Reverse Engineering

Week 3: C++, More Difficult Algorithms, Don’t RE what you don’t have to

John Berry, Sergey Bratus -- Dartmouth College -- Winter 2022

Overview

What & Why

- Function Pointers
  - You saw a little with the jump table for switches
- C++
  - With things like virtual functions and name mangling this is a different beast

Motivation

Cool RE

- http://tech.mit.edu/V128/N30/subway/Defcon_Presentation.pdf
- This example is more than just reverse engineering and their story is funny. They wanted to give a talk but the MBTA tried to silence them.
  - https://www.discovermagazine.com/technology/mit-students-who-hacked-boston-subway-silenced-report-gets-out-anyway
  - Streisand effect

Function Pointers

What & Why

- Like regular pointers but they point to executable code.
- Allows a programmer to dynamically select functions to execute
- Will show up in assembly as: *call <register>*

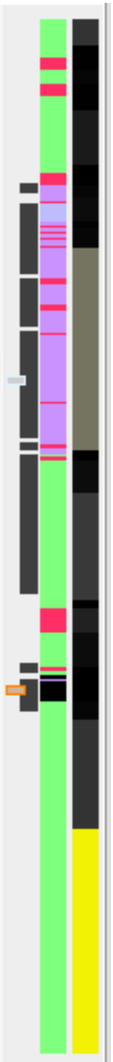
Function Pointers

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

void even() { printf("Even\n"); }
void odd()  { printf("Odd\n"); }

int main(int argc, char **argv)
{
  if ( argc != 2 ) {
    exit(0);
  }

  void (*parity)() = NULL;

  int x = atoi(argv[1]);

  if ( x % 2 ) {
    parity = &odd;
  } else {
    parity = &even;
  }

  parity();

  return 0;
}
"function_pointer.c" 26L, 334C written                    3,0-1         All
```

Function Pointer

Listing

```
                    LAB_001011da                                    XREF[1]:     001011ce(j)
001011da 48 c7 45        MOV        qword ptr [RBP + parity],0x0
         f8 00 00
         00 00
001011e2 48 8b 45 e0     MOV        RAX,qword ptr [RBP + argv]
001011e6 48 83 c0 08     ADD        RAX,0x8
001011ea 48 8b 00        MOV        RAX,qword ptr [RAX]
001011ed 48 89 c7        MOV        RDI,RAX
001011f0 e8 8b fe        CALL       <EXTERNAL>::atoi                                          int
         ff ff
001011f5 89 45 f4        MOV        dword ptr [RBP + arg1_val],EAX
001011f8 8b 45 f4        MOV        EAX,dword ptr [RBP + arg1_val]
001011fb 83 e0 01        AND        EAX,0x1
001011fe 85 c0           TEST       EAX,EAX
00101200 74 0d           JZ         LAB_0010120f
00101202 48 8d 05        LEA        RAX,[even]
         80 ff ff ff
00101209 48 89 45 f8     MOV        qword ptr [RBP + parity],RAX=>even
0010120d eb 0b           JMP        LAB_0010121a

                    LAB_0010120f                                    XREF[1]:     00101200(j)
0010120f 48 8d 05        LEA        RAX,[odd]
         8a ff ff ff
00101216 48 89 45 f8     MOV        qword ptr [RBP + parity],RAX=>odd

                    LAB_0010121a                                    XREF[1]:     0010120d(j)
0010121a 48 8b 55 f8     MOV        RDX,qword ptr [RBP + parity]
0010121e b8 00 00        MOV        EAX,0x0
         00 00
00101223 ff d2           CALL       RDX=>even                                                 und
                                                                                              und
00101225 b8 00 00        MOV        EAX,0x0
         00 00
```



C++

What

- C++ has some features that make reverse engineering it bit more difficult
- Classes
  - Like structures but with functions, and more complicated rules of access
- Vftables
  - A Virtual Function Table is an array of function pointers for the virtual functions of a class.
- Polymorphism and Class Inheritance
  - Results in multiple vftables and a more complex organization of the data structure
- Constructors / Destructors

C++

Identification

- How to know if you are up against a C++ binary?
  - Lots of usage of RDI-relative addressing in g++-compiled binaries, RCX in Windows binaries.
    - Contains the *this* pointer when calling class functions
  - You may see some weird mangled functions names like:
    - _ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc

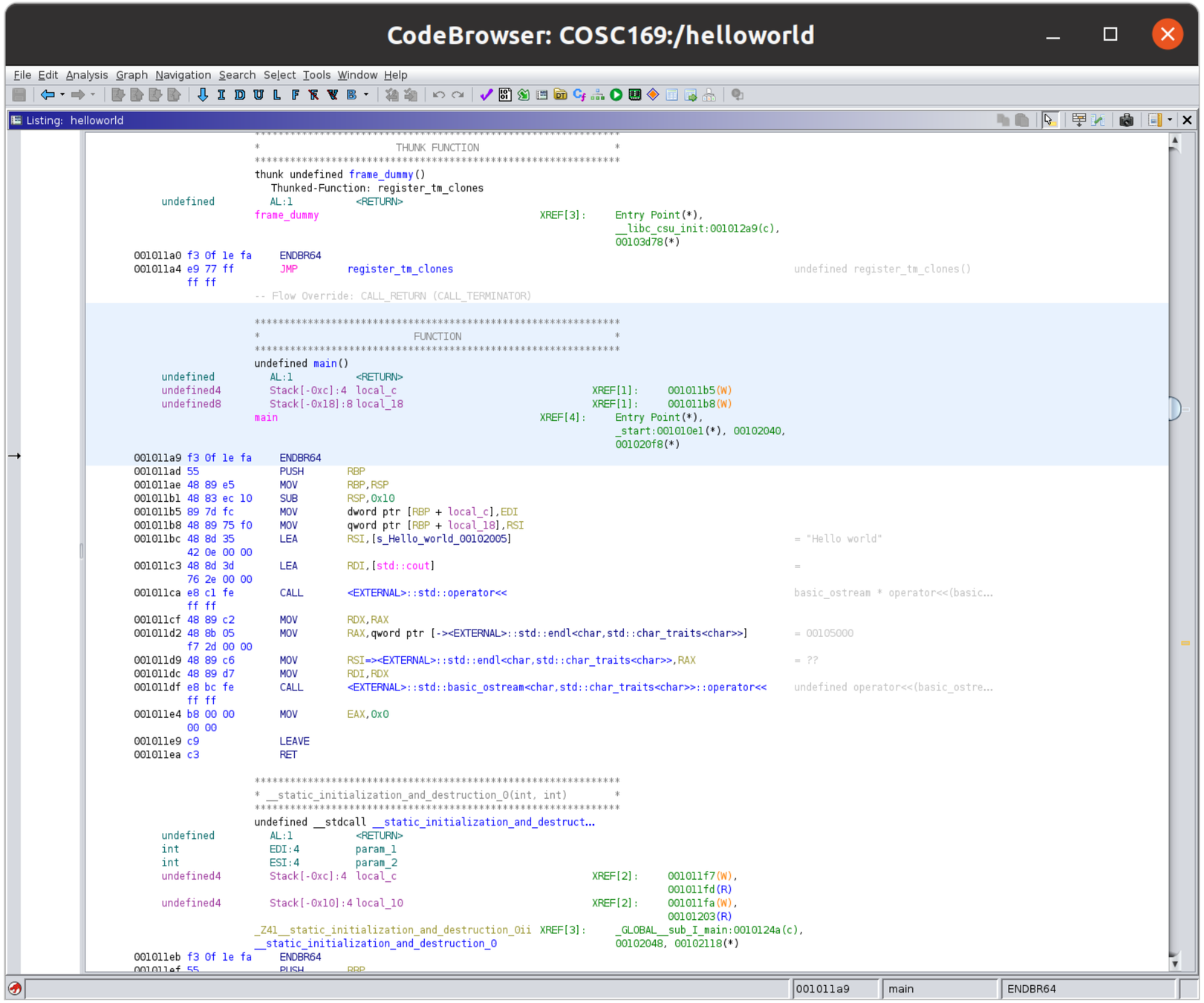
C++

Demangling

- The mangled names mentioned earlier are not actually a C++ standard. Each C++ compiler has their own format
  - Wikipedia article on Name Mangling has good examples
- Fortunately, Ghidra, Binary Ninja, and IDA can all demangle mostly automatically

C++

Demangling



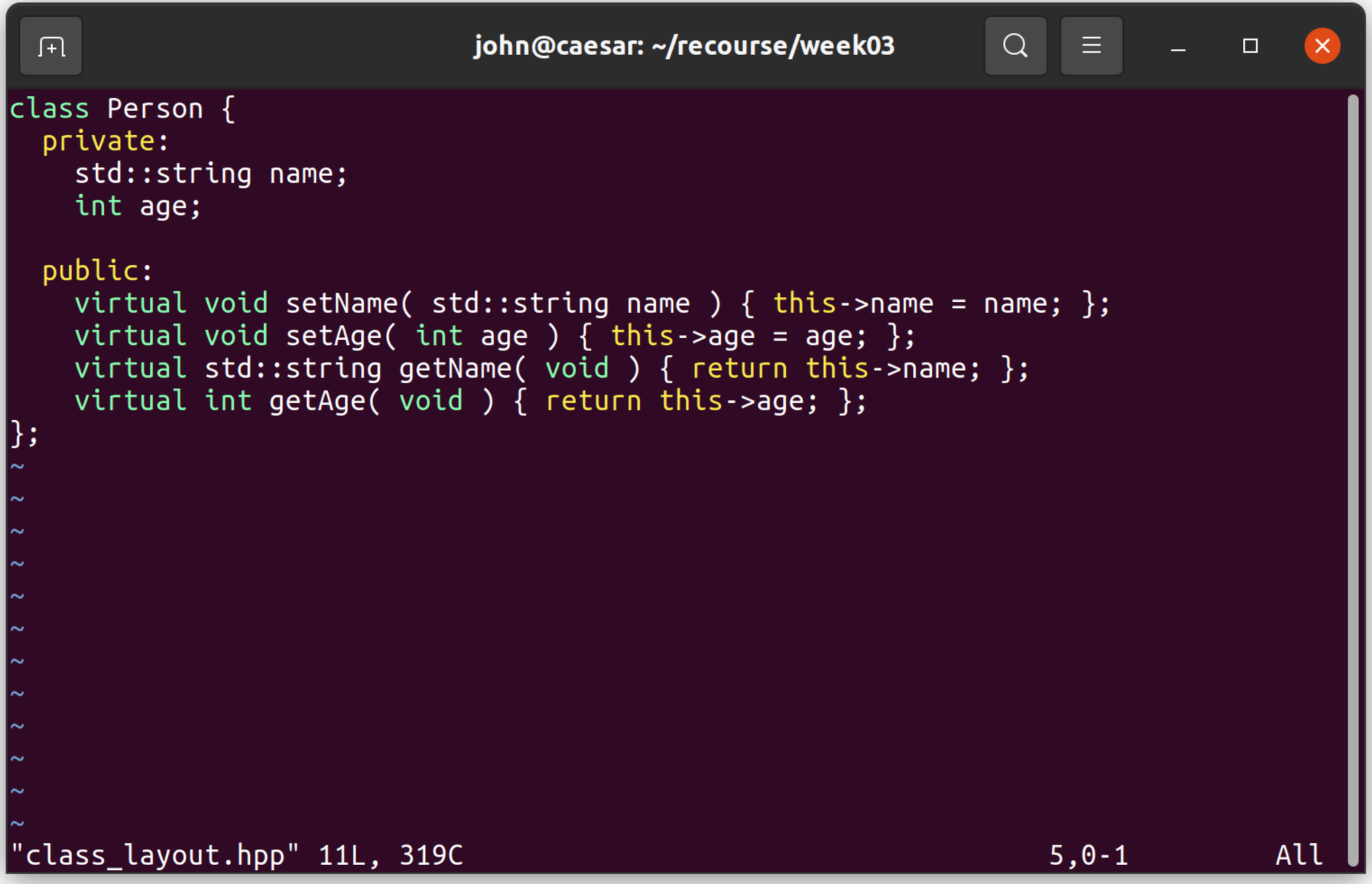
C++

Classes

- You likely know what classes are already but what do they look like in memory?
- Let's take a look at a basic example

C++

Example


john@caesar: ~/recourse/week03
class Person {
private:
std::string name;
int age;
public:
virtual void setName(std::string name) { this->name = name; };
virtual void setAge(int age) { this->age = age; };
virtual std::string getName(void) { return this->name; };
virtual int getAge(void) { return this->age; };
};
"class_layout.hpp" 11L, 319C
5,0-1
All

C++

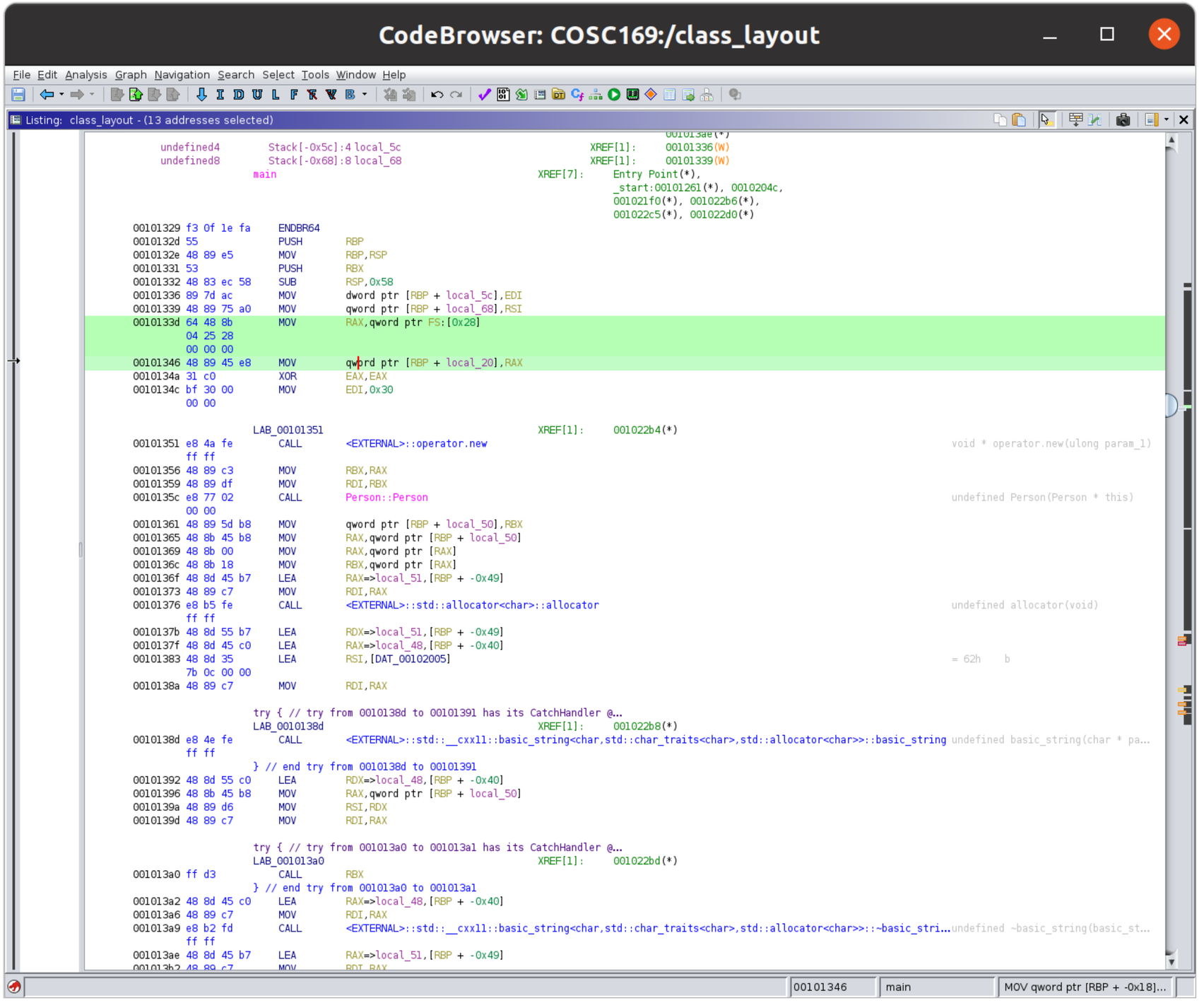
Example

john@caesar: ~/recourse/week03

```
#include <iostream>

#include "class_layout.hpp"

int main(int argc, char **argv)
{
  Person *bob = new Person;

  bob->setName("bob");
  bob->setAge(42);

  std::cout << "Name: " << bob->getName() << " Age: " << bob->getAge() << std::endl;

  return 0;
}
~
~
~
~
~
~
~
~
~
                                                                    2,0-1         All
```

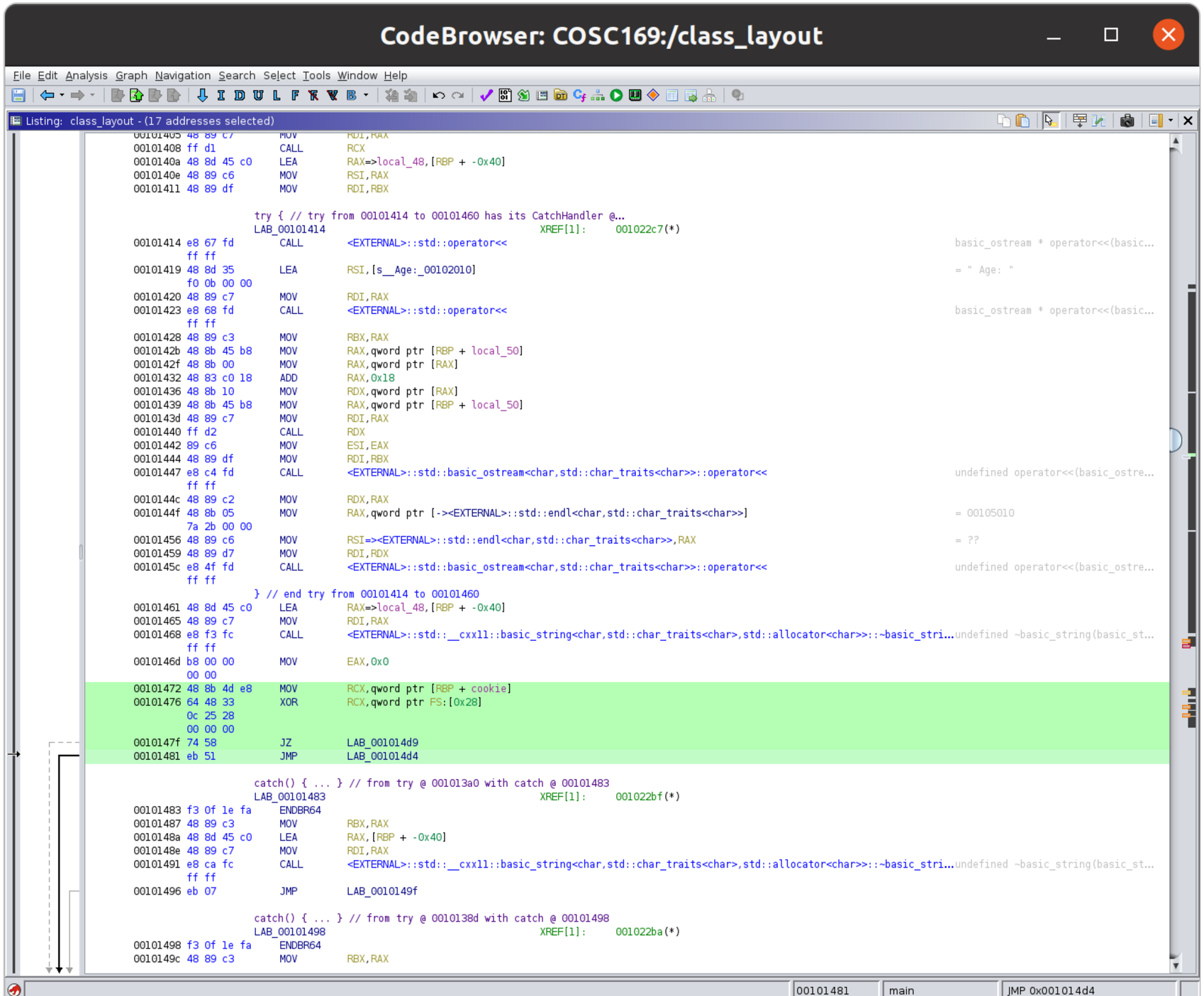
Cookies: A Segue

Not just bad for your teeth



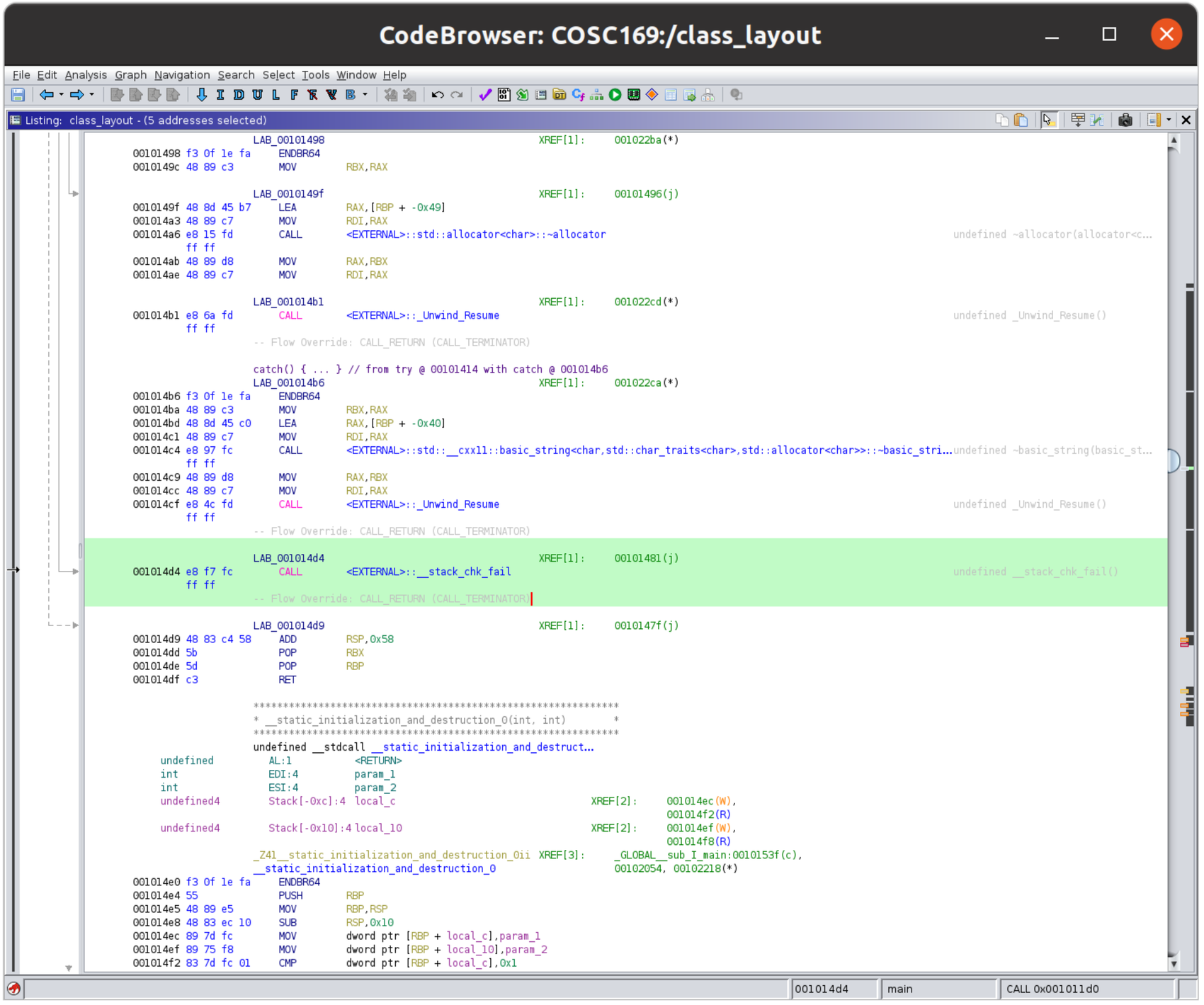
Cookies: A Segue

Not just bad for your teeth



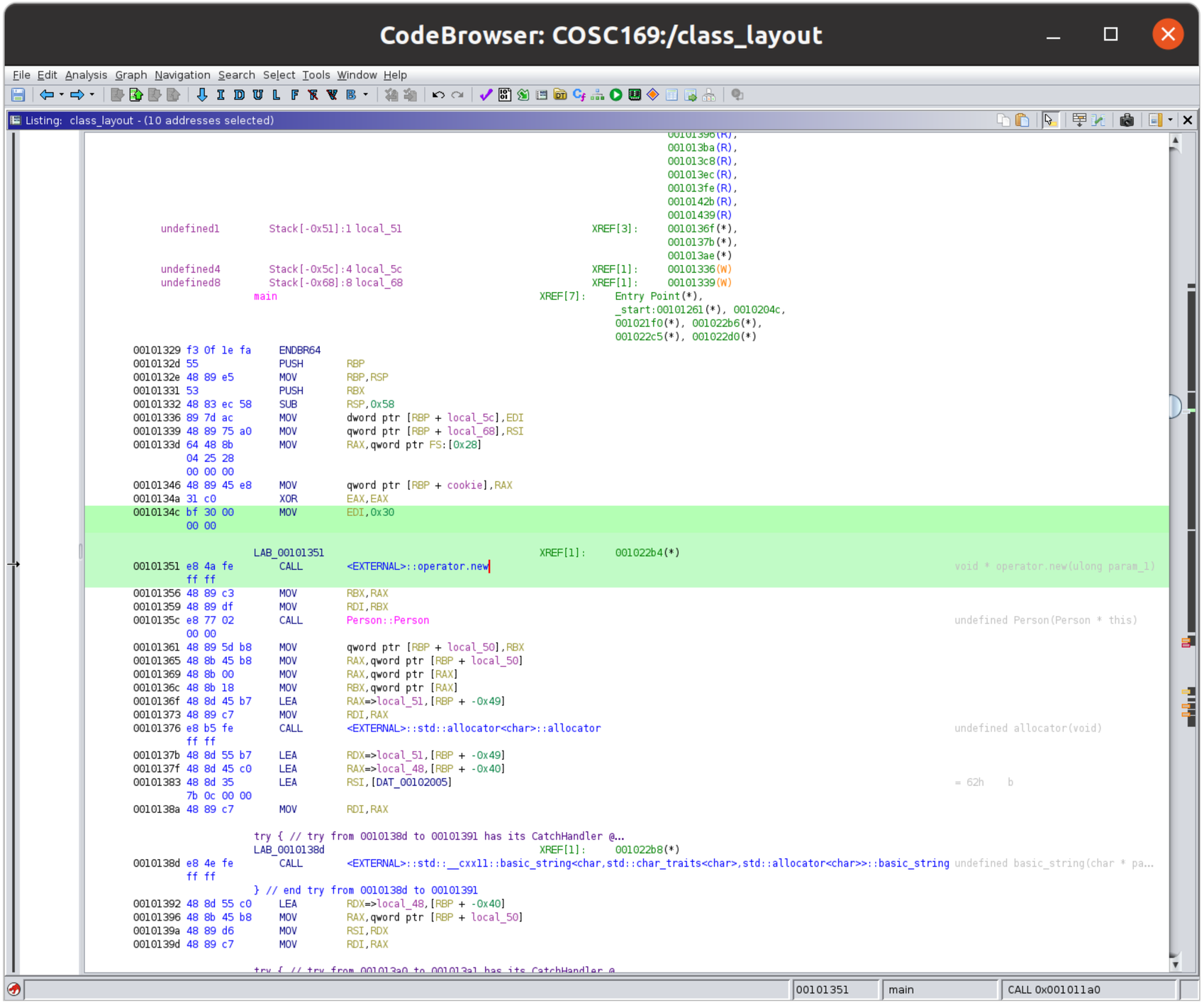
Cookies: A Segue

Not just bad for your teeth



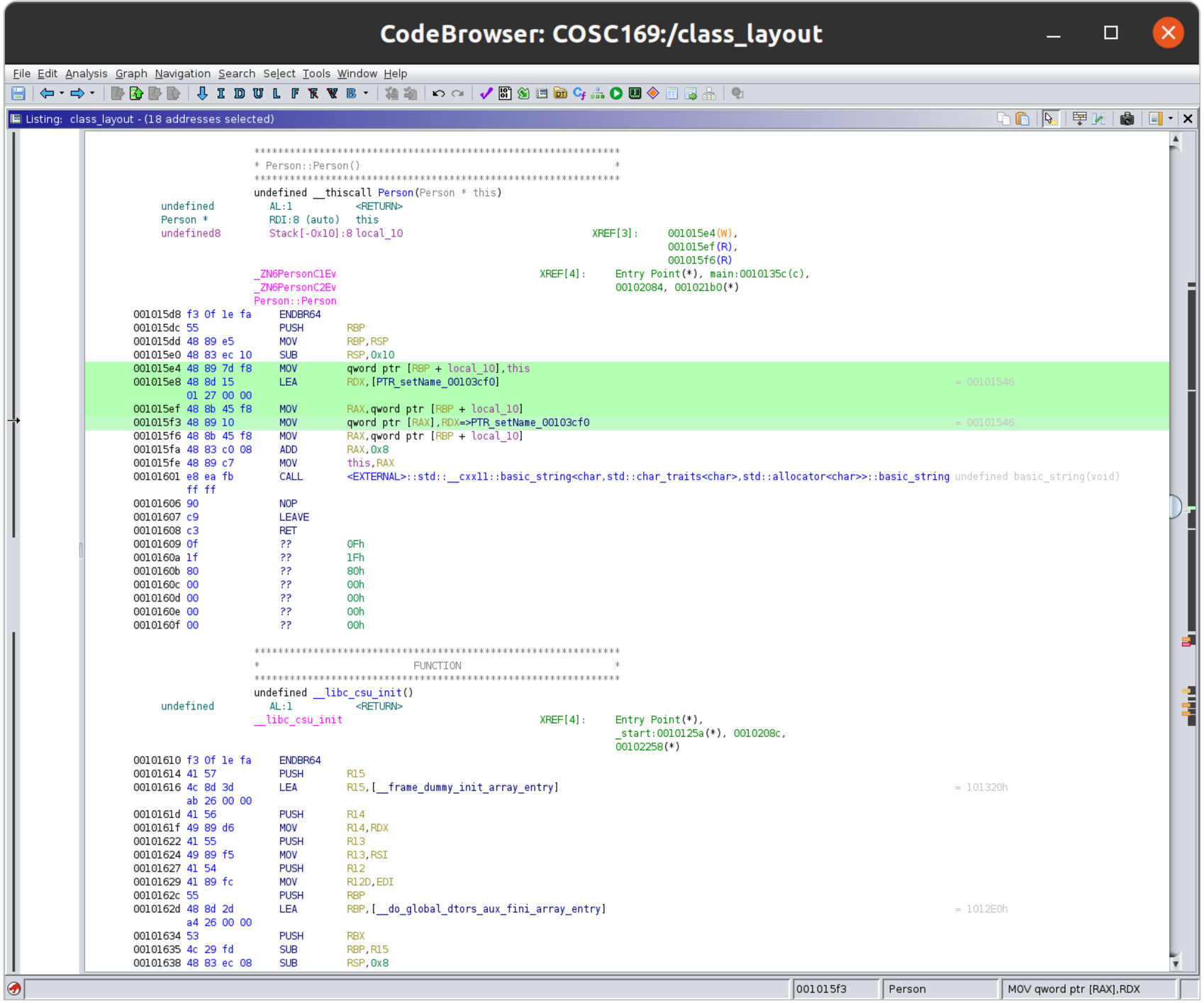
C++

Allocation



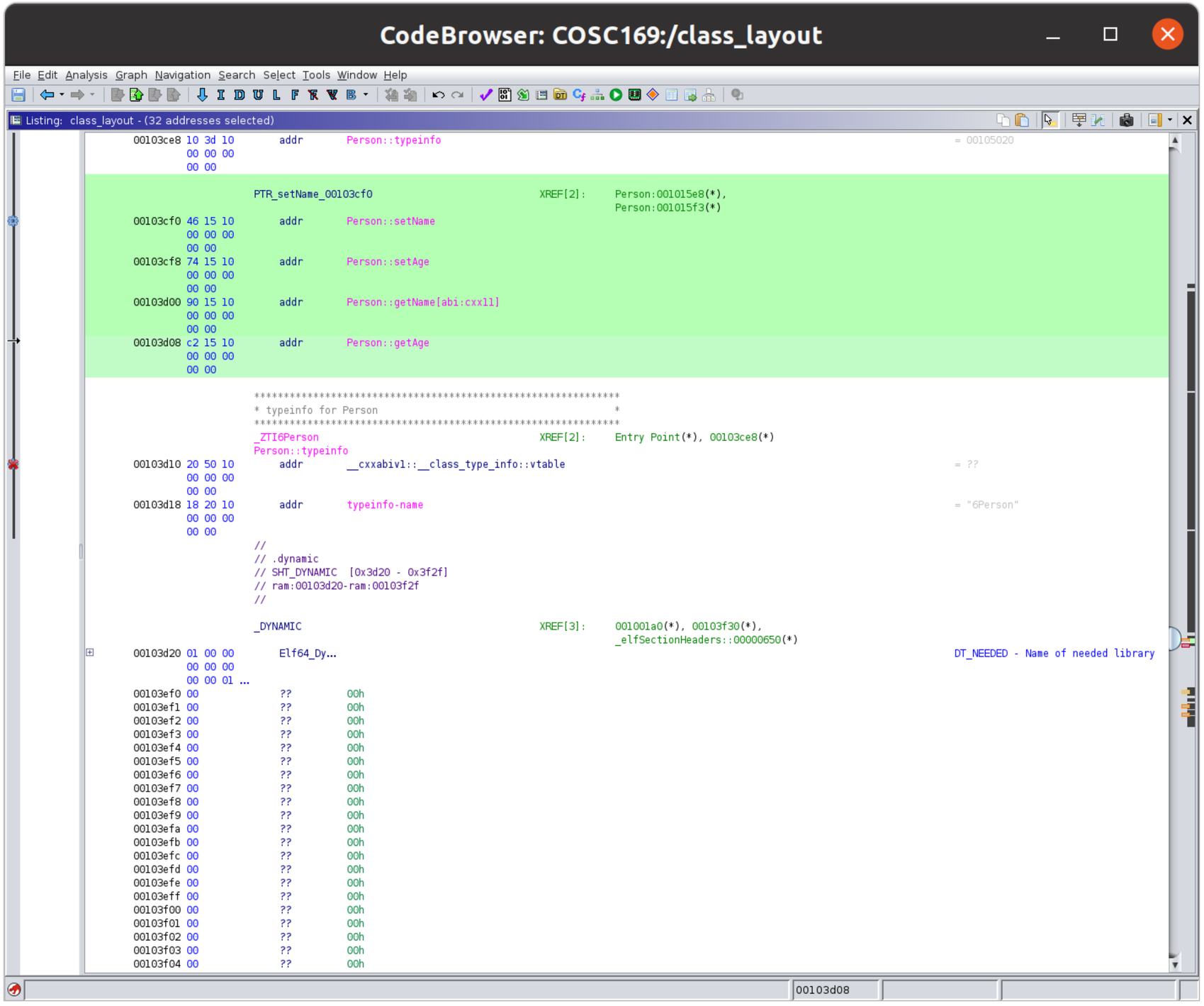
C++

Setting up the Vftable



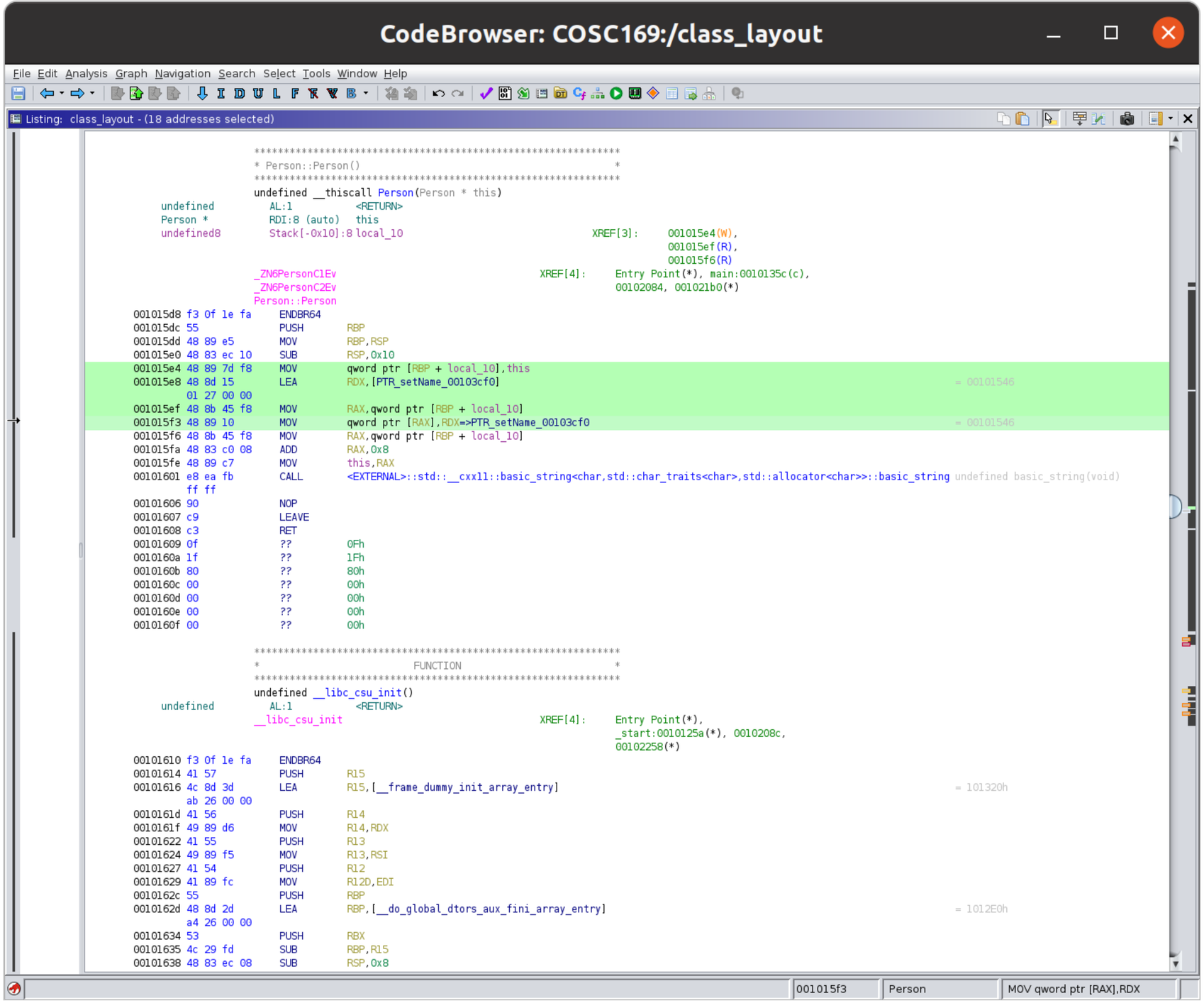
C++

The Vftable



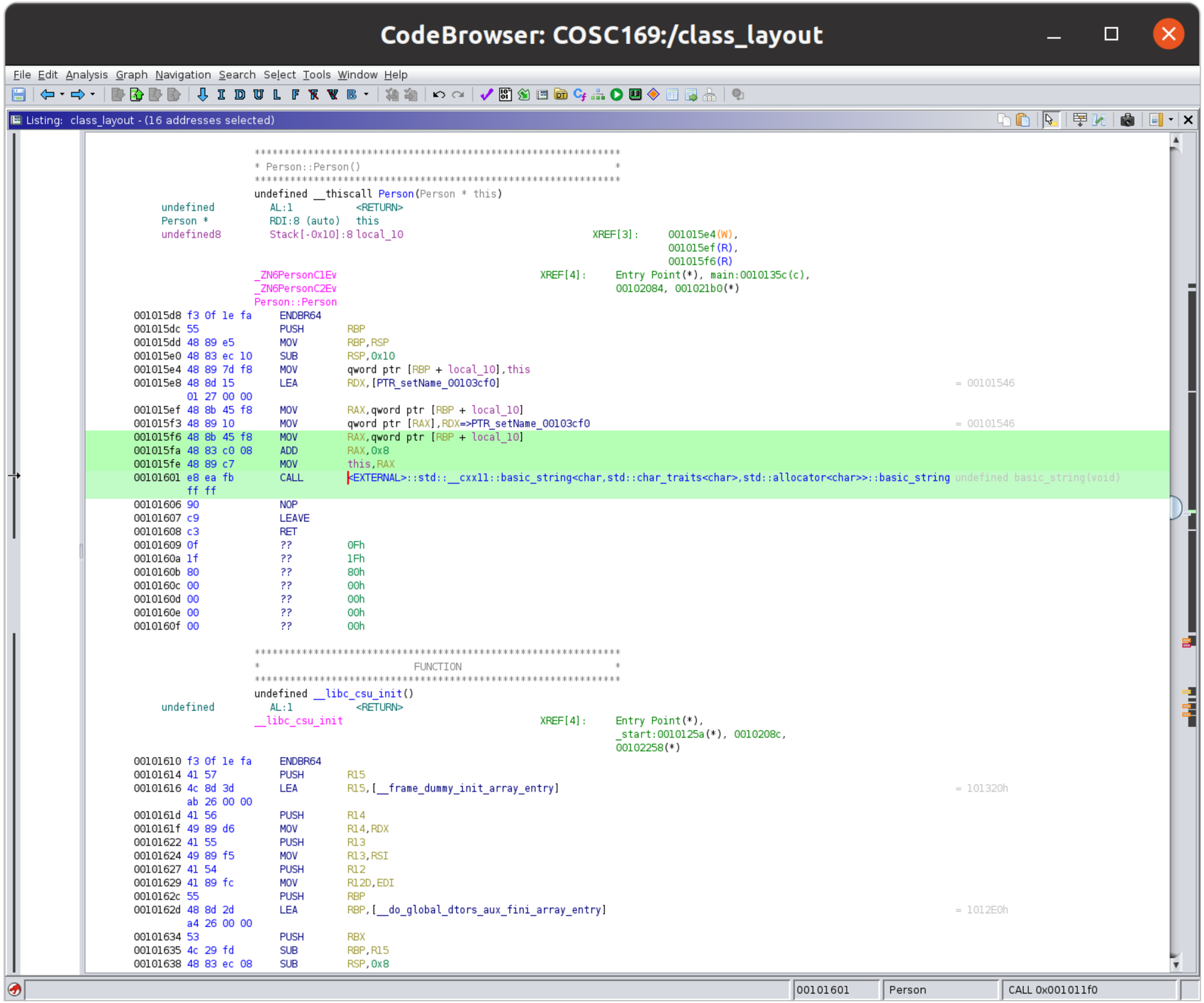
C++

Setting up the Vftable



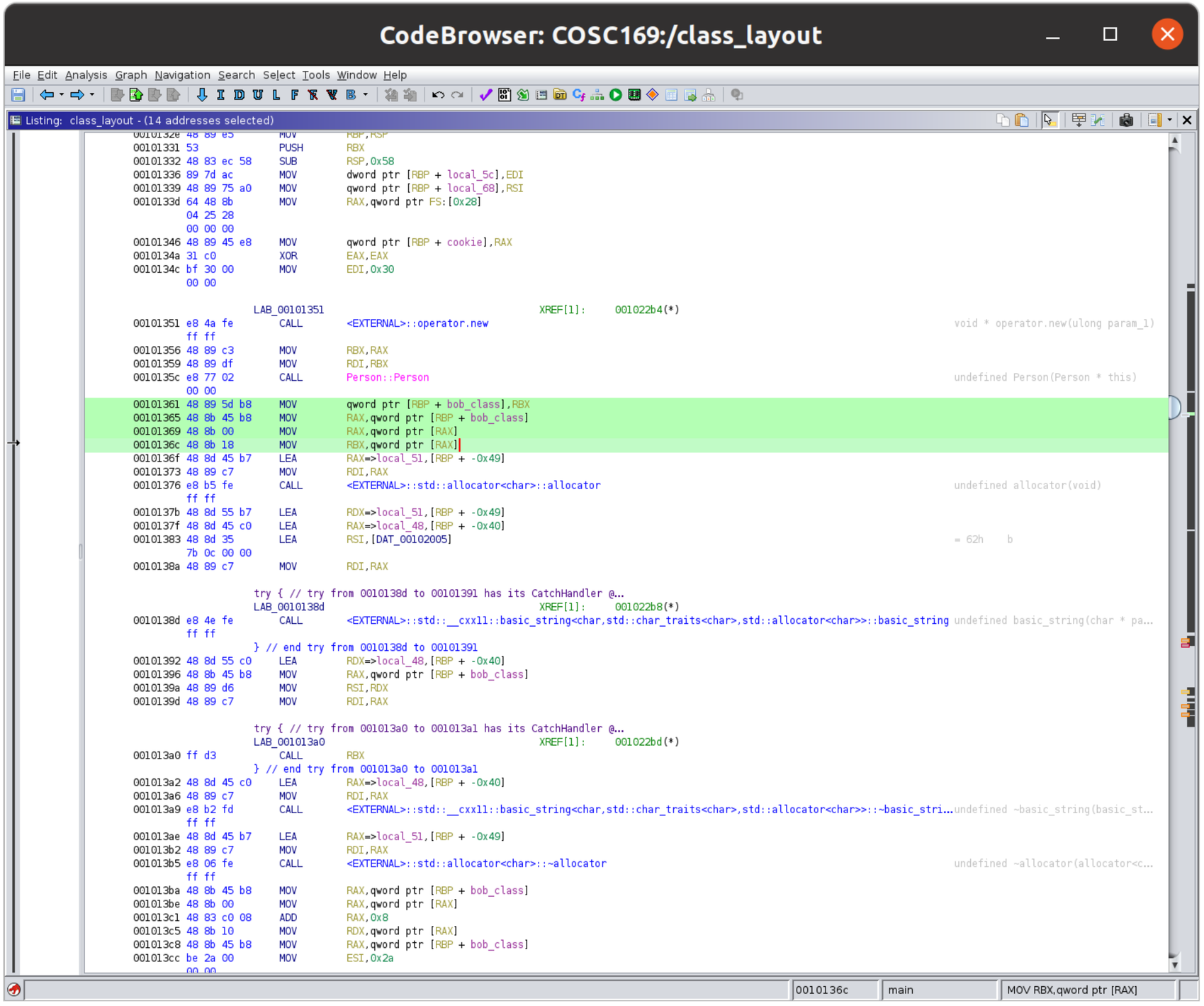
C++

Initializing the name string



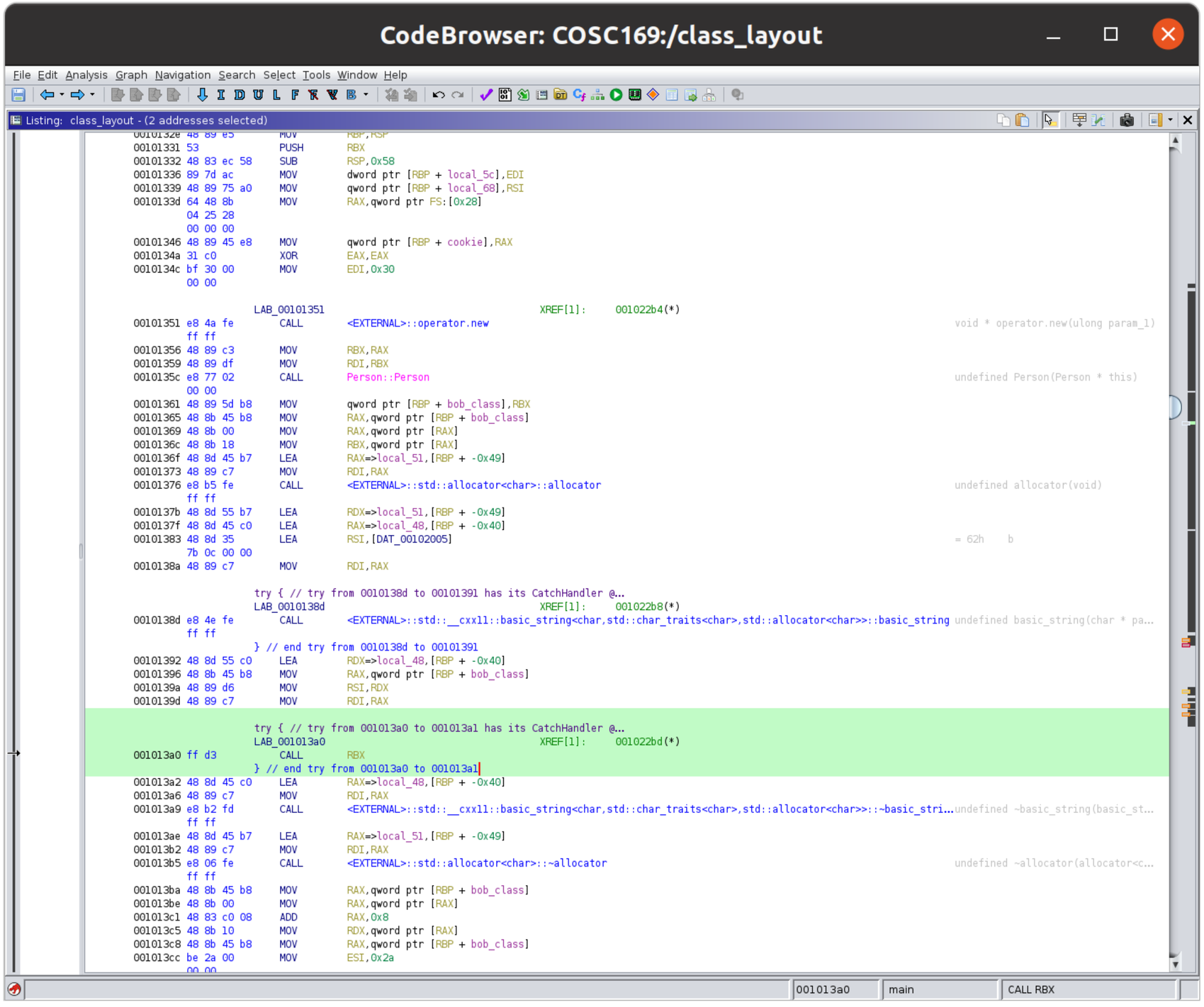
C++

Calling a virtual function



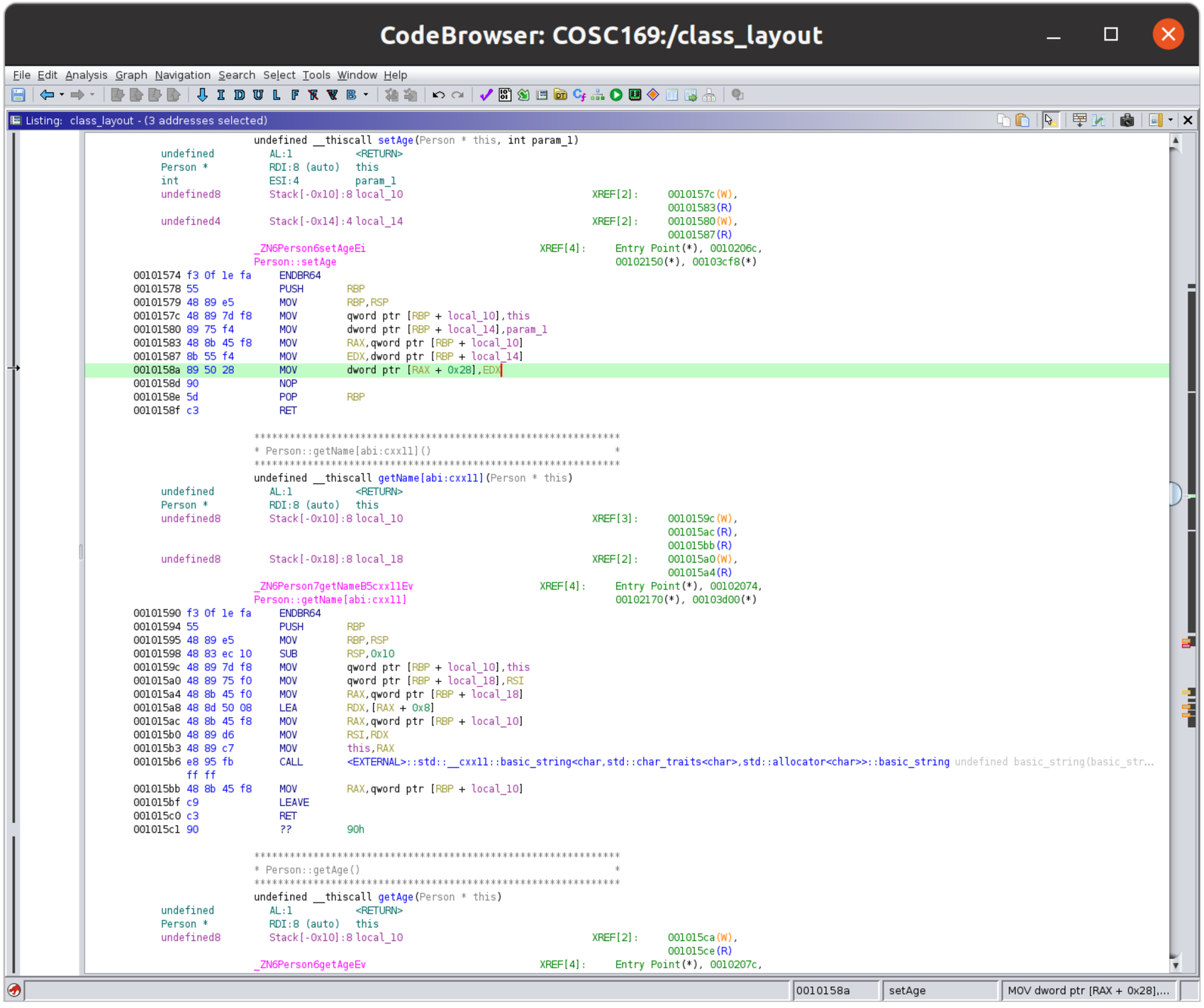
C++

Calling a virtual function



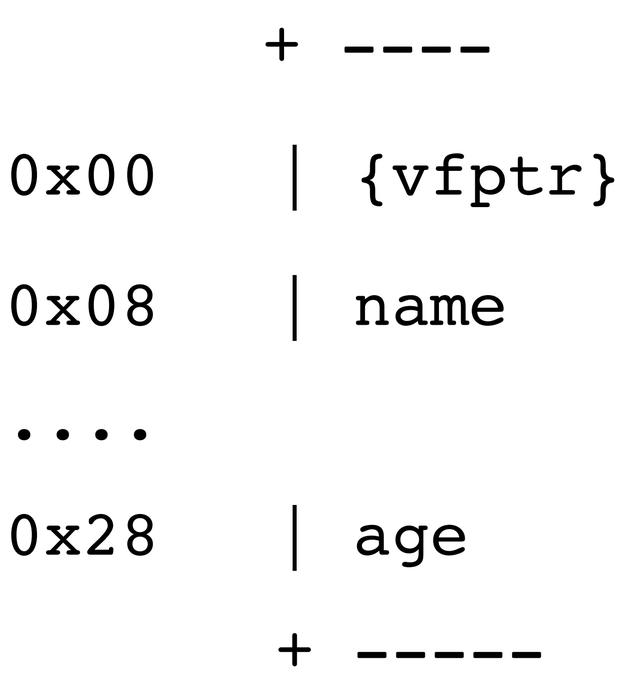
C++

Where is the age field?



C++

Layout of bob



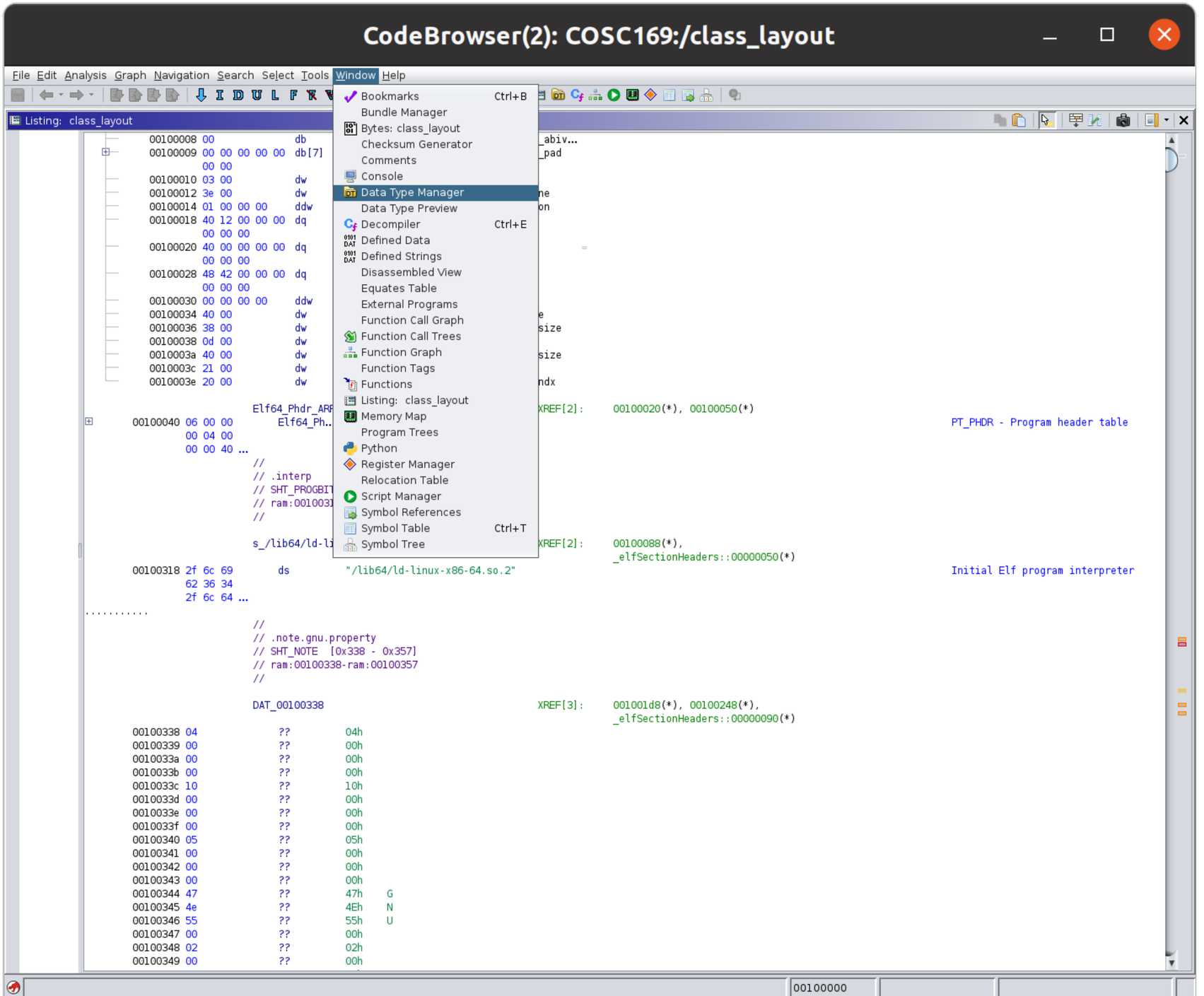
C++

Ghidra tip: Defining Data Types

- Up to this point we have only updated the disassembly with variable names but now we should start defining data types

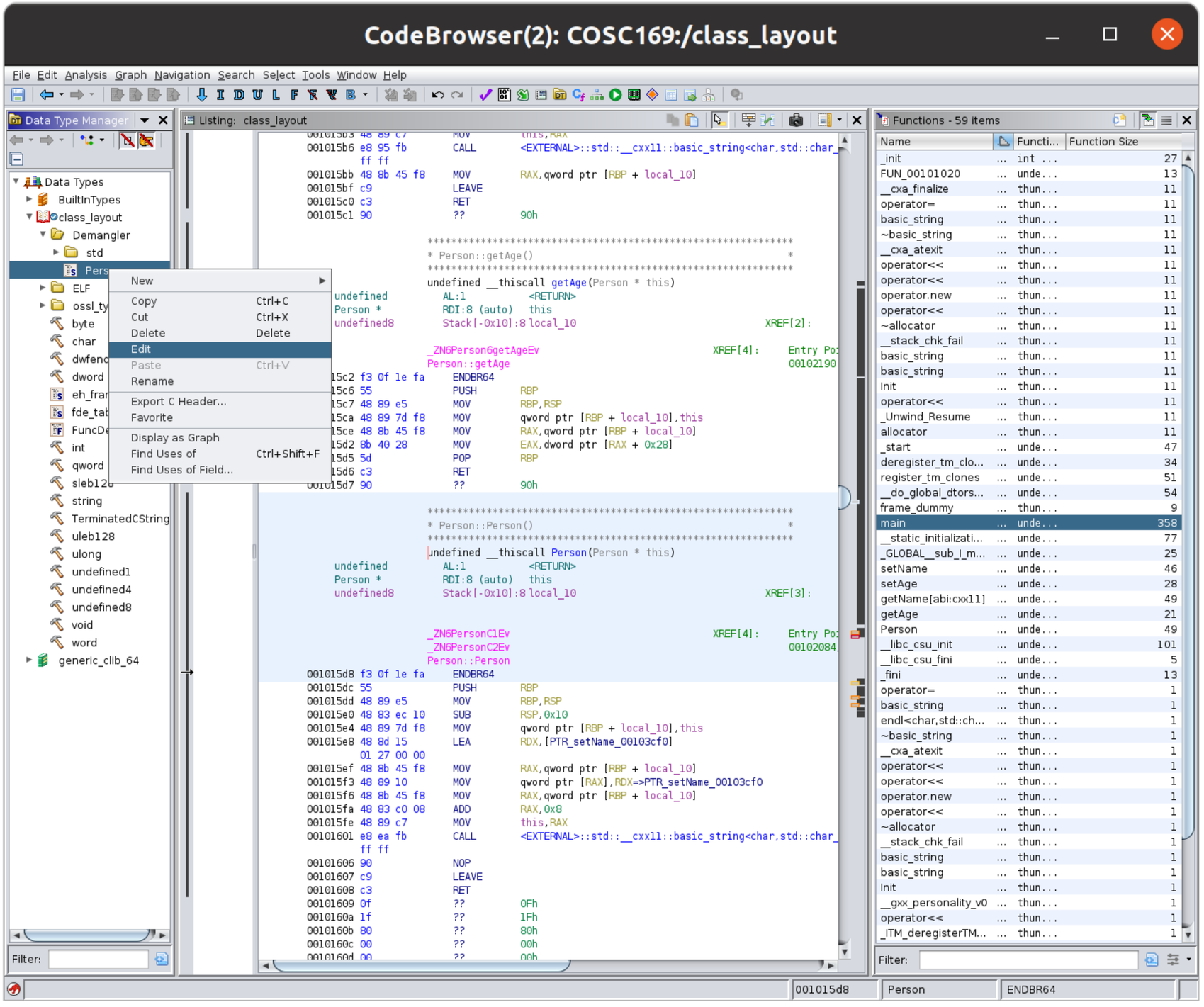
C++

Open the Data Type Manager



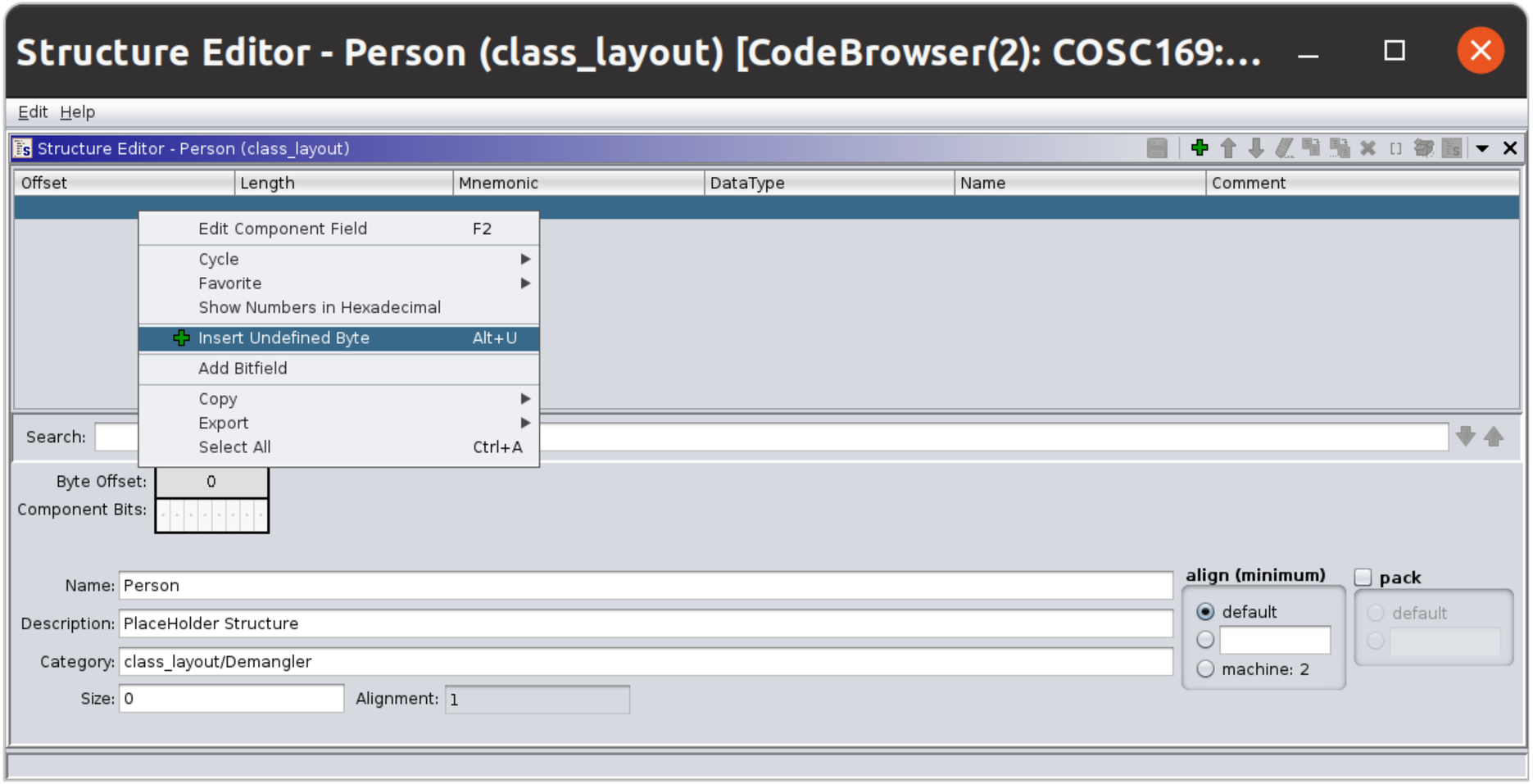
C++

Expand the Demangler



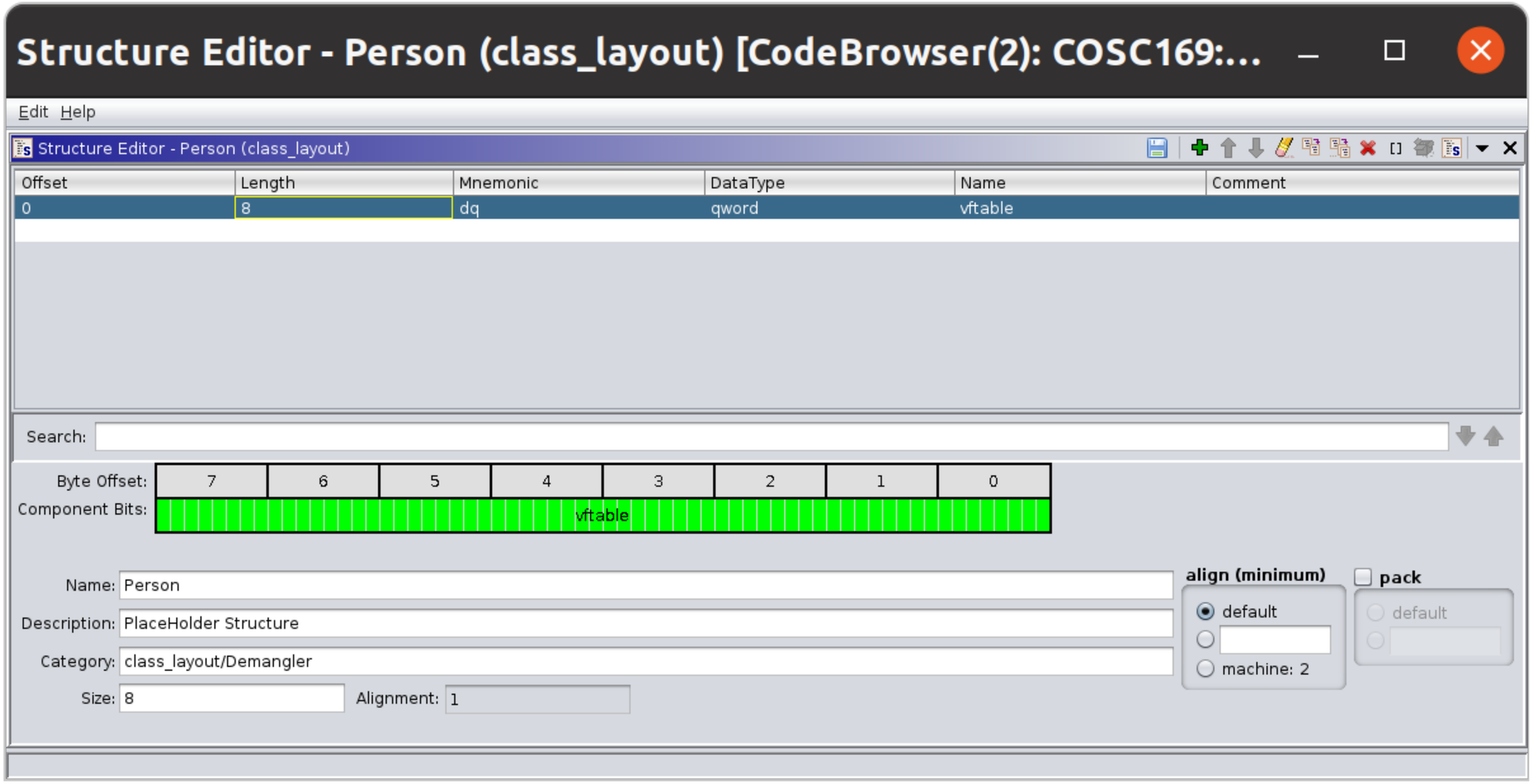
C++

Add an undefined byte (8x)



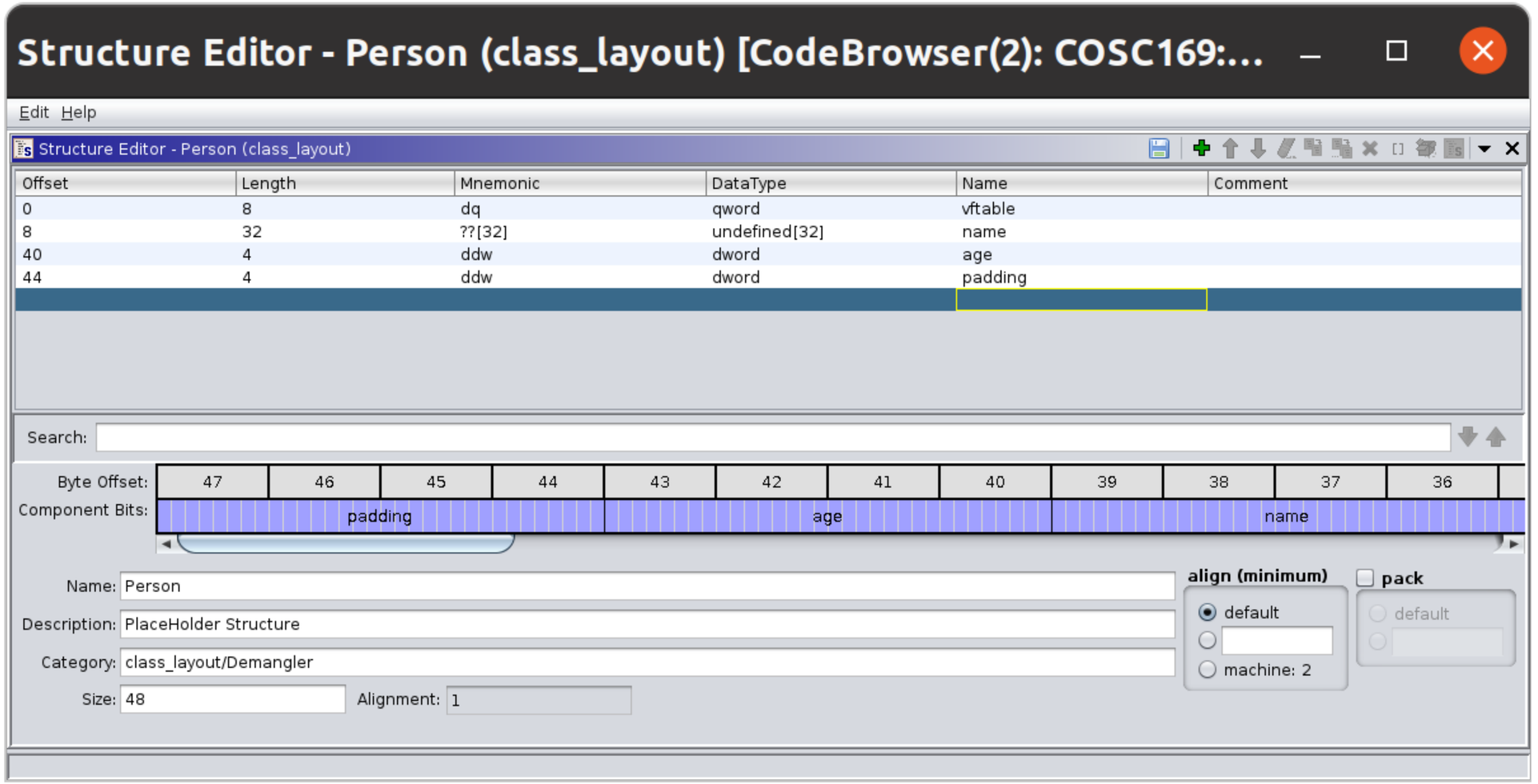
C++

Make it a qword and name it. The hot key is probably 'b'



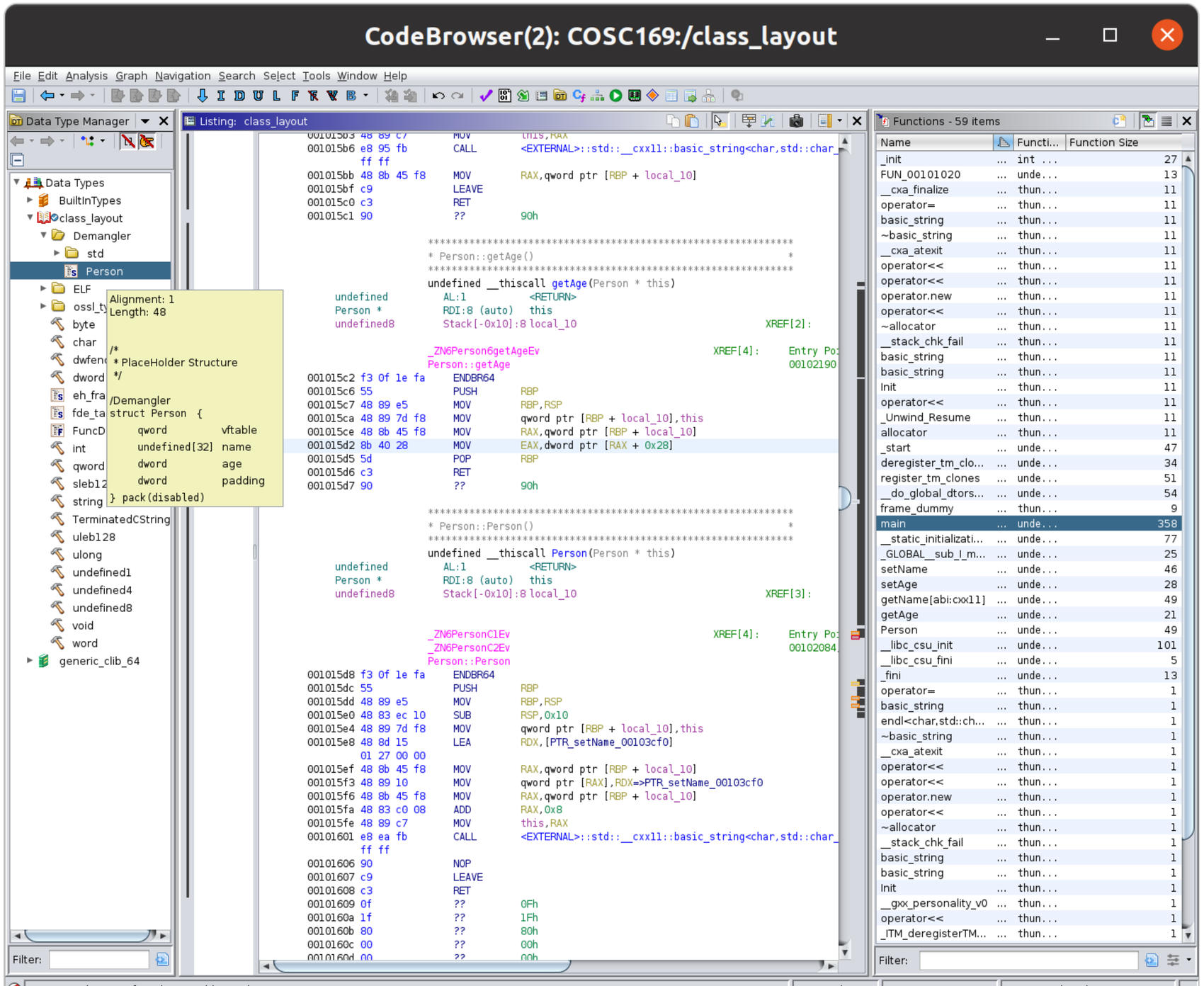
C++

Add the remaining two fields then click the save icon



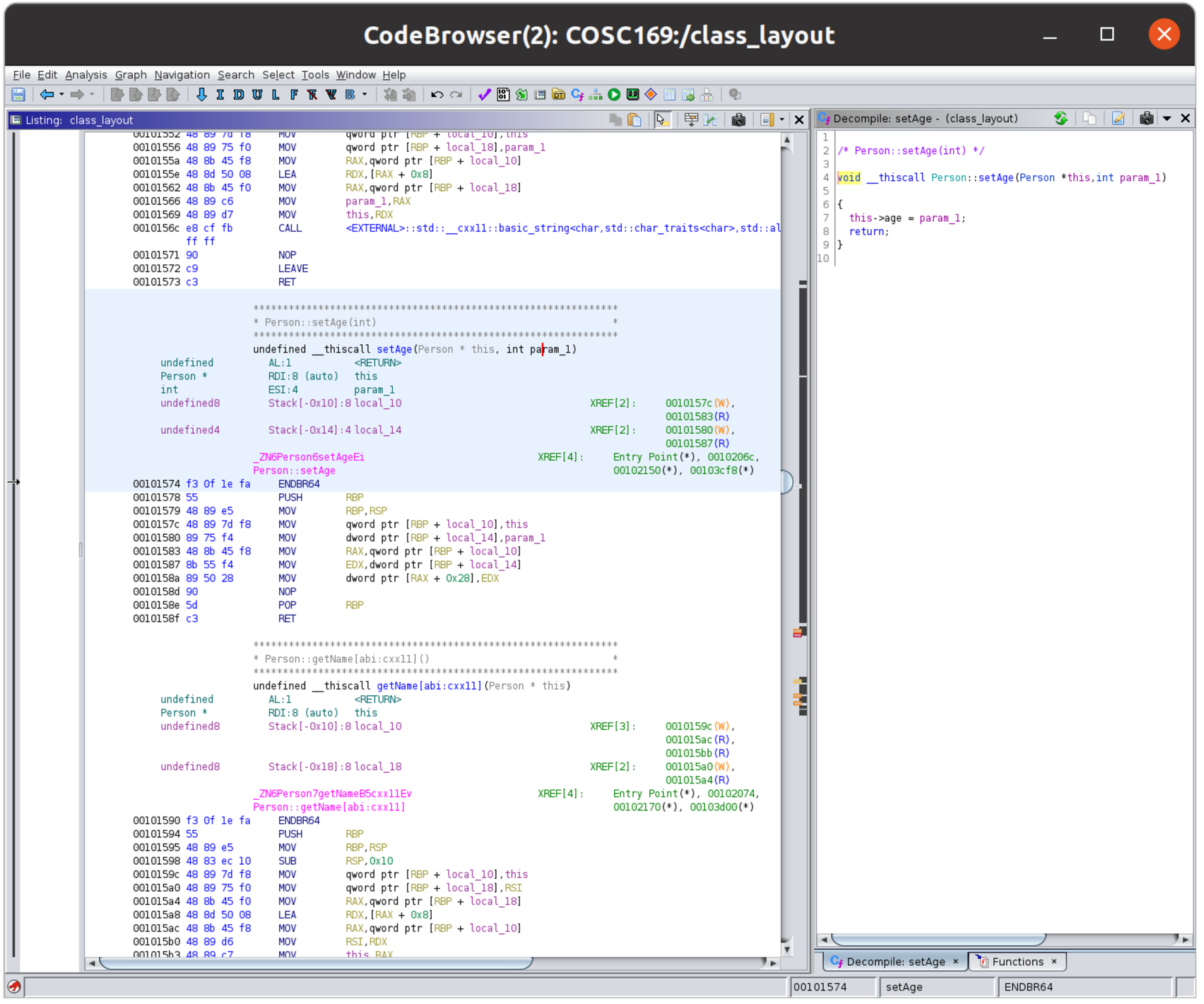
C++

The Person struct will now be updated



C++

Results? Open the Decompilation of setAge



C++

Inheritance

- What if a class has a parent class from which it inherits?

We stopped here in class

C++
Inheritance

john@caesar: ~/recourse/week03

```
class Building {
  private:
    int height;
    int width;
    std::string address;

  public:
    virtual void setHeight( int height ) { this->height = height; }
    virtual void setWidth( int width ) { this->width = width; }
    virtual void setAddress( std::string address ) { this->address = address; }

    virtual int getHeight( void ) { return this->height; }
    virtual int getWidth( void ) { return this->width; }
    virtual std::string getAddress( void ) { return this->address; }
};
~
~
~
~
~
~
~
~
~
"building.hpp" 15L, 500C                                    6,0-1         All
```

C++
Inheritance

john@caesar: ~/recourse/week03

```
#include "building.hpp"

class House: public Building {
  private:
    int rooms;
    int baths;

  public:
    virtual void setRooms( int rooms ) { this->rooms = rooms; }
    virtual void setBaths( int baths ) { this->baths = baths; }

    virtual int getRooms( void ) { return this->rooms; }
    virtual int getBaths( void ) { return this->baths; }
};
~
~
~
~
~
~
~
~
~
~
"house.hpp" 14L, 354C                                 2,0-1         All
```

C++
Inheritance

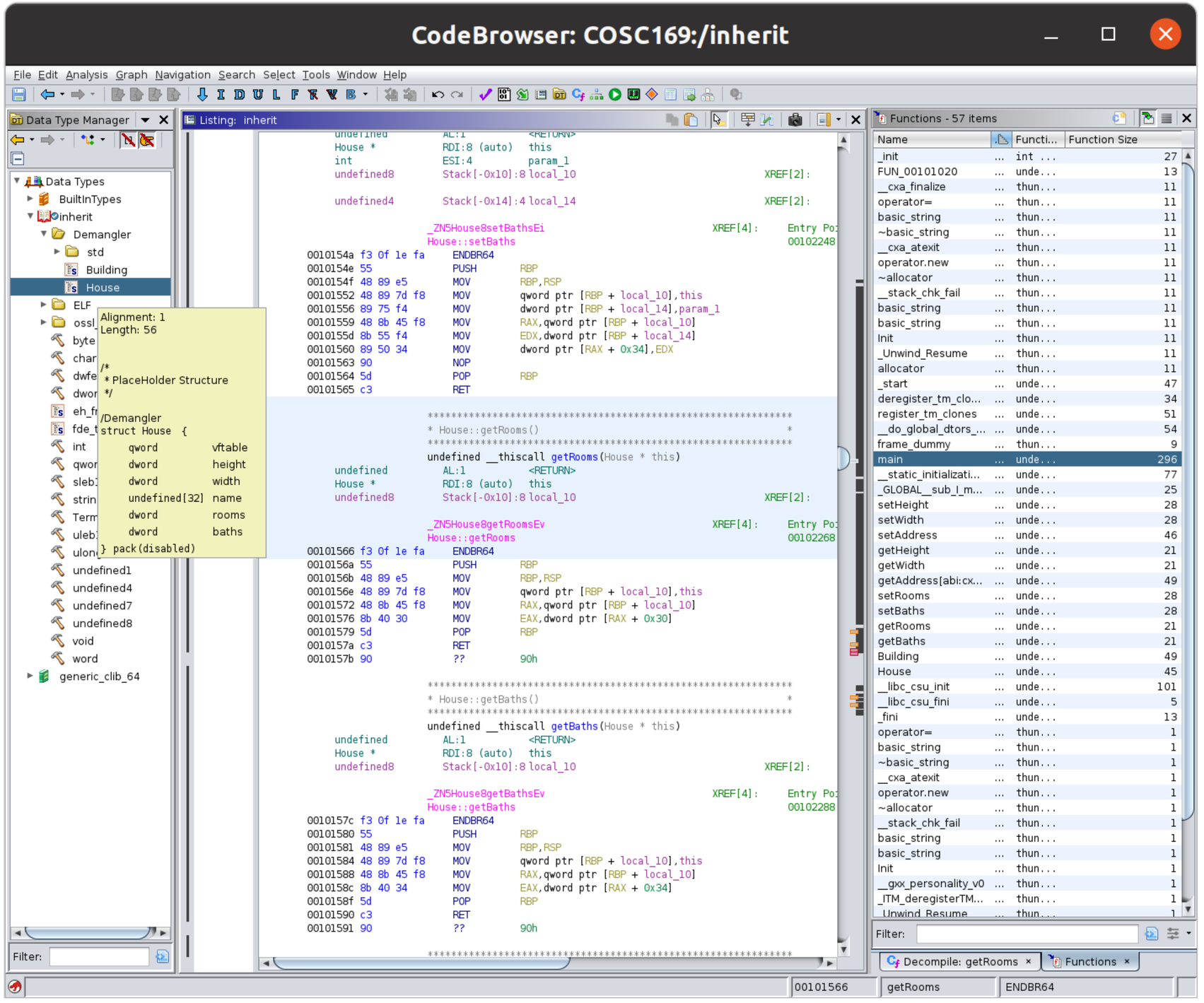
john@caesar: ~/recourse/week03

```
#include <iostream>

#include "house.hpp"

int main( int argc, char** argv)
{
  House *home = new House;

  home->setHeight(10);
  home->setWidth(10);
  home->setAddress("Nowhere");
  home->setRooms(4);
  home->setBaths(4);

  return 0;
}
~
~
~
~
~
~
~
"inherit.cpp" 16L, 239C                                    4,0-1         All
```

C++
Short Lab

- Open the *inherit* binary in Ghidra and using a combination of static and dynamic analysis work out the memory layout of the class

C++
Short Lab: Results



Homework

- Difficult crackme written in C++
- The difficulty with this one is likely to be in figuring out the algorithm
- Determine what input will result in printing SUCCESS
- A small write-up describing the algorithm