# CS 410 Project One

Project One

Elijah Hickey

CS-410

Southern New Hampshire University

Table of Contents

**main function3**

Assembly Code Block3

Explanation of Functionality4

**CheckUserPermissionAccess Function6**

Assembly Code Block6

Explanation of Functionality8

**DisplayInfo Function9**

Assembly Code Block9

Explanation of Functionality11

**ChangeCustomerChoice Function13**

Assembly Code Block13

Explanation of Functionality14

**Assembly of Project1.out 16**

NOTE: Disassembly of Project1.out can be found at the end of this document. The cpp file I created is labeled Project1EHv2.cpp and the object file is labeled Project1EHv2.o. I have gone through multiple iterations of the cpp files, and am confident that this is extremely close in both functionality and design to the original files.

### *main function*

**Assembly Code Block - main**

0x0000000000000000 <+0>: push %rbp

0x0000000000000001 <+1>: mov %rsp,%rbp

0x0000000000000004 <+4>: lea 0x0(%rip),%rsi # 0xb <main+11>

0x000000000000000b <+11>: lea 0x0(%rip),%rdi # 0x12 <main+18>

0x0000000000000012 <+18>: callq 0x17 <main+23>

0x0000000000000017 <+23>: callq 0x1c <main+28>

0x000000000000001c <+28>: mov %eax,0x0(%rip) # 0x22 <main+34>

0x0000000000000022 <+34>: mov 0x0(%rip),%eax # 0x28 <main+40>

0x0000000000000028 <+40>: cmp $0x1,%eax

0x000000000000002b <+43>: je 0x40 <main+64>

0x000000000000002d <+45>: lea 0x0(%rip),%rsi # 0x34 <main+52>

0x0000000000000034 <+52>: lea 0x0(%rip),%rdi # 0x3b <main+59>

0x000000000000003b <+59>: callq 0x40 <main+64>

0x0000000000000040 <+64>: mov 0x0(%rip),%eax # 0x46 <main+70>

0x0000000000000046 <+70>: cmp $0x1,%eax

0x0000000000000049 <+73>: je 0x4d <main+77>

0x000000000000004b <+75>: jmp 0x17 <main+23>

0x000000000000004d <+77>: lea 0x0(%rip),%rsi # 0x54 <main+84>

0x0000000000000054 <+84>: lea 0x0(%rip),%rdi # 0x5b <main+91>

0x000000000000005b <+91>: callq 0x60 <main+96>

0x0000000000000060 <+96>: lea 0x0(%rip),%rsi # 0x67 <main+103>

0x0000000000000067 <+103>: lea 0x0(%rip),%rdi # 0x6e <main+110>

0x000000000000006e <+110>: callq 0x73 <main+115>

0x0000000000000073 <+115>: lea 0x0(%rip),%rsi # 0x7a <main+122>

0x000000000000007a <+122>: lea 0x0(%rip),%rdi # 0x81 <main+129>

0x0000000000000081 <+129>: callq 0x86 <main+134>

0x0000000000000086 <+134>: lea 0x0(%rip),%rsi # 0x8d <main+141>

0x000000000000008d <+141>: lea 0x0(%rip),%rdi # 0x94 <main+148>

0x0000000000000094 <+148>: callq 0x99 <main+153>

0x0000000000000099 <+153>: lea 0x0(%rip),%rsi # 0xa0 <main+160>

0x00000000000000a0 <+160>: lea 0x0(%rip),%rdi # 0xa7 <main+167>

0x00000000000000a7 <+167>: callq 0xac <main+172>

0x00000000000000ac <+172>: lea 0x0(%rip),%rsi # 0xb3 <main+179>

0x00000000000000b3 <+179>: lea 0x0(%rip),%rdi # 0xba <main+186>

0x00000000000000ba <+186>: callq 0xbf <main+191>

0x00000000000000bf <+191>: mov %rax,%rdx

0x00000000000000c2 <+194>: mov 0x0(%rip),%eax # 0xc8 <main+200>

0x00000000000000c8 <+200>: mov %eax,%esi

0x00000000000000ca <+202>: mov %rdx,%rdi

0x00000000000000cd <+205>: callq 0xd2 <main+210>

0x00000000000000d2 <+210>: mov %rax,%rdx

0x00000000000000d5 <+213>: mov 0x0(%rip),%rax # 0xdc <main+220>

0x00000000000000dc <+220>: mov %rax,%rsi

0x00000000000000df <+223>: mov %rdx,%rdi

0x00000000000000e2 <+226>: callq 0xe7 <main+231>

0x00000000000000e7 <+231>: mov 0x0(%rip),%eax # 0xed <main+237>

0x00000000000000ed <+237>: cmp $0x1,%eax

0x00000000000000f0 <+240>: jne 0xf9 <main+249>

0x00000000000000f2 <+242>: callq 0xf7 <main+247>

0x00000000000000f7 <+247>: jmp 0x109 <main+265>

0x00000000000000f9 <+249>: mov 0x0(%rip),%eax # 0xff <main+255>

0x00000000000000ff <+255>: cmp $0x2,%eax

0x0000000000000102 <+258>: jne 0x109 <main+265>

0x0000000000000104 <+260>: callq 0x109 <main+265>

0x0000000000000109 <+265>: mov 0x0(%rip),%eax # 0x10f <main+271>

0x000000000000010f <+271>: cmp $0x3,%eax

0x0000000000000112 <+274>: je 0x119 <main+281>

0x0000000000000114 <+276>: jmpq 0x4d <main+77>

0x0000000000000119 <+281>: mov $0x0,%eax

0x000000000000011e <+286>: pop %rbp

0x000000000000011f <+287>: retq

**Explanation of Functionality – main**

The main function will start by loading the memory address of a string and passing it to the cout function call. This will output “Hello! Welcome to our Investment Company.” Then it will call the function CheckUserPermissionAccess (Main +23.) CheckUserPermissionAccess essentially prompts the user for their username and then their password, and will return the password to main. Back in main, the ‘password’ from CheckUserPermissionAccess is moved into %eax (a variable called answer in the original .o file.) This is then compared to 123 (I cannot figure out in assembly where 123 is grabbed from but I know this to be the case.) If the password is equal to 123, the user has been validated, and it will now move to the next portion of main. If it is not equal, it will load the memory address of the string “Invalid Password. Please try again” and passes it to cout to display the string. At this point, the users password is checked again against 123 (it will fail if it failed before, this is just done for the loops logic.) Since it will fail again, the code then jumps back to main + 23 which is the line that calls CheckUserPermissionAccess.

Lets assume that the user enters 123. It will jump over the lines that would display invalid password, and jump over the line that loops back to main +23. Now, the next parts are simply loading memory locations of strings via lea, the calling cout to display the strings. This is used to display the menu. The menu displays

“What would you like to do?

DISPLAY the client list (enter 1)

CHANGE a client’s choice (enter 2)

Exit the program.. (enter 3)

It will then call cin to store the users choice into a variable called choice (found thanks to the original .out file.) Then a cout statement prints “You chose” << choice “

This prints what the users choice was.

We are currently in another loop. This loop is as far as I can tell a while loop. It will Display the menu (seen above.) collect the users input and store it into choice. Display the users choice. If the user chooses 1, it will call the function DisplayInfo. If the user chooses 2, it will call the function ChangeCustomerChoice. If the user chooses 3, it will leave the loop and end the program. Any other input will cause the loop to repeat, displaying the menu, then prompting the user for their input again. I can tell this ‘choice’ variable is an int since a non-numeric enter causes an infinite loop of repeating menu displays. At loop starts at main+77, and the end is main+276 (this is the line that will jump back to +77 if the user does not enter 1 2 or 3.)

### *CheckUserPermissionAccess function*

**Assembly Code Block – CheckUserPermissionAccess**

0x0000000000000120 <+0>: push %rbp

0x0000000000000121 <+1>: mov %rsp,%rbp

0x0000000000000124 <+4>: push %rbx

0x0000000000000125 <+5>: sub $0x48,%rsp

0x0000000000000129 <+9>: mov %fs:0x28,%rax

0x0000000000000132 <+18>: mov %rax,-0x18(%rbp)

0x0000000000000136 <+22>: xor %eax,%eax

0x0000000000000138 <+24>: lea -0x45(%rbp),%rax

0x000000000000013c <+28>: mov %rax,%rdi

0x000000000000013f <+31>: callq 0x144 <\_Z25CheckUserPermissionAccessv+36>

0x0000000000000144 <+36>: lea -0x45(%rbp),%rdx

0x0000000000000148 <+40>: lea -0x40(%rbp),%rax

0x000000000000014c <+44>: lea 0x0(%rip),%rsi # 0x153 <\_Z25CheckUserPermissionAccessv+51>

0x0000000000000153 <+51>: mov %rax,%rdi

0x0000000000000156 <+54>: callq 0x15b <\_Z25CheckUserPermissionAccessv+59>

0x000000000000015b <+59>: lea -0x45(%rbp),%rax

0x000000000000015f <+63>: mov %rax,%rdi

0x0000000000000162 <+66>: callq 0x167 <\_Z25CheckUserPermissionAccessv+71>

0x0000000000000167 <+71>: movl $0x0,-0x44(%rbp)

0x000000000000016e <+78>: lea 0x0(%rip),%rsi # 0x175 <\_Z25CheckUserPermissionAccessv+85>

0x0000000000000175 <+85>: lea 0x0(%rip),%rdi # 0x17c <\_Z25CheckUserPermissionAccessv+92>

0x000000000000017c <+92>: callq 0x181 <\_Z25CheckUserPermissionAccessv+97>

0x0000000000000181 <+97>: lea 0x0(%rip),%rsi # 0x188 <\_Z25CheckUserPermissionAccessv+104>

0x0000000000000188 <+104>: lea 0x0(%rip),%rdi # 0x18f <\_Z25CheckUserPermissionAccessv+111>

0x000000000000018f <+111>: callq 0x194 <\_Z25CheckUserPermissionAccessv+116>

0x0000000000000194 <+116>: lea 0x0(%rip),%rsi # 0x19b <\_Z25CheckUserPermissionAccessv+123>

0x000000000000019b <+123>: lea 0x0(%rip),%rdi # 0x1a2 <\_Z25CheckUserPermissionAccessv+130>

0x00000000000001a2 <+130>: callq 0x1a7 <\_Z25CheckUserPermissionAccessv+135>

0x00000000000001a7 <+135>: lea -0x40(%rbp),%rax

0x00000000000001ab <+139>: mov %rax,%rsi

0x00000000000001ae <+142>: lea 0x0(%rip),%rdi # 0x1b5 <\_Z25CheckUserPermissionAccessv+149>

0x00000000000001b5 <+149>: callq 0x1ba <\_Z25CheckUserPermissionAccessv+154>

0x00000000000001ba <+154>: lea -0x40(%rbp),%rax

0x00000000000001be <+158>: lea 0x0(%rip),%rsi # 0x1c5 <\_Z25CheckUserPermissionAccessv+165>

0x00000000000001c5 <+165>: mov %rax,%rdi

0x00000000000001c8 <+168>: callq 0x1cd <\_Z25CheckUserPermissionAccessv+173>

0x00000000000001cd <+173>: mov %eax,-0x44(%rbp)

0x00000000000001d0 <+176>: cmpl $0x0,-0x44(%rbp)

0x00000000000001d4 <+180>: jne 0x1dd <\_Z25CheckUserPermissionAccessv+189>

0x00000000000001d6 <+182>: mov $0x1,%ebx

0x00000000000001db <+187>: jmp 0x1e2 <\_Z25CheckUserPermissionAccessv+194>

0x00000000000001dd <+189>: mov $0x2,%ebx

0x00000000000001e2 <+194>: lea -0x40(%rbp),%rax

0x00000000000001e6 <+198>: mov %rax,%rdi

0x00000000000001e9 <+201>: callq 0x1ee <\_Z25CheckUserPermissionAccessv+206>

0x00000000000001ee <+206>: mov %ebx,%eax

0x00000000000001f0 <+208>: mov -0x18(%rbp),%rcx

0x00000000000001f4 <+212>: xor %fs:0x28,%rcx

0x00000000000001fd <+221>: je 0x23a <\_Z25CheckUserPermissionAccessv+282>

0x00000000000001ff <+223>: jmp 0x235 <\_Z25CheckUserPermissionAccessv+277>

0x0000000000000201 <+225>: mov %rax,%rbx

0x0000000000000204 <+228>: lea -0x45(%rbp),%rax

0x0000000000000208 <+232>: mov %rax,%rdi

0x000000000000020b <+235>: callq 0x210 <\_Z25CheckUserPermissionAccessv+240>

0x0000000000000210 <+240>: mov %rbx,%rax

0x0000000000000213 <+243>: mov %rax,%rdi

0x0000000000000216 <+246>: callq 0x21b <\_Z25CheckUserPermissionAccessv+251>

0x000000000000021b <+251>: mov %rax,%rbx

0x000000000000021e <+254>: lea -0x40(%rbp),%rax

0x0000000000000222 <+258>: mov %rax,%rdi

0x0000000000000225 <+261>: callq 0x22a <\_Z25CheckUserPermissionAccessv+266>

0x000000000000022a <+266>: mov %rbx,%rax

0x000000000000022d <+269>: mov %rax,%rdi

0x0000000000000230 <+272>: callq 0x235 <\_Z25CheckUserPermissionAccessv+277>

0x0000000000000235 <+277>: callq 0x23a <\_Z25CheckUserPermissionAccessv+282>

0x000000000000023a <+282>: add $0x48,%rsp

0x000000000000023e <+286>: pop %rbx

0x000000000000023f <+287>: pop %rbp

0x0000000000000240 <+288>: retq

**Explanation of Functionality – CheckUserPermissionAccess**

This function works as follows. At +92 it will call its first cout. This will load the string address from memory and pass it to cout. It will display “Enter your username:”. It will then take the users input and store it into a variable called ‘username’ (this is found in the original .out file during disassembly.) This is stored in

-0x44 %rbp. Next it will load a string address from memory, pass it to cout, and display “Enter your password:.” Then it will grab the users input and store it in a variable called password. This is found at -0x40 %rbp. Then this function will clear out some of the registers, and finally it will use pop to return %rbp. This returns password to main, which is used for mains comparison. I cannot determine if this means that username is also passed to main, but I have seen no use for username throughout the program, so my cpp file only returns password.

### *DisplayInfo Function*

**Assembly Code Block – DisplayInfo**

0x0000000000000241 <+0>: push %rbp

0x0000000000000242 <+1>: mov %rsp,%rbp

0x0000000000000245 <+4>: lea 0x0(%rip),%rsi # 0x24c <\_Z11DisplayInfov+11>

0x000000000000024c <+11>: lea 0x0(%rip),%rdi # 0x253 <\_Z11DisplayInfov+18>

0x0000000000000253 <+18>: callq 0x258 <\_Z11DisplayInfov+23>

0x0000000000000258 <+23>: mov %rax,%rdx

0x000000000000025b <+26>: mov 0x0(%rip),%rax # 0x262 <\_Z11DisplayInfov+33>

0x0000000000000262 <+33>: mov %rax,%rsi

0x0000000000000265 <+36>: mov %rdx,%rdi

0x0000000000000268 <+39>: callq 0x26d <\_Z11DisplayInfov+44>

0x000000000000026d <+44>: lea 0x0(%rip),%rsi # 0x274 <\_Z11DisplayInfov+51>

0x0000000000000274 <+51>: lea 0x0(%rip),%rdi # 0x27b <\_Z11DisplayInfov+58>

0x000000000000027b <+58>: callq 0x280 <\_Z11DisplayInfov+63>

0x0000000000000280 <+63>: lea 0x0(%rip),%rsi # 0x287 <\_Z11DisplayInfov+70>

0x0000000000000287 <+70>: mov %rax,%rdi

0x000000000000028a <+73>: callq 0x28f <\_Z11DisplayInfov+78>

0x000000000000028f <+78>: lea 0x0(%rip),%rsi # 0x296 <\_Z11DisplayInfov+85>

0x0000000000000296 <+85>: mov %rax,%rdi

0x0000000000000299 <+88>: callq 0x29e <\_Z11DisplayInfov+93>

0x000000000000029e <+93>: mov %rax,%rdx

0x00000000000002a1 <+96>: mov 0x0(%rip),%eax # 0x2a7 <\_Z11DisplayInfov+102>

0x00000000000002a7 <+102>: mov %eax,%esi

0x00000000000002a9 <+104>: mov %rdx,%rdi

0x00000000000002ac <+107>: callq 0x2b1 <\_Z11DisplayInfov+112>

0x00000000000002b1 <+112>: mov %rax,%rdx

0x00000000000002b4 <+115>: mov 0x0(%rip),%rax # 0x2bb <\_Z11DisplayInfov+122>

0x00000000000002bb <+122>: mov %rax,%rsi

0x00000000000002be <+125>: mov %rdx,%rdi

0x00000000000002c1 <+128>: callq 0x2c6 <\_Z11DisplayInfov+133>

0x00000000000002c6 <+133>: lea 0x0(%rip),%rsi # 0x2cd <\_Z11DisplayInfov+140>

0x00000000000002cd <+140>: lea 0x0(%rip),%rdi # 0x2d4 <\_Z11DisplayInfov+147>

0x00000000000002d4 <+147>: callq 0x2d9 <\_Z11DisplayInfov+152>

0x00000000000002d9 <+152>: lea 0x0(%rip),%rsi # 0x2e0 <\_Z11DisplayInfov+159>

0x00000000000002e0 <+159>: mov %rax,%rdi

0x00000000000002e3 <+162>: callq 0x2e8 <\_Z11DisplayInfov+167>

0x00000000000002e8 <+167>: lea 0x0(%rip),%rsi # 0x2ef <\_Z11DisplayInfov+174>

0x00000000000002ef <+174>: mov %rax,%rdi

0x00000000000002f2 <+177>: callq 0x2f7 <\_Z11DisplayInfov+182>

0x00000000000002f7 <+182>: mov %rax,%rdx

0x00000000000002fa <+185>: mov 0x0(%rip),%eax # 0x300 <\_Z11DisplayInfov+191>

0x0000000000000300 <+191>: mov %eax,%esi

0x0000000000000302 <+193>: mov %rdx,%rdi

0x0000000000000305 <+196>: callq 0x30a <\_Z11DisplayInfov+201>

0x000000000000030a <+201>: mov %rax,%rdx

0x000000000000030d <+204>: mov 0x0(%rip),%rax # 0x314 <\_Z11DisplayInfov+211>

0x0000000000000314 <+211>: mov %rax,%rsi

0x0000000000000317 <+214>: mov %rdx,%rdi

0x000000000000031a <+217>: callq 0x31f <\_Z11DisplayInfov+222>

0x000000000000031f <+222>: lea 0x0(%rip),%rsi # 0x326 <\_Z11DisplayInfov+229>

0x0000000000000326 <+229>: lea 0x0(%rip),%rdi # 0x32d <\_Z11DisplayInfov+236>

0x000000000000032d <+236>: callq 0x332 <\_Z11DisplayInfov+241>

0x0000000000000332 <+241>: lea 0x0(%rip),%rsi # 0x339 <\_Z11DisplayInfov+248>

0x0000000000000339 <+248>: mov %rax,%rdi

0x000000000000033c <+251>: callq 0x341 <\_Z11DisplayInfov+256>

0x0000000000000341 <+256>: lea 0x0(%rip),%rsi # 0x348 <\_Z11DisplayInfov+263>

0x0000000000000348 <+263>: mov %rax,%rdi

0x000000000000034b <+266>: callq 0x350 <\_Z11DisplayInfov+271>

0x0000000000000350 <+271>: mov %rax,%rdx

0x0000000000000353 <+274>: mov 0x0(%rip),%eax # 0x359 <\_Z11DisplayInfov+280>

0x0000000000000359 <+280>: mov %eax,%esi

0x000000000000035b <+282>: mov %rdx,%rdi

0x000000000000035e <+285>: callq 0x363 <\_Z11DisplayInfov+290>

0x0000000000000363 <+290>: mov %rax,%rdx

0x0000000000000366 <+293>: mov 0x0(%rip),%rax # 0x36d <\_Z11DisplayInfov+300>

0x000000000000036d <+300>: mov %rax,%rsi

0x0000000000000370 <+303>: mov %rdx,%rdi

0x0000000000000373 <+306>: callq 0x378 <\_Z11DisplayInfov+311>

0x0000000000000378 <+311>: lea 0x0(%rip),%rsi # 0x37f <\_Z11DisplayInfov+318>

0x000000000000037f <+318>: lea 0x0(%rip),%rdi # 0x386 <\_Z11DisplayInfov+325>

0x0000000000000386 <+325>: callq 0x38b <\_Z11DisplayInfov+330>

0x000000000000038b <+330>: lea 0x0(%rip),%rsi # 0x392 <\_Z11DisplayInfov+337>

0x0000000000000392 <+337>: mov %rax,%rdi

0x0000000000000395 <+340>: callq 0x39a <\_Z11DisplayInfov+345>

0x000000000000039a <+345>: lea 0x0(%rip),%rsi # 0x3a1 <\_Z11DisplayInfov+352>

0x00000000000003a1 <+352>: mov %rax,%rdi

0x00000000000003a4 <+355>: callq 0x3a9 <\_Z11DisplayInfov+360>

0x00000000000003a9 <+360>: mov %rax,%rdx

0x00000000000003ac <+363>: mov 0x0(%rip),%eax # 0x3b2 <\_Z11DisplayInfov+369>

0x00000000000003b2 <+369>: mov %eax,%esi

0x00000000000003b4 <+371>: mov %rdx,%rdi

0x00000000000003b7 <+374>: callq 0x3bc <\_Z11DisplayInfov+379>

0x00000000000003bc <+379>: mov %rax,%rdx

0x00000000000003bf <+382>: mov 0x0(%rip),%rax # 0x3c6 <\_Z11DisplayInfov+389>

0x00000000000003c6 <+389>: mov %rax,%rsi

0x00000000000003c9 <+392>: mov %rdx,%rdi

0x00000000000003cc <+395>: callq 0x3d1 <\_Z11DisplayInfov+400>

0x00000000000003d1 <+400>: lea 0x0(%rip),%rsi # 0x3d8 <\_Z11DisplayInfov+407>

0x00000000000003d8 <+407>: lea 0x0(%rip),%rdi # 0x3df <\_Z11DisplayInfov+414>

0x00000000000003df <+414>: callq 0x3e4 <\_Z11DisplayInfov+419>

0x00000000000003e4 <+419>: lea 0x0(%rip),%rsi # 0x3eb <\_Z11DisplayInfov+426>

0x00000000000003eb <+426>: mov %rax,%rdi

0x00000000000003ee <+429>: callq 0x3f3 <\_Z11DisplayInfov+434>

0x00000000000003f3 <+434>: lea 0x0(%rip),%rsi # 0x3fa <\_Z11DisplayInfov+441>

0x00000000000003fa <+441>: mov %rax,%rdi

0x00000000000003fd <+444>: callq 0x402 <\_Z11DisplayInfov+449>

0x0000000000000402 <+449>: mov %rax,%rdx

0x0000000000000405 <+452>: mov 0x0(%rip),%eax # 0x40b <\_Z11DisplayInfov+458>

0x000000000000040b <+458>: mov %eax,%esi

0x000000000000040d <+460>: mov %rdx,%rdi

0x0000000000000410 <+463>: callq 0x415 <\_Z11DisplayInfov+468>

0x0000000000000415 <+468>: mov %rax,%rdx

0x0000000000000418 <+471>: mov 0x0(%rip),%rax # 0x41f <\_Z11DisplayInfov+478>

0x000000000000041f <+478>: mov %rax,%rsi

0x0000000000000422 <+481>: mov %rdx,%rdi

0x0000000000000425 <+484>: callq 0x42a <\_Z11DisplayInfov+489>

0x000000000000042a <+489>: nop

0x000000000000042b <+490>: pop %rbp

0x000000000000042c <+491>: retq

**Explanation of Functionality – DisplayInfo**

While this is one of the longer assemblies, it is probably the most straight forward. While At first I was unsure what data type was used to store client info. Based on the application, I thought that linked lists or nested arrays would work best (I originally designed my cpp file to use nested arrays of lists,) however I swapped to a manor that I felt best copied the original program. I believe the original program used global variables, and used two variables for each user, a name and a num (with postfix equal to the user’s assigned number.) I can see from the out file that for 1-5, there is name and num (name1, then num1 for example.)

This function is extremely straight forward, it is simply 6 cout statements. The first cout statement will load the string “Client’s Name Service Selected (1 = Brokerage, 2 = Retirement)” This has some formatting which I did my best to mimic.

Then, the rest of DisplayInfo is a series of 5 cout statements. I found in blesshex 1. 2. 3. 4. 5. Looking through this I don’t see anything that loops and iterates, so I am thinking that the first part of each cout statement simply grabs the string “1.” Or whatever number pertains to that client. They will also all grab the same string of

“ selected option “

For each client, there is one cout statement that grabs data from multiple locations. Based on the code, im guess the original cpp file looks something like this.

cout << “1. “ << name1 << “ selected option “ << num1 << endl;

cout << “2. “ << name2 << “ selected option “ << num2 << endl;

cout << “3. “ << name3 << “ selected option “ << num3 << endl;

cout << “4. “ << name4 << “ selected option “ << num4 << endl;

cout << “5. “ << name5 << “ selected option “ << num5 << endl;

After printing the 5 clients in such a manor, the function returns to main, where it will display the menu again and prompt the user for their choice.

### *ChangeCustomerChoice Function*

**Assembly Code Block – ChangeCustomerChoice**

0x000000000000042d <+0>: push %rbp

0x000000000000042e <+1>: mov %rsp,%rbp

0x0000000000000431 <+4>: lea 0x0(%rip),%rsi # 0x438 <\_Z20ChangeCustomerChoicev+11>

0x0000000000000438 <+11>: lea 0x0(%rip),%rdi # 0x43f <\_Z20ChangeCustomerChoicev+18>

0x000000000000043f <+18>: callq 0x444 <\_Z20ChangeCustomerChoicev+23>

0x0000000000000444 <+23>: lea 0x0(%rip),%rsi # 0x44b <\_Z20ChangeCustomerChoicev+30>

0x000000000000044b <+30>: lea 0x0(%rip),%rdi # 0x452 <\_Z20ChangeCustomerChoicev+37>

0x0000000000000452 <+37>: callq 0x457 <\_Z20ChangeCustomerChoicev+42>

0x0000000000000457 <+42>: lea 0x0(%rip),%rsi # 0x45e <\_Z20ChangeCustomerChoicev+49>

0x000000000000045e <+49>: lea 0x0(%rip),%rdi # 0x465 <\_Z20ChangeCustomerChoicev+56>

0x0000000000000465 <+56>: callq 0x46a <\_Z20ChangeCustomerChoicev+61>

0x000000000000046a <+61>: lea 0x0(%rip),%rsi # 0x471 <\_Z20ChangeCustomerChoicev+68>

0x0000000000000471 <+68>: lea 0x0(%rip),%rdi # 0x478 <\_Z20ChangeCustomerChoicev+75>

0x0000000000000478 <+75>: callq 0x47d <\_Z20ChangeCustomerChoicev+80>

0x000000000000047d <+80>: mov 0x0(%rip),%eax # 0x483 <\_Z20ChangeCustomerChoicev+86>

0x0000000000000483 <+86>: cmp $0x1,%eax

0x0000000000000486 <+89>: jne 0x496 <\_Z20ChangeCustomerChoicev+105>

0x0000000000000488 <+91>: mov 0x0(%rip),%eax # 0x48e <\_Z20ChangeCustomerChoicev+97>

0x000000000000048e <+97>: mov %eax,0x0(%rip) # 0x494 <\_Z20ChangeCustomerChoicev+103>

0x0000000000000494 <+103>: jmp 0x4f8 <\_Z20ChangeCustomerChoicev+203>

0x0000000000000496 <+105>: mov 0x0(%rip),%eax # 0x49c <\_Z20ChangeCustomerChoicev+111>

0x000000000000049c <+111>: cmp $0x2,%eax

0x000000000000049f <+114>: jne 0x4af <\_Z20ChangeCustomerChoicev+130>

0x00000000000004a1 <+116>: mov 0x0(%rip),%eax # 0x4a7 <\_Z20ChangeCustomerChoicev+122>

0x00000000000004a7 <+122>: mov %eax,0x0(%rip) # 0x4ad <\_Z20ChangeCustomerChoicev+128>

0x00000000000004ad <+128>: jmp 0x4f8 <\_Z20ChangeCustomerChoicev+203>

0x00000000000004af <+130>: mov 0x0(%rip),%eax # 0x4b5 <\_Z20ChangeCustomerChoicev+136>

0x00000000000004b5 <+136>: cmp $0x3,%eax

0x00000000000004b8 <+139>: jne 0x4c8 <\_Z20ChangeCustomerChoicev+155>

0x00000000000004ba <+141>: mov 0x0(%rip),%eax # 0x4c0 <\_Z20ChangeCustomerChoicev+147>

0x00000000000004c0 <+147>: mov %eax,0x0(%rip) # 0x4c6 <\_Z20ChangeCustomerChoicev+153>

0x00000000000004c6 <+153>: jmp 0x4f8 <\_Z20ChangeCustomerChoicev+203>

0x00000000000004c8 <+155>: mov 0x0(%rip),%eax # 0x4ce <\_Z20ChangeCustomerChoicev+161>

0x00000000000004ce <+161>: cmp $0x4,%eax

0x00000000000004d1 <+164>: jne 0x4e1 <\_Z20ChangeCustomerChoicev+180>

0x00000000000004d3 <+166>: mov 0x0(%rip),%eax # 0x4d9 <\_Z20ChangeCustomerChoicev+172>

0x00000000000004d9 <+172>: mov %eax,0x0(%rip) # 0x4df <\_Z20ChangeCustomerChoicev+178>

0x00000000000004df <+178>: jmp 0x4f8 <\_Z20ChangeCustomerChoicev+203>

0x00000000000004e1 <+180>: mov 0x0(%rip),%eax # 0x4e7 <\_Z20ChangeCustomerChoicev+186>

0x00000000000004e7 <+186>: cmp $0x5,%eax

0x00000000000004ea <+189>: jne 0x4f8 <\_Z20ChangeCustomerChoicev+203>

0x00000000000004ec <+191>: mov 0x0(%rip),%eax # 0x4f2 <\_Z20ChangeCustomerChoicev+197>

0x00000000000004f2 <+197>: mov %eax,0x0(%rip) # 0x4f8 <\_Z20ChangeCustomerChoicev+203>

0x00000000000004f8 <+203>: nop

0x00000000000004f9 <+204>: pop %rbp

0x00000000000004fa <+205>: retq

**Explanation of Functionality – ChangeCustomerChoice**

As discussed above, I just noticed that the original program was utilizing two seemingly unconnected variables for each client.

Looking at this assembly, it seems that is starts with a cout statement by loading the memory address of the string “Enter the number of the client that you wish to change.”

This number is stored into the variable changechoice via cin. Then it will load the memory address of the string “Enter the client’s new service choice (1 = Brokerage, 2 = Retirement)”. Following, it will store the users input into the variable newservice.

In assembly the loop works as follows: it will compare changechoice to 1. If not equal, it will compare to 2, and so forth (if it compares to 5 and is still not equal it simply leaves the function as this is the end of the function.)

If change choice equals one of the 1-5 numbers, it will then pass newservice into %eax, then pass %eax into the corrisponding num. Lets say the user enters 3 for the changechoice and 2 for newservice. It will jump over numbers 1 and 2 since they arent equal. Since changechoice == 3, it will pass newservice (2) into %eax, then mov %eax into num3. Since changechoice does not equal 4 or 5, it will jump over those and then leave the function since it has reached the end. This does no validation. If it is non-numeric, it causes infinite looping as seen throughout this program. If the number does not match a client (lets say the user enters 6) it will fail all 5 comparisons and leave the function. The newservice choice’s should only be 1 or 2 to indicate Brokerage or Retirement respectively, however there still lacks validation, so any other number can be passed and will be entered for the clients new service.

*Project1.out DISASSEMBLY*

Project1.out – main

0x0000000000000e4a <+0>: push %rbp

0x0000000000000e4b <+1>: mov %rsp,%rbp

0x0000000000000e4e <+4>: lea 0x5eb(%rip),%rsi # 0x1440

0x0000000000000e55 <+11>: lea 0x201244(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x0000000000000e5c <+18>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x0000000000000e61 <+23>: callq 0xf6a <\_Z25CheckUserPermissionAccessv>

0x0000000000000e66 <+28>: mov %eax,0x201494(%rip) # 0x202300 <answer>

0x0000000000000e6c <+34>: mov 0x20148e(%rip),%eax # 0x202300 <answer>

0x0000000000000e72 <+40>: cmp $0x1,%eax

0x0000000000000e75 <+43>: je 0xe8a <main+64>

0x0000000000000e77 <+45>: lea 0x5f2(%rip),%rsi # 0x1470

0x0000000000000e7e <+52>: lea 0x20121b(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x0000000000000e85 <+59>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x0000000000000e8a <+64>: mov 0x201470(%rip),%eax # 0x202300 <answer>

0x0000000000000e90 <+70>: cmp $0x1,%eax

0x0000000000000e93 <+73>: je 0xe97 <main+77>

0x0000000000000e95 <+75>: jmp 0xe61 <main+23>

0x0000000000000e97 <+77>: lea 0x5f6(%rip),%rsi # 0x1494

0x0000000000000e9e <+84>: lea 0x2011fb(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x0000000000000ea5 <+91>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x0000000000000eaa <+96>: lea 0x5ff(%rip),%rsi # 0x14b0

0x0000000000000eb1 <+103>: lea 0x2011e8(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x0000000000000eb8 <+110>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x0000000000000ebd <+115>: lea 0x614(%rip),%rsi # 0x14d8

0x0000000000000ec4 <+122>: lea 0x2011d5(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x0000000000000ecb <+129>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x0000000000000ed0 <+134>: lea 0x625(%rip),%rsi # 0x14fc

0x0000000000000ed7 <+141>: lea 0x2011c2(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x0000000000000ede <+148>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x0000000000000ee3 <+153>: lea 0x20140a(%rip),%rsi # 0x2022f4 <choice>

0x0000000000000eea <+160>: lea 0x2012cf(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4>

0x0000000000000ef1 <+167>: callq 0xc60 <\_ZNSirsERi@plt>

0x0000000000000ef6 <+172>: lea 0x61d(%rip),%rsi # 0x151a

0x0000000000000efd <+179>: lea 0x20119c(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x0000000000000f04 <+186>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x0000000000000f09 <+191>: mov %rax,%rdx

0x0000000000000f0c <+194>: mov 0x2013e2(%rip),%eax # 0x2022f4 <choice>

0x0000000000000f12 <+200>: mov %eax,%esi

0x0000000000000f14 <+202>: mov %rdx,%rdi

0x0000000000000f17 <+205>: callq 0xd00 <\_ZNSolsEi@plt>

0x0000000000000f1c <+210>: mov %rax,%rdx

0x0000000000000f1f <+213>: mov 0x2010aa(%rip),%rax # 0x201fd0

0x0000000000000f26 <+220>: mov %rax,%rsi

0x0000000000000f29 <+223>: mov %rdx,%rdi

0x0000000000000f2c <+226>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt>

0x0000000000000f31 <+231>: mov 0x2013bd(%rip),%eax # 0x2022f4 <choice>

0x0000000000000f37 <+237>: cmp $0x1,%eax

0x0000000000000f3a <+240>: jne 0xf43 <main+249>

0x0000000000000f3c <+242>: callq 0x108b <\_Z11DisplayInfov>

0x0000000000000f41 <+247>: jmp 0xf53 <main+265>

0x0000000000000f43 <+249>: mov 0x2013ab(%rip),%eax # 0x2022f4 <choice>

0x0000000000000f49 <+255>: cmp $0x2,%eax

0x0000000000000f4c <+258>: jne 0xf53 <main+265>

0x0000000000000f4e <+260>: callq 0x1277 <\_Z20ChangeCustomerChoicev>

0x0000000000000f53 <+265>: mov 0x20139b(%rip),%eax # 0x2022f4 <choice>

0x0000000000000f59 <+271>: cmp $0x3,%eax

0x0000000000000f5c <+274>: je 0xf63 <main+281>

0x0000000000000f5e <+276>: jmpq 0xe97 <main+77>

0x0000000000000f63 <+281>: mov $0x0,%eax

0x0000000000000f68 <+286>: pop %rbp

0x0000000000000f69 <+287>: retq

Project1.out - CheckUserPermissionAccess

0x0000000000000f6a <+0>: push %rbp

0x0000000000000f6b <+1>: mov %rsp,%rbp

0x0000000000000f6e <+4>: push %rbx

0x0000000000000f6f <+5>: sub $0x48,%rsp

0x0000000000000f73 <+9>: mov %fs:0x28,%rax

0x0000000000000f7c <+18>: mov %rax,-0x18(%rbp)

0x0000000000000f80 <+22>: xor %eax,%eax

0x0000000000000f82 <+24>: lea -0x45(%rbp),%rax

0x0000000000000f86 <+28>: mov %rax,%rdi

0x0000000000000f89 <+31>: callq 0xd20 <\_ZNSaIcEC1Ev@plt>

0x0000000000000f8e <+36>: lea -0x45(%rbp),%rdx

0x0000000000000f92 <+40>: lea -0x40(%rbp),%rax

0x0000000000000f96 <+44>: lea 0x588(%rip),%rsi # 0x1525

0x0000000000000f9d <+51>: mov %rax,%rdi

0x0000000000000fa0 <+54>: callq 0xce0 <\_ZNSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEEC1EPKcRKS3\_@plt>

0x0000000000000fa5 <+59>: lea -0x45(%rbp),%rax

0x0000000000000fa9 <+63>: mov %rax,%rdi

0x0000000000000fac <+66>: callq 0xcb0 <\_ZNSaIcED1Ev@plt>

0x0000000000000fb1 <+71>: movl $0x0,-0x44(%rbp)

0x0000000000000fb8 <+78>: lea 0x567(%rip),%rsi # 0x1526

0x0000000000000fbf <+85>: lea 0x2010da(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x0000000000000fc6 <+92>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x0000000000000fcb <+97>: lea 0x20130e(%rip),%rsi # 0x2022e0 <username>

0x0000000000000fd2 <+104>: lea 0x2011e7(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4>

0x0000000000000fd9 <+111>: callq 0xc40 <\_ZStrsIcSt11char\_traitsIcEERSt13basic\_istreamIT\_T0\_ES6\_PS3\_@plt>

0x0000000000000fde <+116>: lea 0x558(%rip),%rsi # 0x153d

0x0000000000000fe5 <+123>: lea 0x2010b4(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x0000000000000fec <+130>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x0000000000000ff1 <+135>: lea -0x40(%rbp),%rax

0x0000000000000ff5 <+139>: mov %rax,%rsi

0x0000000000000ff8 <+142>: lea 0x2011c1(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4>

0x0000000000000fff <+149>: callq 0xcd0 <\_ZStrsIcSt11char\_traitsIcESaIcEERSt13basic\_istreamIT\_T0\_ES7\_RNSt7\_\_cxx1112basic\_stringIS4\_S5\_T1\_EE@plt>

0x0000000000001004 <+154>: lea -0x40(%rbp),%rax

0x0000000000001008 <+158>: lea 0x545(%rip),%rsi # 0x1554

0x000000000000100f <+165>: mov %rax,%rdi

0x0000000000001012 <+168>: callq 0xc50 <\_ZNKSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEE7compareEPKc@plt>

0x0000000000001017 <+173>: mov %eax,-0x44(%rbp)

0x000000000000101a <+176>: cmpl $0x0,-0x44(%rbp)

0x000000000000101e <+180>: jne 0x1027 <\_Z25CheckUserPermissionAccessv+189>

0x0000000000001020 <+182>: mov $0x1,%ebx

0x0000000000001025 <+187>: jmp 0x102c <\_Z25CheckUserPermissionAccessv+194>

0x0000000000001027 <+189>: mov $0x2,%ebx

0x000000000000102c <+194>: lea -0x40(%rbp),%rax

0x0000000000001030 <+198>: mov %rax,%rdi

0x0000000000001033 <+201>: callq 0xc70 <\_ZNSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEED1Ev@plt>

0x0000000000001038 <+206>: mov %ebx,%eax

0x000000000000103a <+208>: mov -0x18(%rbp),%rcx

0x000000000000103e <+212>: xor %fs:0x28,%rcx

0x0000000000001047 <+221>: je 0x1084 <\_Z25CheckUserPermissionAccessv+282>

0x0000000000001049 <+223>: jmp 0x107f <\_Z25CheckUserPermissionAccessv+277>

0x000000000000104b <+225>: mov %rax,%rbx

0x000000000000104e <+228>: lea -0x45(%rbp),%rax

0x0000000000001052 <+232>: mov %rax,%rdi

0x0000000000001055 <+235>: callq 0xcb0 <\_ZNSaIcED1Ev@plt>

0x000000000000105a <+240>: mov %rbx,%rax

0x000000000000105d <+243>: mov %rax,%rdi

0x0000000000001060 <+246>: callq 0xd10 <\_Unwind\_Resume@plt>

0x0000000000001065 <+251>: mov %rax,%rbx

0x0000000000001068 <+254>: lea -0x40(%rbp),%rax

0x000000000000106c <+258>: mov %rax,%rdi

0x000000000000106f <+261>: callq 0xc70 <\_ZNSt7\_\_cxx1112basic\_stringIcSt11char\_traitsIcESaIcEED1Ev@plt>

0x0000000000001074 <+266>: mov %rbx,%rax

0x0000000000001077 <+269>: mov %rax,%rdi

0x000000000000107a <+272>: callq 0xd10 <\_Unwind\_Resume@plt>

0x000000000000107f <+277>: callq 0xcc0 <\_\_stack\_chk\_fail@plt>

0x0000000000001084 <+282>: add $0x48,%rsp

0x0000000000001088 <+286>: pop %rbx

0x0000000000001089 <+287>: pop %rbp

0x000000000000108a <+288>: retq

Project1.out - DisplayInfo

0x000000000000108b <+0>: push %rbp

0x000000000000108c <+1>: mov %rsp,%rbp

0x000000000000108f <+4>: lea 0x4c2(%rip),%rsi # 0x1558

0x0000000000001096 <+11>: lea 0x201003(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x000000000000109d <+18>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x00000000000010a2 <+23>: mov %rax,%rdx

0x00000000000010a5 <+26>: mov 0x200f24(%rip),%rax # 0x201fd0

0x00000000000010ac <+33>: mov %rax,%rsi

0x00000000000010af <+36>: mov %rdx,%rdi

0x00000000000010b2 <+39>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt>

0x00000000000010b7 <+44>: lea 0x4de(%rip),%rsi # 0x159c

0x00000000000010be <+51>: lea 0x200fdb(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x00000000000010c5 <+58>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x00000000000010ca <+63>: lea 0x200f3f(%rip),%rsi # 0x202010 <name1>

0x00000000000010d1 <+70>: mov %rax,%rdi

0x00000000000010d4 <+73>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x00000000000010d9 <+78>: lea 0x4c0(%rip),%rsi # 0x15a0

0x00000000000010e0 <+85>: mov %rax,%rdi

0x00000000000010e3 <+88>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x00000000000010e8 <+93>: mov %rax,%rdx

0x00000000000010eb <+96>: mov 0x200f6f(%rip),%eax # 0x202060 <num1>

0x00000000000010f1 <+102>: mov %eax,%esi

0x00000000000010f3 <+104>: mov %rdx,%rdi

0x00000000000010f6 <+107>: callq 0xd00 <\_ZNSolsEi@plt>

0x00000000000010fb <+112>: mov %rax,%rdx

0x00000000000010fe <+115>: mov 0x200ecb(%rip),%rax # 0x201fd0

0x0000000000001105 <+122>: mov %rax,%rsi

0x0000000000001108 <+125>: mov %rdx,%rdi

0x000000000000110b <+128>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt>

0x0000000000001110 <+133>: lea 0x49b(%rip),%rsi # 0x15b2

0x0000000000001117 <+140>: lea 0x200f82(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x000000000000111e <+147>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x0000000000001123 <+152>: lea 0x200ef6(%rip),%rsi # 0x202020 <name2>

0x000000000000112a <+159>: mov %rax,%rdi

0x000000000000112d <+162>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x0000000000001132 <+167>: lea 0x467(%rip),%rsi # 0x15a0

0x0000000000001139 <+174>: mov %rax,%rdi

0x000000000000113c <+177>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x0000000000001141 <+182>: mov %rax,%rdx

0x0000000000001144 <+185>: mov 0x200f1a(%rip),%eax # 0x202064 <num2>

0x000000000000114a <+191>: mov %eax,%esi

0x000000000000114c <+193>: mov %rdx,%rdi

0x000000000000114f <+196>: callq 0xd00 <\_ZNSolsEi@plt>

0x0000000000001154 <+201>: mov %rax,%rdx

0x0000000000001157 <+204>: mov 0x200e72(%rip),%rax # 0x201fd0

0x000000000000115e <+211>: mov %rax,%rsi

0x0000000000001161 <+214>: mov %rdx,%rdi

0x0000000000001164 <+217>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt>

0x0000000000001169 <+222>: lea 0x446(%rip),%rsi # 0x15b6

0x0000000000001170 <+229>: lea 0x200f29(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x0000000000001177 <+236>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x000000000000117c <+241>: lea 0x200ead(%rip),%rsi # 0x202030 <name3>

0x0000000000001183 <+248>: mov %rax,%rdi

0x0000000000001186 <+251>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x000000000000118b <+256>: lea 0x40e(%rip),%rsi # 0x15a0

0x0000000000001192 <+263>: mov %rax,%rdi

0x0000000000001195 <+266>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x000000000000119a <+271>: mov %rax,%rdx

0x000000000000119d <+274>: mov 0x200ec5(%rip),%eax # 0x202068 <num3>

0x00000000000011a3 <+280>: mov %eax,%esi

0x00000000000011a5 <+282>: mov %rdx,%rdi

0x00000000000011a8 <+285>: callq 0xd00 <\_ZNSolsEi@plt>

0x00000000000011ad <+290>: mov %rax,%rdx

0x00000000000011b0 <+293>: mov 0x200e19(%rip),%rax # 0x201fd0

0x00000000000011b7 <+300>: mov %rax,%rsi

0x00000000000011ba <+303>: mov %rdx,%rdi

0x00000000000011bd <+306>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt>

0x00000000000011c2 <+311>: lea 0x3f1(%rip),%rsi # 0x15ba

0x00000000000011c9 <+318>: lea 0x200ed0(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x00000000000011d0 <+325>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x00000000000011d5 <+330>: lea 0x200e64(%rip),%rsi # 0x202040 <name4>

0x00000000000011dc <+337>: mov %rax,%rdi

0x00000000000011df <+340>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x00000000000011e4 <+345>: lea 0x3b5(%rip),%rsi # 0x15a0

0x00000000000011eb <+352>: mov %rax,%rdi

0x00000000000011ee <+355>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x00000000000011f3 <+360>: mov %rax,%rdx

0x00000000000011f6 <+363>: mov 0x200e70(%rip),%eax # 0x20206c <num4>

0x00000000000011fc <+369>: mov %eax,%esi

0x00000000000011fe <+371>: mov %rdx,%rdi

0x0000000000001201 <+374>: callq 0xd00 <\_ZNSolsEi@plt>

0x0000000000001206 <+379>: mov %rax,%rdx

0x0000000000001209 <+382>: mov 0x200dc0(%rip),%rax # 0x201fd0

0x0000000000001210 <+389>: mov %rax,%rsi

0x0000000000001213 <+392>: mov %rdx,%rdi

0x0000000000001216 <+395>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt>

0x000000000000121b <+400>: lea 0x39c(%rip),%rsi # 0x15be

0x0000000000001222 <+407>: lea 0x200e77(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x0000000000001229 <+414>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x000000000000122e <+419>: lea 0x200e1b(%rip),%rsi # 0x202050 <name5>

0x0000000000001235 <+426>: mov %rax,%rdi

0x0000000000001238 <+429>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x000000000000123d <+434>: lea 0x35c(%rip),%rsi # 0x15a0

0x0000000000001244 <+441>: mov %rax,%rdi

0x0000000000001247 <+444>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x000000000000124c <+449>: mov %rax,%rdx

0x000000000000124f <+452>: mov 0x200e1b(%rip),%eax # 0x202070 <num5>

0x0000000000001255 <+458>: mov %eax,%esi

0x0000000000001257 <+460>: mov %rdx,%rdi

0x000000000000125a <+463>: callq 0xd00 <\_ZNSolsEi@plt>

0x000000000000125f <+468>: mov %rax,%rdx

0x0000000000001262 <+471>: mov 0x200d67(%rip),%rax # 0x201fd0

0x0000000000001269 <+478>: mov %rax,%rsi

0x000000000000126c <+481>: mov %rdx,%rdi

0x000000000000126f <+484>: callq 0xca0 <\_ZNSolsEPFRSoS\_E@plt>

0x0000000000001274 <+489>: nop

0x0000000000001275 <+490>: pop %rbp

0x0000000000001276 <+491>: retq

Project1.out – ChangeCustomerChoice()

0x0000000000001277 <+0>: push %rbp

0x0000000000001278 <+1>: mov %rsp,%rbp

0x000000000000127b <+4>: lea 0x346(%rip),%rsi # 0x15c8

0x0000000000001282 <+11>: lea 0x200e17(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x0000000000001289 <+18>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x000000000000128e <+23>: lea 0x201063(%rip),%rsi # 0x2022f8 <changechoice>

0x0000000000001295 <+30>: lea 0x200f24(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4>

0x000000000000129c <+37>: callq 0xc60 <\_ZNSirsERi@plt>

0x00000000000012a1 <+42>: lea 0x358(%rip),%rsi # 0x1600

0x00000000000012a8 <+49>: lea 0x200df1(%rip),%rdi # 0x2020a0 <\_ZSt4cout@@GLIBCXX\_3.4>

0x00000000000012af <+56>: callq 0xc90 <\_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@plt>

0x00000000000012b4 <+61>: lea 0x201041(%rip),%rsi # 0x2022fc <newservice>

0x00000000000012bb <+68>: lea 0x200efe(%rip),%rdi # 0x2021c0 <\_ZSt3cin@@GLIBCXX\_3.4>

0x00000000000012c2 <+75>: callq 0xc60 <\_ZNSirsERi@plt>

0x00000000000012c7 <+80>: mov 0x20102b(%rip),%eax # 0x2022f8 <changechoice>

0x00000000000012cd <+86>: cmp $0x1,%eax

0x00000000000012d0 <+89>: jne 0x12e0 <\_Z20ChangeCustomerChoicev+105>

0x00000000000012d2 <+91>: mov 0x201024(%rip),%eax # 0x2022fc <newservice>

0x00000000000012d8 <+97>: mov %eax,0x200d82(%rip) # 0x202060 <num1>

0x00000000000012de <+103>: jmp 0x1342 <\_Z20ChangeCustomerChoicev+203>

0x00000000000012e0 <+105>: mov 0x201012(%rip),%eax # 0x2022f8 <changechoice>

0x00000000000012e6 <+111>: cmp $0x2,%eax

0x00000000000012e9 <+114>: jne 0x12f9 <\_Z20ChangeCustomerChoicev+130>

0x00000000000012eb <+116>: mov 0x20100b(%rip),%eax # 0x2022fc <newservice>

0x00000000000012f1 <+122>: mov %eax,0x200d6d(%rip) # 0x202064 <num2>

0x00000000000012f7 <+128>: jmp 0x1342 <\_Z20ChangeCustomerChoicev+203>

0x00000000000012f9 <+130>: mov 0x200ff9(%rip),%eax # 0x2022f8 <changechoice>

0x00000000000012ff <+136>: cmp $0x3,%eax

0x0000000000001302 <+139>: jne 0x1312 <\_Z20ChangeCustomerChoicev+155>

0x0000000000001304 <+141>: mov 0x200ff2(%rip),%eax # 0x2022fc <newservice>

0x000000000000130a <+147>: mov %eax,0x200d58(%rip) # 0x202068 <num3>

0x0000000000001310 <+153>: jmp 0x1342 <\_Z20ChangeCustomerChoicev+203>

0x0000000000001312 <+155>: mov 0x200fe0(%rip),%eax # 0x2022f8 <changechoice>

0x0000000000001318 <+161>: cmp $0x4,%eax

0x000000000000131b <+164>: jne 0x132b <\_Z20ChangeCustomerChoicev+180>

0x000000000000131d <+166>: mov 0x200fd9(%rip),%eax # 0x2022fc <newservice>

0x0000000000001323 <+172>: mov %eax,0x200d43(%rip) # 0x20206c <num4>

0x0000000000001329 <+178>: jmp 0x1342 <\_Z20ChangeCustomerChoicev+203>

0x000000000000132b <+180>: mov 0x200fc7(%rip),%eax # 0x2022f8 <changechoice>

0x0000000000001331 <+186>: cmp $0x5,%eax

0x0000000000001334 <+189>: jne 0x1342 <\_Z20ChangeCustomerChoicev+203>

0x0000000000001336 <+191>: mov 0x200fc0(%rip),%eax # 0x2022fc <newservice>

0x000000000000133c <+197>: mov %eax,0x200d2e(%rip) # 0x202070 <num5>

0x0000000000001342 <+203>: nop

0x0000000000001343 <+204>: pop %rbp

0x0000000000001344 <+205>: retq