio de nombre



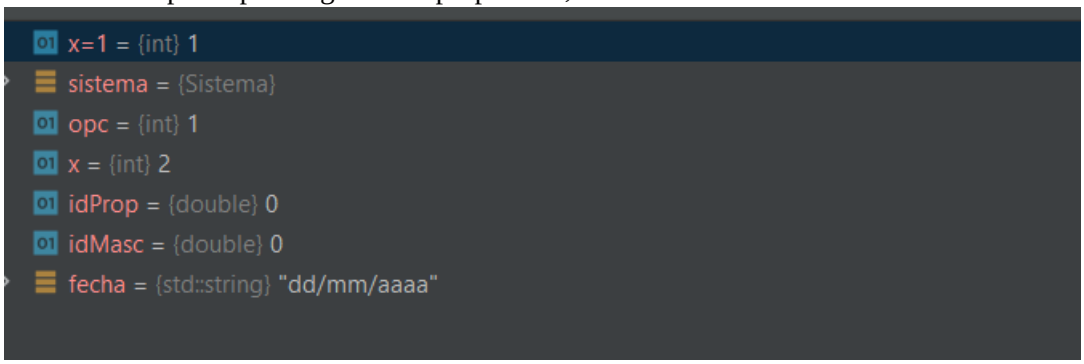
3) Breakpoint condicional: Activar breakpoint cuando la variable x sea igual a 2



4) Prueba de breakpoint condicional evaluando si x es igual a 5 en ese momento



5) Watch de la opción para ingresar un propietario, mascota o ambos



Diferencia entre step over y step into

Step over a la hora de de pasar por la la línea donde se creo el breakpoint, pasa mostrando los valores que se tienen hasta ese momento y sigue hasta la siguiente línea donde se llame la funcionalidad o variable a la que se le aplico el breakpoint mostrando siempre solo la información hasta el momento, mientras que el step into pasa a la línea donde se sitúa el breakpoint y se adentra a las funcionalidades o siguientes instrucciones a realizar mostrando el recorrido del proceso de las siguientes líneas después del breakpoint.

Ejemplo:

En nuestro proyecto colocar un breakpoint con la variable opc que se utiliza para seleccionar una opción en el menú de funciones, con el step over indicara las líneas donde se usa o se cambia la variable, mientras que con el step into después del break point aparte de señalar las líneas donde se usa la variable también seguirá la ejecución de cada línea que siga después de esta, es decir que en el swich el step over señalara, donde se declara, cambia de valor y se evalua en el while, dentro del main, mientras que el step into hará lo mismo pero si opc = 2, entonces se introducira a el case 2: y seguirá la ejecución de líneas que se encuentre ahí.