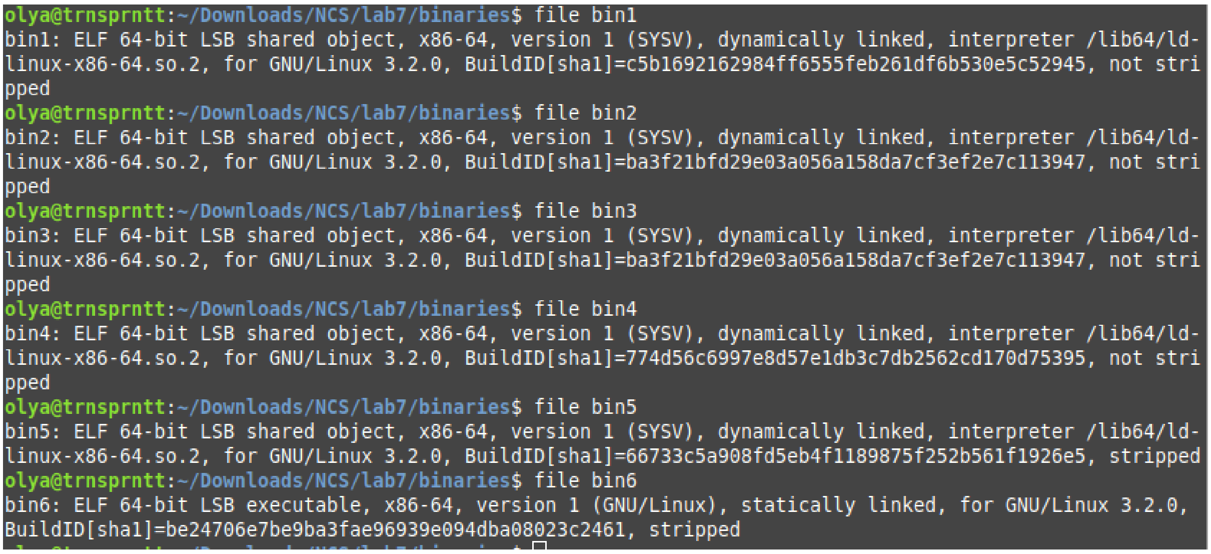
NCS lab 7
Reversing

Olga Chernukhina
BS-18-SB-01

Task 2 - Theory

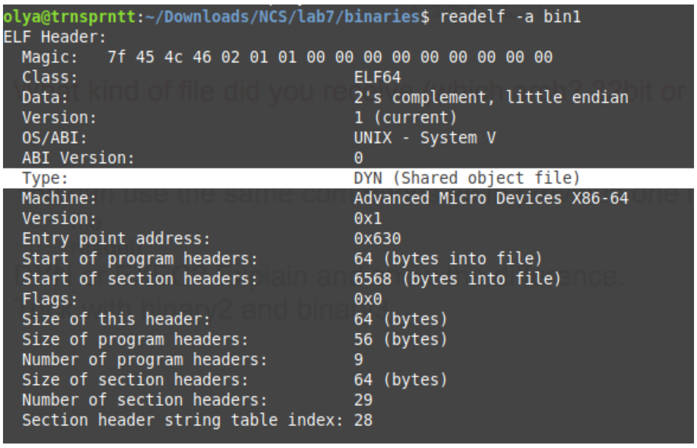
1. What kind of file did you receive (which arch? 32bit or 64bit?)?

Information corresponding each file is presented on the screen

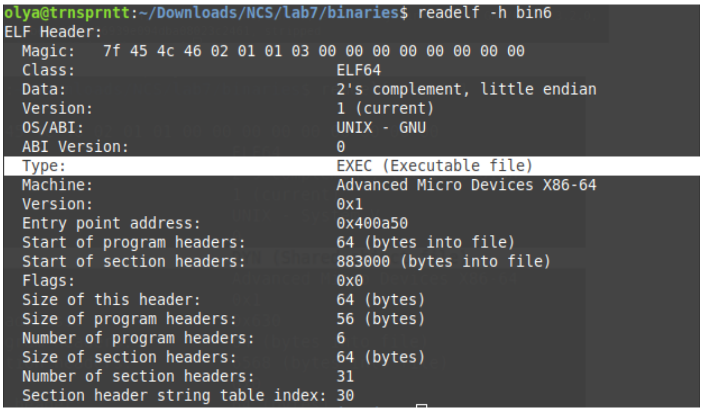


```
olya@trnsprntt:~/Downloads/NCS/lab7/binaries$ file bin1
bin1: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-
linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=c5b1692162984ff6555feb261df6b530e5c52945, not stri
pped
olya@trnsprntt:~/Downloads/NCS/lab7/binaries$ file bin2
bin2: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-
linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=ba3f21bfd29e03a056a158da7cf3ef2e7c113947, not stri
pped
olya@trnsprntt:~/Downloads/NCS/lab7/binaries$ file bin3
bin3: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-
linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=ba3f21bfd29e03a056a158da7cf3ef2e7c113947, not stri
pped
olya@trnsprntt:~/Downloads/NCS/lab7/binaries$ file bin4
bin4: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-
linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=774d56c6997e8d57e1db3c7db2562cd170d75395, not stri
pped
olya@trnsprntt:~/Downloads/NCS/lab7/binaries$ file bin5
bin5: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-
linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=66733c5a908fd5eb4f1189875f252b561f1926e5, stripped
olya@trnsprntt:~/Downloads/NCS/lab7/binaries$ file bin6
bin6: ELF 64-bit LSB executable, x86-64, version 1 (GNU/Linux), statically linked, for GNU/Linux 3.2.0,
BuildID[sha1]=be24706e7be9ba3fae96939e094dba08023c2461, stripped
```

If GCC is configured to build `-pie` (position independent executable) then the output is `DYN`, otherwise, if it is linked using `gcc -no-pie EXEC` type would be produced.



```
olya@trnsprntt:~/Downloads/NCS/lab7/binaries$ readelf -a bin1
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              DYN (Shared object file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x630
  Start of program headers:          64 (bytes into file)
  Start of section headers:          6568 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         9
  Size of section headers:           64 (bytes)
  Number of section headers:         29
  Section header string table index: 28
```



```
olya@trnsprntt:~/Downloads/NCS/lab7/binaries$ readelf -h bin6
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 03 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - GNU
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x400a50
  Start of program headers:          64 (bytes into file)
  Start of section headers:          883000 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         6
  Size of section headers:           64 (bytes)
  Number of section headers:         31
  Section header string table index: 30
```

Also, it is important to notice bin2 and bin3 have the same sha.

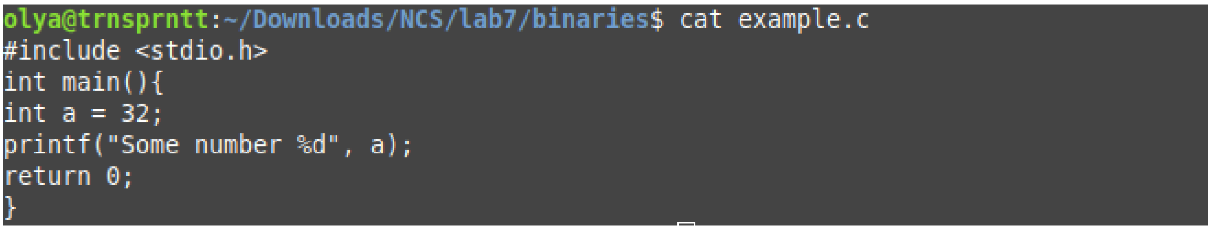
2. What do stripped binaries mean?

	stripped binaries	not stripped binaries
contain debugging information	no	yes
comparative size	small	large
compilation options	gcc -s	gcc -g

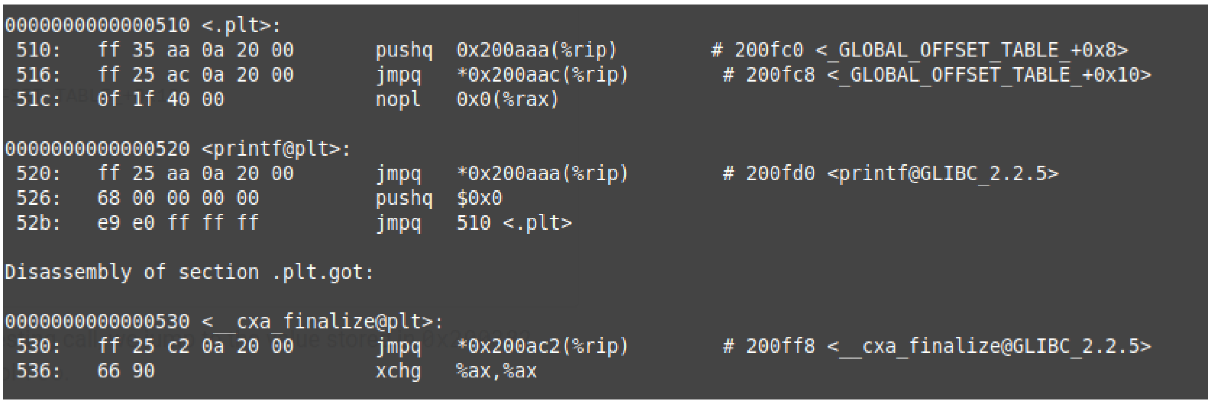
3. What are GOT and PLT?

GOT and PLT make code read-only but preserve the ability to access external libraries and function indirectly. PLT (Procedure Linkage Table) is used to call external functions - on each call a stub that contains the jump opcode to the corresponding record in GOT is created. GOT (global offset table) record contains the actual address of the external source written by the linker at runtime.

A simple example:



```
olya@trnsprntt:~/Downloads/NCS/lab7/binaries$ cat example.c
#include <stdio.h>
int main(){
int a = 32;
printf("Some number %d", a);
return 0;
}
```

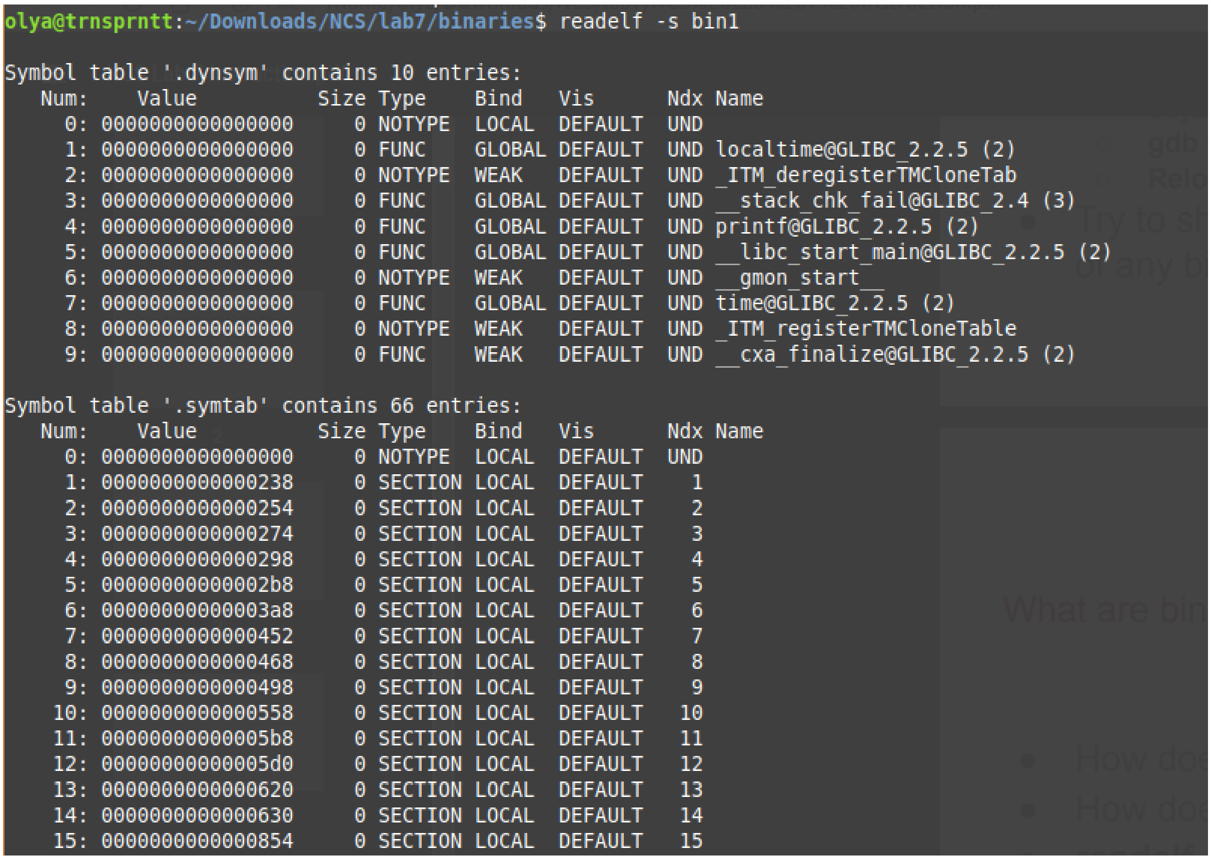


```
0000000000000510 <.plt>:
 510:   ff 35 aa 0a 20 00       pushq  0x200aaa(%rip)        # 200fc0 <_GLOBAL_OFFSET_TABLE_+0x8>
 516:   ff 25 ac 0a 20 00       jmpq   *0x200aac(%rip)        # 200fc8 <_GLOBAL_OFFSET_TABLE_+0x10>
 51c:   0f 1f 40 00             nopl   0x0(%rax)

0000000000000520 <printf@plt>:
 520:   ff 25 aa 0a 20 00       jmpq   *0x200aaa(%rip)        # 200fd0 <printf@GLIBC_2.2.5>
 526:   68 00 00 00 00          pushq  $0x0
 52b:   e9 e0 ff ff ff          jmpq   510 <.plt>

Disassembly of section .plt.got:

0000000000000530 <__cxa_finalize@plt>:
 530:   ff 25 c2 0a 20 00       jmpq   *0x200ac2(%rip)        # 200ff8 <__cxa_finalize@GLIBC_2.2.5>
 536:   66 90                   xchg   %ax,%ax
```

4. What are binary symbols in reverse engineering? How does it help?

Binary symbols are a part of a symbol table that stores the name of program components (e.g. functions, included files) and their corresponding location, size, type etc. Stripped files only have dynamic symbols left, while static symbols get wiped out from them. Symbolic tables help to understand the program structure and logic based on its function names and included source files’ names. In reverse engineering.

ex. - part of the bin1 symbolic table

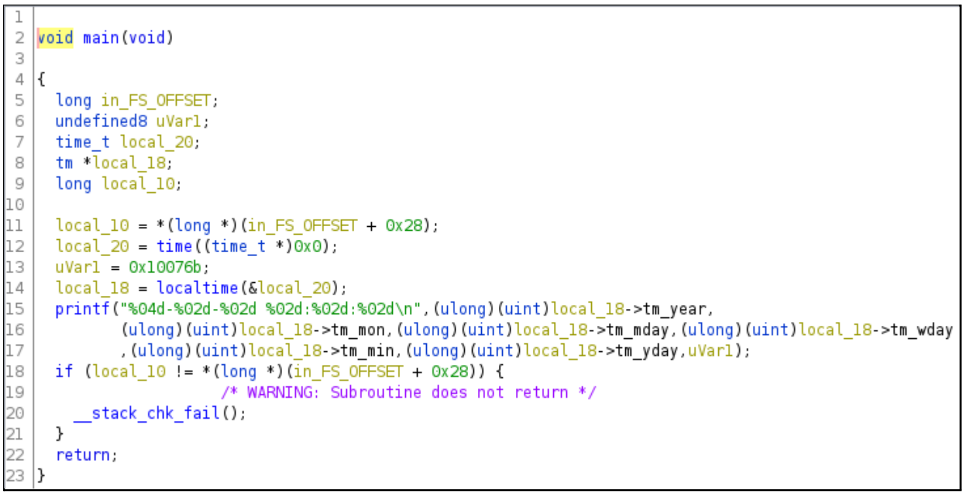


```
olya@trnsprntt:~/Downloads/NCS/lab7/binaries$ readelf -s bin1

Symbol table '.dynsym' contains 10 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND localtime@GLIBC_2.2.5 (2)
     2: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_deregisterTMCloneTab
     3: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __stack_chk_fail@GLIBC_2.4 (3)
     4: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND printf@GLIBC_2.2.5 (2)
     5: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND __libc_start_main@GLIBC_2.2.5 (2)
     6: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND __gmon_start__
     7: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND time@GLIBC_2.2.5 (2)
     8: 0000000000000000     0 NOTYPE  WEAK   DEFAULT  UND _ITM_registerTMCloneTable
     9: 0000000000000000     0 FUNC    WEAK   DEFAULT  UND __cxa_finalize@GLIBC_2.2.5 (2)

Symbol table '.symtab' contains 66 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000238     0 SECTION LOCAL  DEFAULT    1
     2: 0000000000000254     0 SECTION LOCAL  DEFAULT    2
     3: 0000000000000274     0 SECTION LOCAL  DEFAULT    3
     4: 0000000000000298     0 SECTION LOCAL  DEFAULT    4
     5: 00000000000002b8     0 SECTION LOCAL  DEFAULT    5
     6: 00000000000003a8     0 SECTION LOCAL  DEFAULT    6
     7: 0000000000000452     0 SECTION LOCAL  DEFAULT    7
     8: 0000000000000468     0 SECTION LOCAL  DEFAULT    8
     9: 0000000000000498     0 SECTION LOCAL  DEFAULT    9
    10: 0000000000000558     0 SECTION LOCAL  DEFAULT   10
    11: 00000000000005b8     0 SECTION LOCAL  DEFAULT   11
    12: 00000000000005d0     0 SECTION LOCAL  DEFAULT   12
    13: 0000000000000620     0 SECTION LOCAL  DEFAULT   13
    14: 0000000000000630     0 SECTION LOCAL  DEFAULT   14
    15: 0000000000000854     0 SECTION LOCAL  DEFAULT   15
```

Task 3 - Reversing

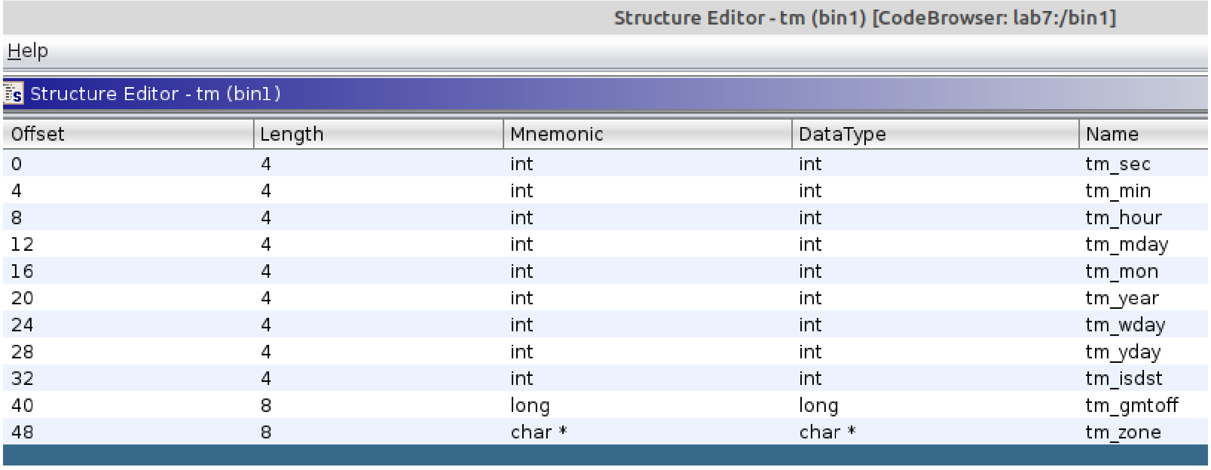
I am sorry in this part the report is not really detailed, since on every step I doubted if it would lead me to the solution and therefore didn’t make many screenshots

bin1

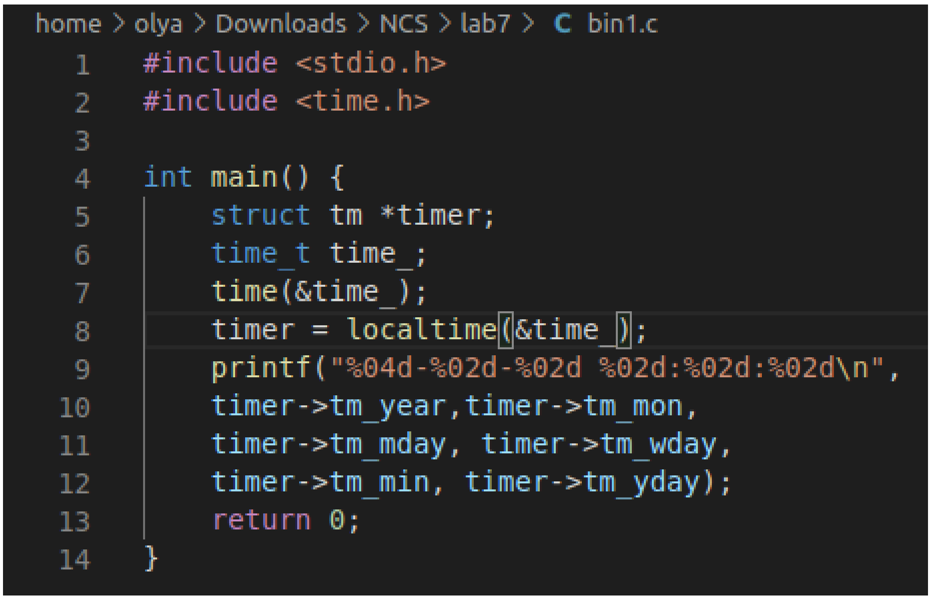


```

void main(void)

{
  long in_FS_OFFSET;
  undefined8 uVar1;
  time_t local_20;
  tm *local_18;
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  local_20 = time((time_t *)0x0);
  uVar1 = 0x10076b;
  local_18 = localtime(&local_20);
  printf("%04d-%02d-%02d %02d:%02d:%02d\n",(ulong)(uint)local_18->tm_year,
         (ulong)(uint)local_18->tm_mon,(ulong)(uint)local_18->tm_mday,(ulong)(uint)local_18->tm_wday
         ,(ulong)(uint)local_18->tm_min,(ulong)(uint)local_18->tm_yday,uVar1);
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return;
}
```

I tried to make the correspondence of tm structure components with respect to this table


Structure Editor - tm (bin1) [CodeBrowser: lab7:/bin1]
Help
Structure Editor - tm (bin1)

Offset	Length	Mnemonic	DataType	Name
0	4	int	int	tm_sec
4	4	int	int	tm_min
8	4	int	int	tm_hour
12	4	int	int	tm_mday
16	4	int	int	tm_mon
20	4	int	int	tm_year
24	4	int	int	tm_wday
28	4	int	int	tm_yday
32	4	int	int	tm_isdst
40	8	long	long	tm_gmtoff
48	8	char *	char *	tm_zone

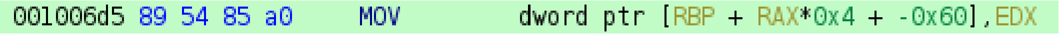


```c
#include <stdio.h>
#include <time.h>

int main() {
    struct tm *timer;
    time_t time_;
    time(&time_);
    timer = localtime(&time_);
    printf("%04d-%02d-%02d %02d:%02d:%02d\n",
    timer->tm_year,timer->tm_mon,
    timer->tm_mday, timer->tm_wday,
    timer->tm_min, timer->tm_yday);
    return 0;
}
```

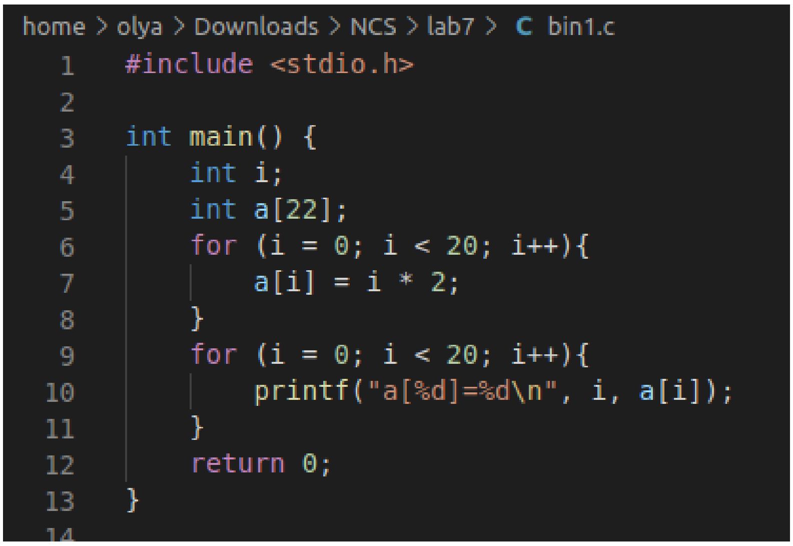
bin2

Since both for and while become while on compilation, I was not sure which is more correct.

Also, it took me a while to get what this line does

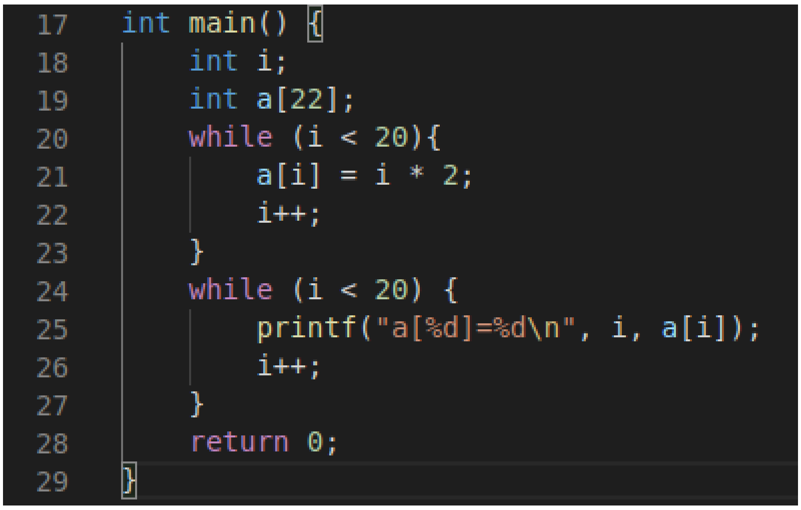


```
001006d5 89 54 85 a0     MOV        dword ptr [RBP + RAX*0x4 + -0x60],EDX
```



home > olya > Downloads > NCS > lab7 > C bin1.c

```c
#include <stdio.h>

int main() {
    int i;
    int a[22];
    for (i = 0; i < 20; i++){
        a[i] = i * 2;
    }
    for (i = 0; i < 20; i++){
        printf("a[%d]=%d\n", i, a[i]);
    }
    return 0;
}
```



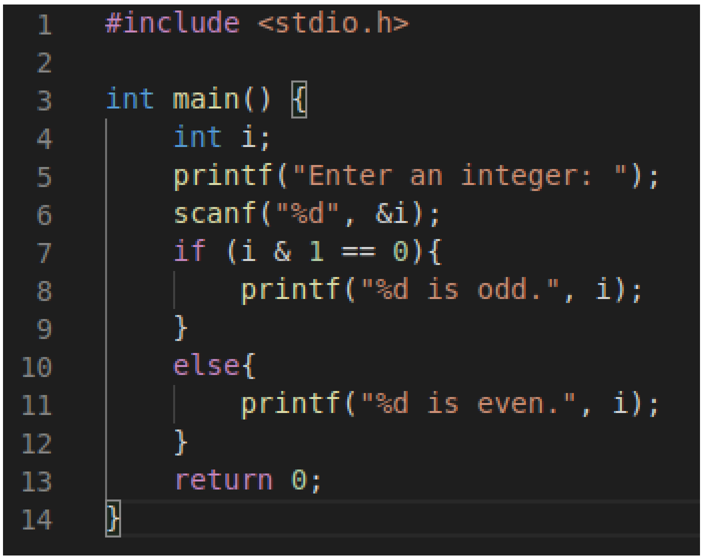
```c
int main() {
    int i;
    int a[22];
    while (i < 20){
        a[i] = i * 2;
        i++;
    }
    while (i < 20) {
        printf("a[%d]=%d\n", i, a[i]);
        i++;
    }
    return 0;
}
```

bin3

As described in task 2.1, this file has the same hash as bin2, so they are the same.

bin4

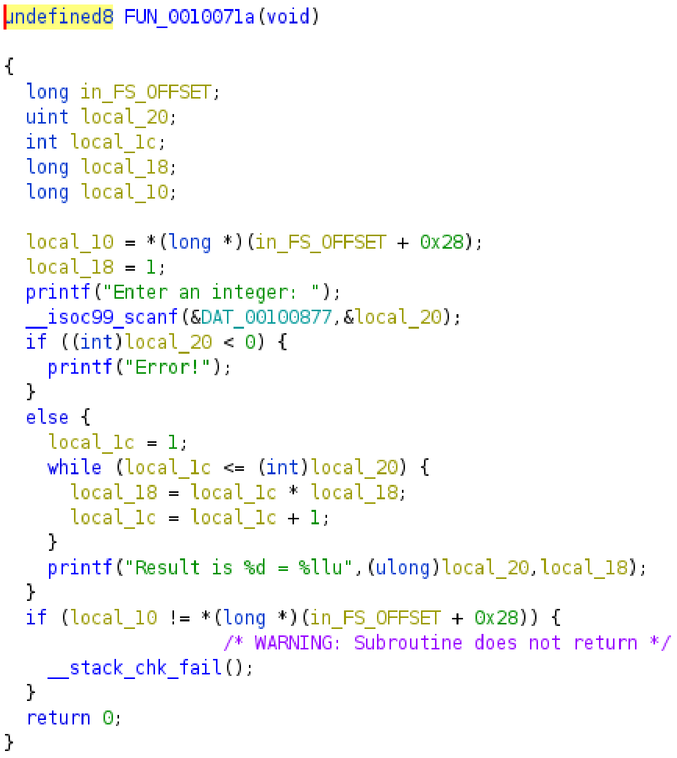
My logic said “Goodbye, my dear!” on the stage of i&1, and I hope it really produces 0 when the number is not divisible by 2



```c
#include <stdio.h>

int main() {
    int i;
    printf("Enter an integer: ");
    scanf("%d", &i);
    if (i & 1 == 0){
        printf("%d is odd.", i);
    }
    else{
        printf("%d is even.", i);
    }
    return 0;
}
```

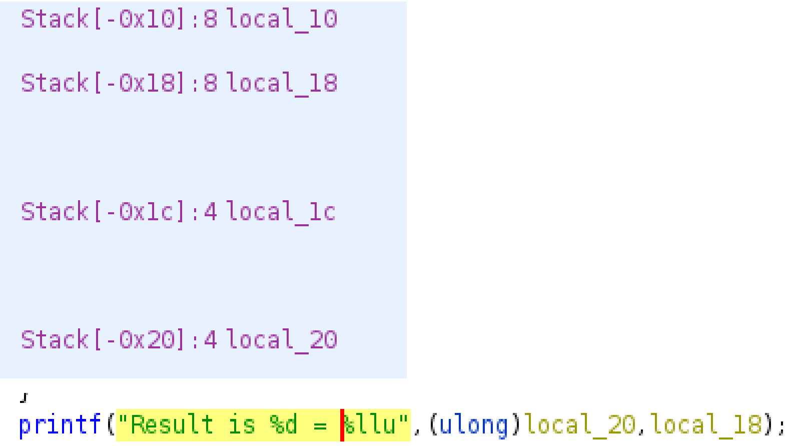
bin5

This one didn’t have an explicitly named main function in Ghidra, so I decompiled the one that looked similar to the mains in previous binaries and this function was actually the only one that had something humanly-meaningful inside.



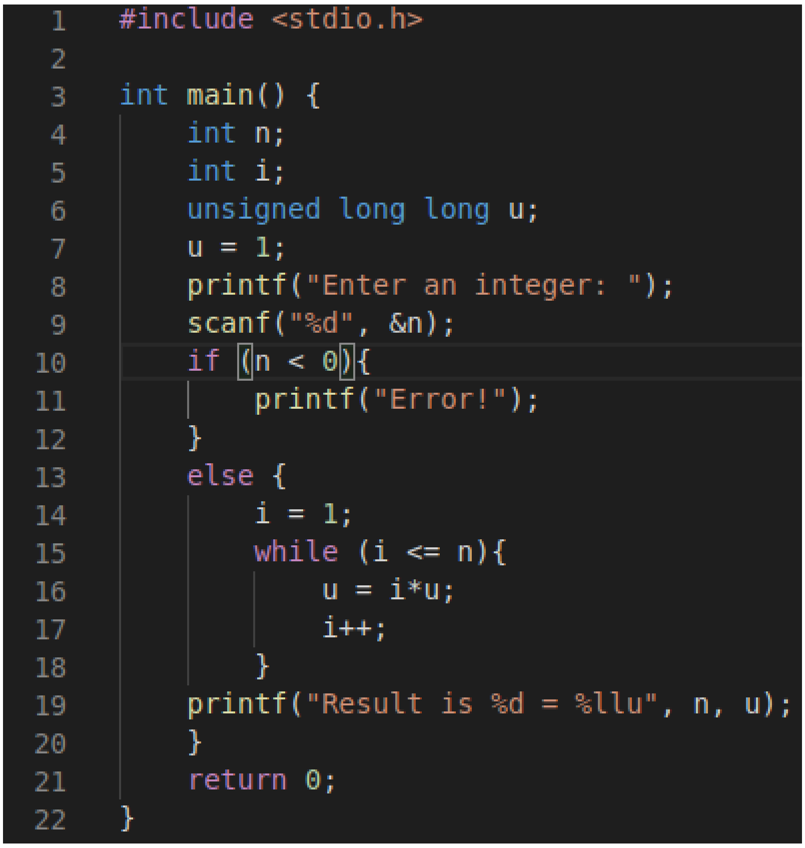
```c
undefined8 FUN_0010071a(void)

{
  long in_FS_OFFSET;
  uint local_20;
  int local_1c;
  long local_18;
  long local_10;

  local_10 = *(long *)(in_FS_OFFSET + 0x28);
  local_18 = 1;
  printf("Enter an integer: ");
  __isoc99_scanf(&DAT_00100877,&local_20);
  if ((int)local_20 < 0) {
    printf("Error!");
  }
  else {
    local_1c = 1;
    while (local_1c <= (int)local_20) {
      local_18 = local_1c * local_18;
      local_1c = local_1c + 1;
    }
    printf("Result is %d = %llu",(ulong)local_20,local_18);
  }
  if (local_10 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    __stack_chk_fail();
  }
  return 0;
}
```

The offsets in the header and llu in printf helped to guess the type of local_18.



```
Stack[-0x10]:8 local_10

Stack[-0x18]:8 local_18


Stack[-0x1c]:4 local_1c


Stack[-0x20]:4 local_20

printf("Result is %d = %llu",(ulong)local_20,local_18);
```

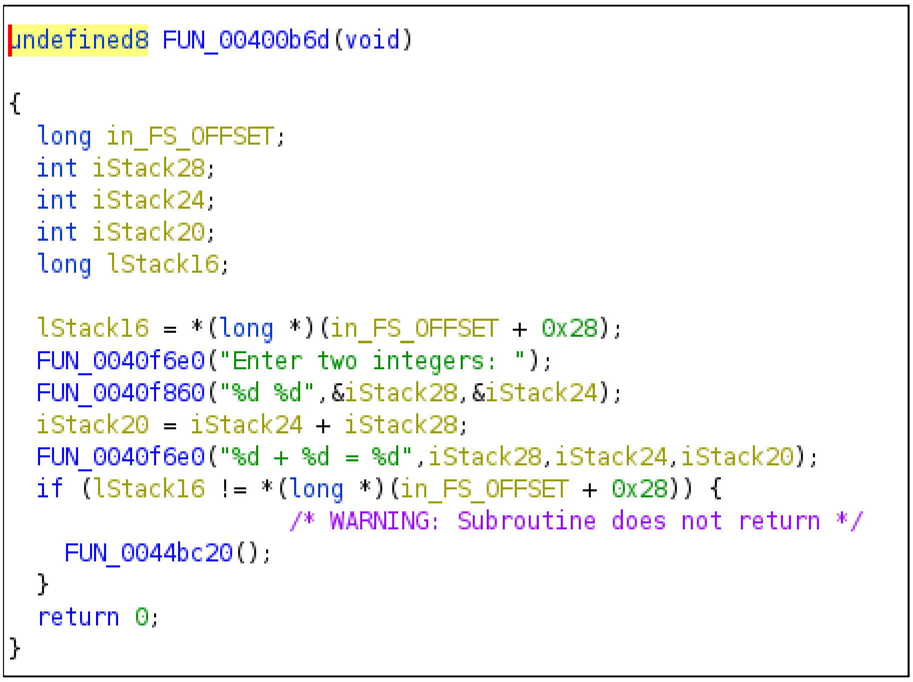
Finally, it is here :)



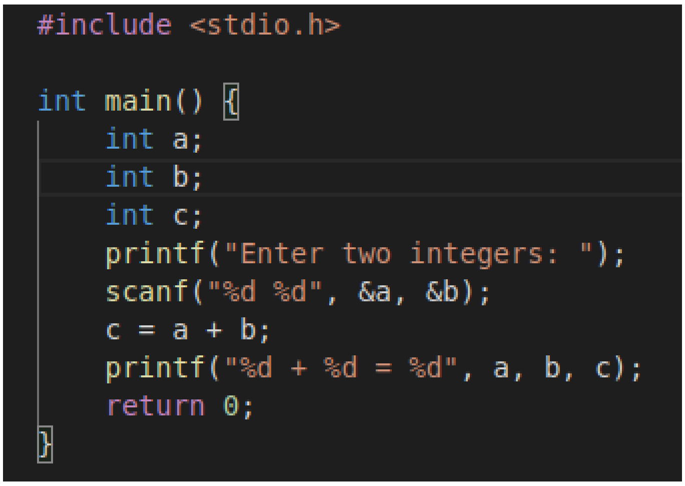
```c
#include <stdio.h>

int main() {
    int n;
    int i;
    unsigned long long u;
    u = 1;
    printf("Enter an integer: ");
    scanf("%d", &n);
    if (n < 0){
        printf("Error!");
    }
    else {
        i = 1;
        while (i <= n){
            u = i*u;
            i++;
        }
    printf("Result is %d = %llu", n, u);
    }
    return 0;
}
```

bin6

I was almost crying searching for the main function, and decided to decompile this one as it seemed more similar to what I saw previously. Later discussing it with friends they told me they used Cutter to find the main and (thanks, god!) it appeared to be the same.

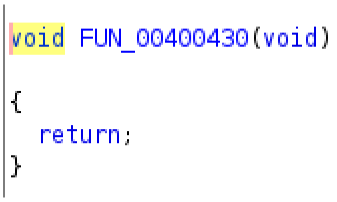


```
undefined8 FUN_00400b6d(void)

{
  long in_FS_OFFSET;
  int iStack28;
  int iStack24;
  int iStack20;
  long lStack16;

  lStack16 = *(long *)(in_FS_OFFSET + 0x28);
  FUN_0040f6e0("Enter two integers: ");
  FUN_0040f860("%d %d",&iStack28,&iStack24);
  iStack20 = iStack24 + iStack28;
  FUN_0040f6e0("%d + %d = %d",iStack28,iStack24,iStack20);
  if (lStack16 != *(long *)(in_FS_OFFSET + 0x28)) {
                    /* WARNING: Subroutine does not return */
    FUN_0044bc20();
  }
  return 0;
}
```



```
#include <stdio.h>

int main() {
    int a;
    int b;
    int c;
    printf("Enter two integers: ");
    scanf("%d %d", &a, &b);
    c = a + b;
    printf("%d + %d = %d", a, b, c);
    return 0;
}
```

In the process of doing this task I found my soulmate function. It does as meaningful and valuable job as I do.



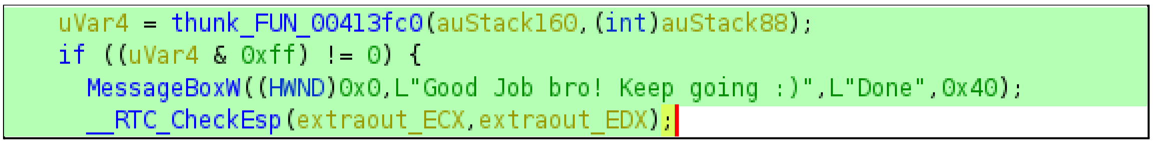
```
void FUN_00400430(void)

{
  return;
}
```

Task 4 - Catch the password from PE

task1.exe

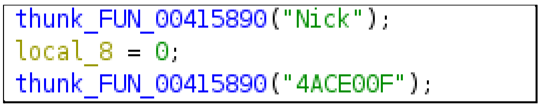
I want to thank my parents and “Method Тыка” for being able to solve this

I found this function that has something related to password in it



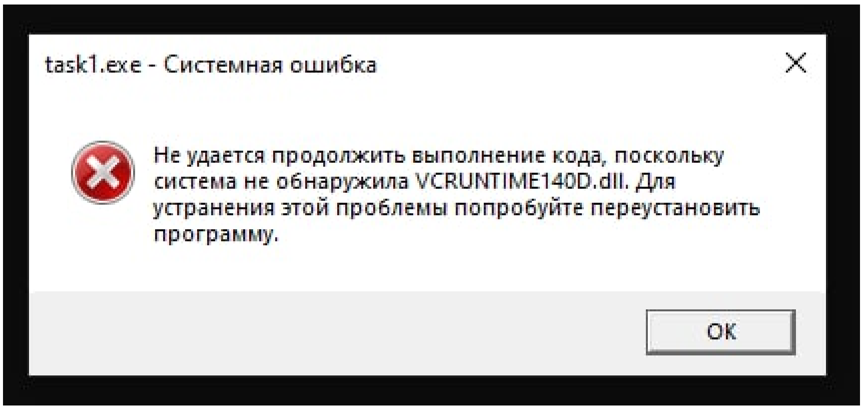
```
  uVar4 = thunk_FUN_00413fc0(auStack160,(int)auStack88);
  if ((uVar4 & 0xff) != 0) {
    MessageBoxW((HWND)0x0,L"Good Job bro! Keep going :)",L"Done",0x40);
    __RTC_CheckEsp(extraout_ECX,extraout_EDX);
```

In the same function several lines above there was something suspicious



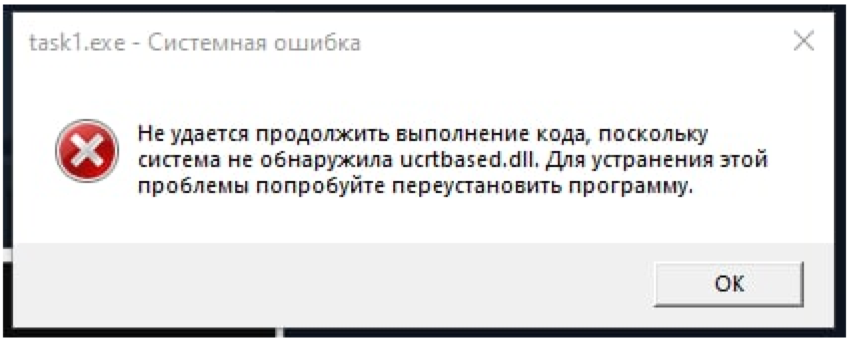
```
thunk_FUN_00415890("Nick");
local_8 = 0;
thunk_FUN_00415890("4ACE00F");
```

Unfortunately, I couldn’t run it to check



task1.exe - Системная ошибка

Не удается продолжить выполнение кода, поскольку система не обнаружила VCRUNTIME140D.dll. Для устранения этой проблемы попробуйте переустановить программу.

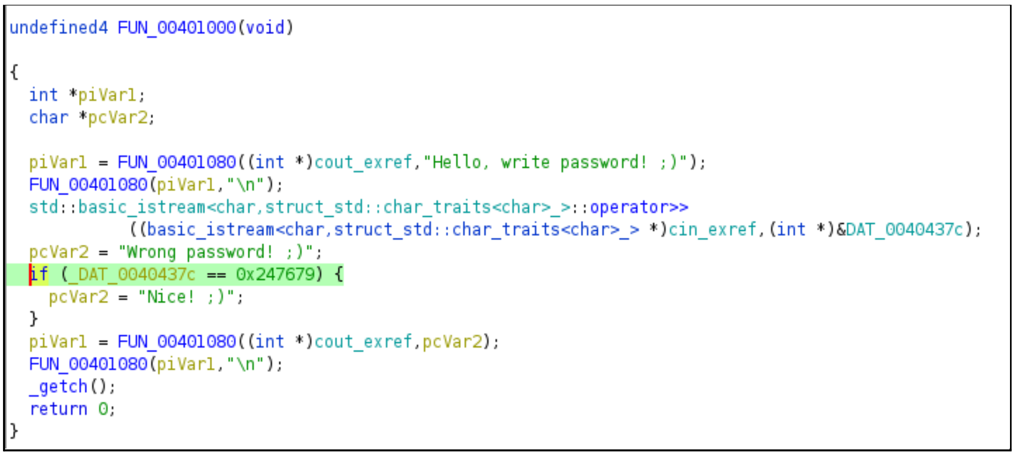


task1.exe - Системная ошибка

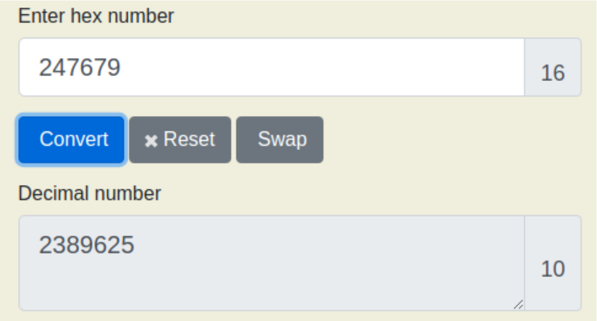
Не удается продолжить выполнение кода, поскольку система не обнаружила ucrtbased.dll. Для устранения этой проблемы попробуйте переустановить программу.

task2.exe

It seemed to be easier that the first one, I just followed the instructions given in the lab.



```
undefined4 FUN_00401000(void)

{
  int *piVar1;
  char *pcVar2;

  piVar1 = FUN_00401080((int *)cout_exref,"Hello, write password! ;)");
  FUN_00401080(piVar1,"\n");
  std::basic_istream<char,struct_std::char_traits<char>_>::operator>>
            ((basic_istream<char,struct_std::char_traits<char>_> *)cin_exref,(int *)&DAT_0040437c);
  pcVar2 = "Wrong password! ;)";
  if (_DAT_0040437c == 0x247679) {
    pcVar2 = "Nice! ;)";
  }
  piVar1 = FUN_00401080((int *)cout_exref,pcVar2);
  FUN_00401080(piVar1,"\n");
  _getch();
  return 0;
}
```

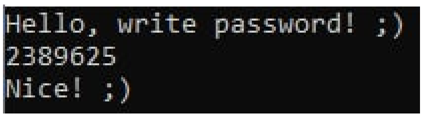


Enter hex number

247679 16

Decimal number

2389625 10



```
Hello, write password! ;)
2389625
Nice! ;)
```

To conclude in a couple of words - I am extremely sorry for choosing Reverse Engineering elective, since I anticipate completing it white-haired and having a twitching eye. Though, it was interesting.