

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(МОСКОВСКИЙ ПОЛИТЕХ)**

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
КАФЕДРА «ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ»

Отчет о выполнении задания на экзамене по дисциплине

Реверс инжиниринг программного кода

Выполнила: студентка 4 курса
Группы 171-341
Решетникова Дарья Алексеевна

Преподаватель:
Недогарок Антон Александрович

Москва 2021 г.

Вариант №8

Блок 1

Блок 1.1

Привести в отчёте снимки заголовков PE-файла. Найти:

- Границы DOS-заголовка.
- Границы основного и дополнительного PE-заголовка.

После сигнатуры идёт основной PE-заголовок, который называется COFF File Header (Object and Image).

- Поле Machine.
- Поле Characteristics.

Привести снимки найденных и выделенных элементов в отчёте.

The screenshot shows the HxD hex editor interface. The file being edited is 'Var8_Reverse_Exam.exe'. The hex view shows the following data:

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Текст декодирован	
00000000	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....яя..	
00000010	B8	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	ë.....@.....	
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	F0	00	00p...	
00000040	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	...e...r.H!ë.LH!Th	
00000050	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	is program canno	
00000060	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	t be run in DOS	
00000070	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	mode....\$.....	
00000080	BE	CE	E2	F0	FA	AF	8C	A3	FA	AF	8C	A3	FA	AF	8C	A3	sOapъIъJъIъJъIъJ	
00000090	F3	D7	1F	A3	EA	AF	8C	A3	44	DE	88	A2	F0	AF	8C	A3	yЧ.JкIъJDKъJpIъJ	
000000A0	44	DE	8F	A2	F9	AF	8C	A3	44	DE	89	A2	E0	AF	8C	A3	DKOъMъIъJDKOъyaIъJ	
000000B0	44	DE	8D	A2	FD	AF	8C	A3	A1	C7	8D	A2	F8	AF	8C	A3	DKOъeIъJyъKъMъIъJ	
000000C0	FA	AF	8D	A3	9C	AF	8C	A3	62	DD	85	A2	F9	AF	8C	A3	ъIKъMъIъJbъS...yъMъIъJ	
000000D0	62	DD	73	A3	FB	AF	8C	A3	62	DD	8E	A2	FB	AF	8C	A3	bъSъMъIъJbъSъMъIъJ	
000000E0	52	69	63	68	FA	AF	8C	A3	00	00	00	00	00	00	00	00	RichъIъJ.....	
000000F0	50	45	00	00	64	86	06	00	BC	49	12	60	00	00	00	00	PE..dt+..jI.`....	
00000100	00	00	00	00	F0	00	22	00	0B	02	0E	1C	00	20	00	00p.".....	
00000110	00	36	00	00	00	00	00	00	14	22	00	00	00	10	00	00	.6.....".	
00000120	00	00	00	40	01	00	00	00	10	00	00	00	00	02	00	00	...@.....	
00000130	06	00	00	00	00	00	00	00	06	00	00	00	00	00	00	00	
00000140	00	A0	00	00	00	04	00	00	00	00	00	00	00	03	00	60	81f
00000150	00	00	10	00	00	00	00	00	10	00	00	00	00	00	00	00	
00000160	00	00	10	00	00	00	00	00	10	00	00	00	00	00	00	00	
00000170	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	
00000180	54	45	00	00	04	01	00	00	80	00	00	E0	01	00	00	00	TE.....ъ..a...	
00000190	00	70	00	00	DC	02	00	00	00	00	00	00	00	00	00	00	.p..b.....	
000001A0	00	90	00	00	5C	00	00	00	40	39	00	00	70	00	00	00	.ъ..\\...@ъ..p...	
000001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001C0	00	00	00	00	00	00	00	00	B0	39	00	00	38	01	00	00ъ..8...	
000001D0	00	00	00	00	00	00	00	00	30	00	00	08	03	00	00	000.....	
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000001F0	00	00	00	00	00	00	00	00	FE	74	65	78	74	00	00	00text...	
00000200	73	1F	00	00	10	00	00	00	20	00	00	00	04	00	00	00	s.....	
00000210	00	00	00	00	00	00	00	00	00	00	00	20	00	00	00	60`	

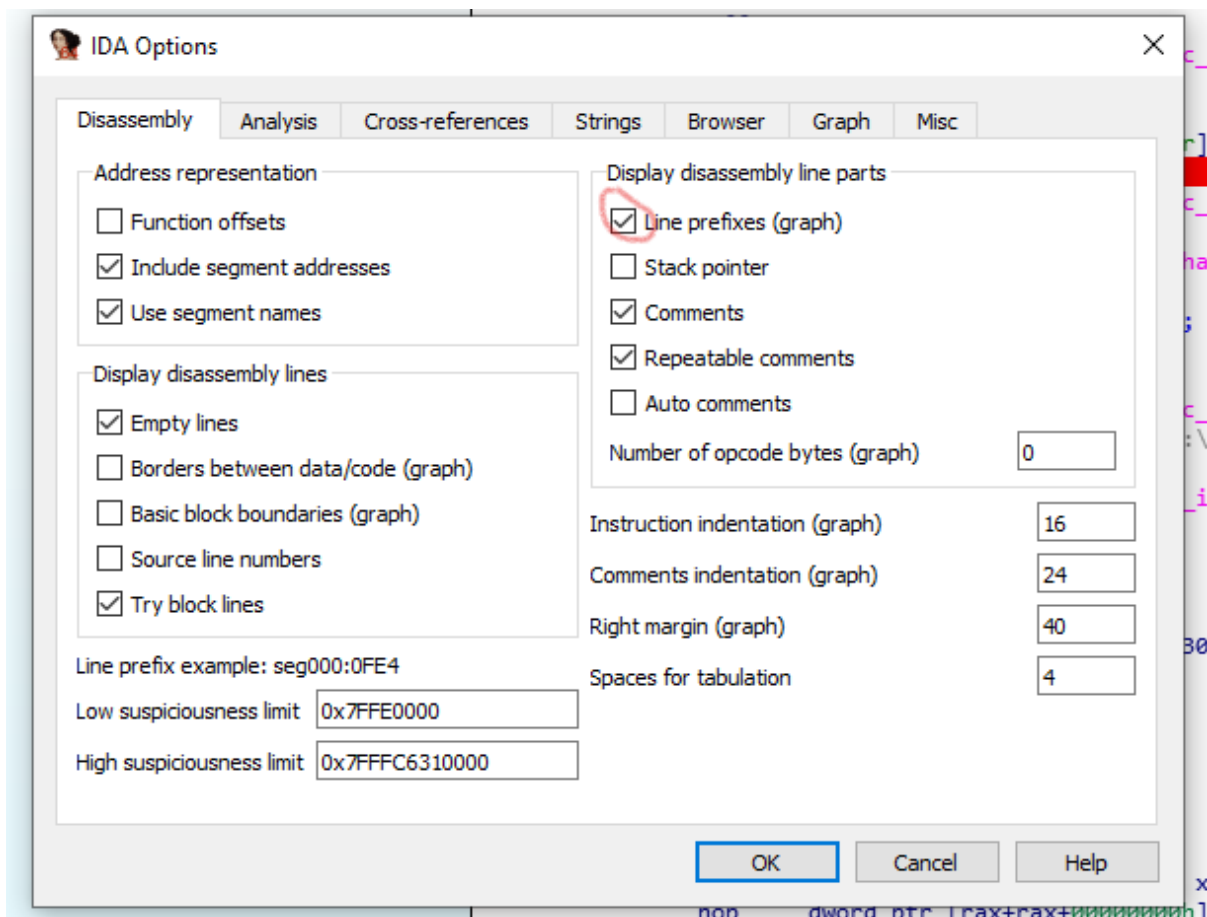
Annotations on the right side of the image:

- DOS-заголовок (Red box around offset 00000000-0000000F)
- Сигнатура (Blue box around offset 00000040-0000004F)
- Machine (Pink box around offset 00000045-00000046)
| Characteristics (Orange box around offset 00000047-00000048) | |
| Основной PE-заголовок (Blue vertical line at offset 00000050) | |
| Дополнительный PE-заголовок (Green vertical line at offset 00000051) | |

Блок 1.2

Найти, пометить комментариями в дизассемблере и привести в отчёте (в виде снимка экрана дизассемблера):

- Адреса (указать числа) и фрагменты кода (снимок), по которым расположены инструкции для считывания символов ключа и серийного номера из интерфейса в память приложения.



```

00007FF6C4BC1419      call     cs:strcmp_s
00007FF6C4BC141F      lea      r8, [rsp+260h+Src+3] ; Src
00007FF6C4BC1424      mov      [rsp+260h+var_20D], 20h
00007FF6C4BC1429      lea      edx, [rbx+4] ; SizeInBytes
00007FF6C4BC142C      lea      rcx, [rsp+260h+var_20C] ; Dst
00007FF6C4BC1431      call     cs:strcmp_s
00007FF6C4BC1437      mov      rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostr
00007FF6C4BC143E      lea      rdx, aSerial ; "Serial: "
00007FF6C4BC1445      call     sub_7FF6C4BC16F0
00007FF6C4BC144A      lea      rcx, [rsp+260h+serial_buffer] ; Format
00007FF6C4BC144F      call     printf
00007FF6C4BC1454      mov      rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostr
00007FF6C4BC145B      lea      rdx, sub_7FF6C4BC1A80
00007FF6C4BC1462      call     cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAEAV01@P6AAEAV01@AEAV01@@@Z6
00007FF6C4BC1468      xor      edx, edx ; Val
00007FF6C4BC146A      lea      rcx, [rbp+160h+key_buffer] ; Dst
00007FF6C4BC146E      mov      r8d, 100h ; Size
00007FF6C4BC1474      call     memset
00007FF6C4BC1479      mov      rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostr
00007FF6C4BC1480      lea      rdx, aEnterKey ; "Enter key:\t"
00007FF6C4BC1487      call     sub_7FF6C4BC16F0
00007FF6C4BC148C      mov      rcx, cs:?cin@std@@3V?$basic_istream@DU?$char_traits@D@std@@@1@A ; std::basic_istre
00007FF6C4BC1493      lea      r8, [rbp+160h+key_buffer]
00007FF6C4BC1497      call     input_key
00007FF6C4BC149C      mov      eax, cs:dword_7FF6C4BC3440
00007FF6C4BC14A2      xorps    xmm0, xmm0
00007FF6C4BC14A5      movups   xmm1, cs:xmmword_7FF6C4BC3430
00007FF6C4BC14AC      mov      [rsp+260h+var_220], eax
00007FF6C4BC14B0      movzx    eax, cs:word_7FF6C4BC3444
00007FF6C4BC14B7      mov      [rsp+260h+var_21C], ax
00007FF6C4BC14BC      movzx    eax, cs:byte_7FF6C4BC3446
00007FF6C4BC14C3      mov      [rsp+260h+var_21A], al
00007FF6C4BC14C7      movsdb   [rsp+260h+var_240], xmm0
00007FF6C4BC14CD      mov      [rbp+160h+var_11], bl
00007FF6C4BC14D3      movups   xmmword ptr [rsp+260h+Str], xmm1
00007FF6C4BC14D8      nop      dword ptr [rax+rax+00000000h]

```

- Адреса (указать числа) и фрагменты кода (снимок), по которым расположено начало сверки серийного номера с ключом.

Здесь располагается какой-то подготовительный код, который будет использоваться далее при проверке пароля. (По факту это значения из ASCII-таблицы, которые используются в serial и будут использоваться в пароле. Подключив логику я подобрала правильный пароль с помощью этих значений):

```

00007FF6C4BC146A      lea      rcx, [rbp+160h+key_buffer] ; Dst
00007FF6C4BC146E      mov      r8d, 100h ; Size
00007FF6C4BC1474      call     memset
00007FF6C4BC1479      mov      rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<char,std::char_traits<char>>
00007FF6C4BC1480      lea      rdx, aEnterKey ; "Enter key:\t"
00007FF6C4BC1487      call     sub_7FF6C4BC16F0
00007FF6C4BC148C      mov      rcx, cs:?cin@std@@3V?$basic_istream@DU?$char_traits@D@std@@@1@A ; std::basic_istream<char,std::char_traits<char>>
00007FF6C4BC1493      lea      r8, [rbp+160h+key_buffer]
00007FF6C4BC1497      call     input_key
00007FF6C4BC149C      mov      eax, cs:dword_7FF6C4BC3440 ; 49484746h
00007FF6C4BC14A2      xorps    xmm0, xmm0
00007FF6C4BC14A5      movups   xmm1, cs:xmmword_7FF6C4BC3430 ; 30h, 31h, 32h, 33h, 34h, 35h, 36h, 37h, 38h, 39h, 20h, 41h, 42h, 43h
00007FF6C4BC14AC      mov      [rsp+260h+var_220], eax
00007FF6C4BC14B0      movzx    eax, cs:word_7FF6C4BC3444 ; 4Ah, 78h
00007FF6C4BC14B7      mov      [rsp+260h+var_21C], ax
00007FF6C4BC14BC      movzx    eax, cs:byte_7FF6C4BC3446
00007FF6C4BC14C3      mov      [rsp+260h+var_21A], al
00007FF6C4BC14C7      movsdb   [rsp+260h+var_240], xmm0
00007FF6C4BC14CD      mov      [rbp+160h+var_11], bl
00007FF6C4BC14D3      movups   xmmword ptr [rsp+260h+Str], xmm1
00007FF6C4BC14D8      nop      dword ptr [rax+rax+00000000h]

```

```

.rdata:00007FF6C4BC3418 ; DATA XREF: sub_7FF6C4BC2900+510
.rdata:00007FF6C4BC342D align 10h
.rdata:00007FF6C4BC3430 xmmword_7FF6C4BC3430 db 30h, 31h, 32h, 33h, 34h, 35h, 36h, 37h, 38h, 39h, 2Dh, 41h, 42h, 43h
.rdata:00007FF6C4BC3430 ; DATA XREF: sub_7FF6C4BC1270+2351r
.rdata:00007FF6C4BC3430 db 44h, 45h
.rdata:00007FF6C4BC3440 dword_7FF6C4BC3440 dd 49484746h ; DATA XREF: sub_7FF6C4BC1270+22C1r
.rdata:00007FF6C4BC3444 word_7FF6C4BC3444 db 4Ah, 78h ; DATA XREF: sub_7FF6C4BC1270+2401r
.rdata:00007FF6C4BC3446 byte_7FF6C4BC3446 db 0 ; DATA XREF: sub_7FF6C4BC1270+24C1r
.rdata:00007FF6C4BC3447 align 8
.rdata:00007FF6C4BC3448 aUnknownExcepti db 'Unknown exception',0

```

Начало сверки серийного номера с ключом:

```

.r], xmm1
0000h]
00007FF6C4BC14E0
00007FF6C4BC14E0 check_key_serial: ; Val
00007FF6C4BC14E0 movsx edx, [rsp+rbx+260h+serial_buffer]
00007FF6C4BC14E5 lea rcx, [rsp+260h+Str] ; Str
00007FF6C4BC14EA call cs:stchr
00007FF6C4BC14F0 movzx ecx, byte ptr [rax+08h]
00007FF6C4BC14F4 cmp [rbp+rbx+160h+key_buffer], cl ; сравнение 2 операндов (вводимое значение, правильное значение - их код в ASCII-таблице)
00007FF6C4BC14F8 jnz display_wrong ; если сравнение не равно (JNZ) оно переносит к Activation failed,
; а если равно, то начинается проверка каждого символа (зациклено)

```

- Адреса (указать числа) и фрагменты кода (снимок), в котором происходит получение логического значения (true или false), зависящее от верности серийного номера и пароля.

```

00007FF6C4BC14E0
00007FF6C4BC14E0 check_key_serial: ; Val
00007FF6C4BC14E0 movsx edx, [rsp+rbx+260h+serial_buffer]
00007FF6C4BC14E5 lea rcx, [rsp+260h+Str] ; Str
00007FF6C4BC14EA call cs:stchr
00007FF6C4BC14F0 movzx ecx, byte ptr [rax+08h]
00007FF6C4BC14F4 cmp [rbp+rbx+160h+key_buffer], cl ; сравнение 2 операндов (вводимое значение, правильное значение
00007FF6C4BC14F8 jnz display_wrong ; если сравнение не равно (JNZ) оно переносит к Activation failed,
; а если равно, то начинается проверка каждого символа (зациклено)
00007FF6C4BC14FF inc rcx ; инкремент +1

```

- Адреса (указать числа) и фрагменты кода (снимок) с инструкцией, выполняющейся первой в ветке приложения, соответствующей верным ключу и серийному номеру.

Если первое значение правильное, то переходит на эту ветку:

```

00007FF6C4BC14F8 ; а если равно, то начинается проверка каждого символа (зациклено)
00007FF6C4BC14FE inc rcx ; инкремент +1
00007FF6C4BC1501 cmp rcx, 7
00007FF6C4BC1505 jl short check_key_serial ; если меньше

```

```
00007FF6C48C14E0 ; Val
00007FF6C48C14E0 check key_serial:
00007FF6C48C14E0 movsx edx, [rsp+rbx*20h+serial_buffer]
00007FF6C48C14E5 lea rcx, [rsp+260h+Str] ; Str
00007FF6C48C14EA call cs:strcmp
00007FF6C48C14F0 movzx ecx, byte ptr [rax+08h]
00007FF6C48C14F4 cmp [rbp+rbx*16h+key_buffer], cl ; сравнение 2 операндов (вводимое значение, правильное значение - их код в ASCII-таблице)
00007FF6C48C14F8 jnz display_wrong ; если сравнение не равно (JNZ) оно переносит к Activation failed,
; а если равно, то начинается проверка каждого символа (защелкнуто)

00007FF6C48C14FE inc rbx ; инкремент +1
00007FF6C48C1501 cmp rbx, 7
00007FF6C48C1505 jl short check_key_serial ; если меньше
```

```

00007FF6C4BC1507    lea     rcx, aSuccessfulKeyC ; "Successful key check!", display_valid
00007FF6C4BC150E    call    printf
00007FF6C4BC1513    mov     rcx, cs:??cout@std@@3V?$basic_ostream@DU?$char_traits@0@std@@@1@A ; std::basic_ostream<char,std::char_traits<char>> st
00007FF6C4BC151A    lea     rdx, sub_7FF6C4BC1A80
00007FF6C4BC1521    call    cs:??f6?$basic_ostream@DU?$char_traits@0@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; std::basic_ostream<char,std::chu
00007FF6C4BC1527    mov     rcx, rax
00007FF6C4BC152A    lea     rdx, aTypeDoubleX ; "Type double X: "
00007FF6C4BC1531    call    sub_7FF6C4BC16F0
00007FF6C4BC1536    mov     rcx, cs:??cin@std@@3V?$basic_istream@DU?$char_traits@0@std@@@1@A ; std::basic_istream<char,std::char_traits<char>> std:
00007FF6C4BC153D    lea     rdx, [rsp+260h+var_240]
00007FF6C4BC1542    call    cs:??f5?$basic_istream@DU?$char_traits@0@std@@@std@@QEAAAEAV01@AEAH@Z ; std::basic_istream<char,std::char_traits<char>
00007FF6C4BC1548    movsd   xmm0, [rsp+260h+var_240]
00007FF6C4BC154E    call    get_f_x ; расчёт значения функции "Math equation result"
00007FF6C4BC1553    movaps  xmm1, xmm0
00007FF6C4BC1556    lea     rcx, aMathEquationRe ; "Math equation result:\t%f"
00007FF6C4BC155D    movq    rdx, xmm0
00007FF6C4BC1562    call    printf
00007FF6C4BC1567    mov     rcx, cs:??cout@std@@3V?$basic_ostream@DU?$char_traits@0@std@@@1@A ; std::basic_ostream<char,std::char_traits<char>> st
00007FF6C4BC156E    lea     rdx, sub_7FF6C4BC1A80
00007FF6C4BC1575    call    cs:??f6?$basic_ostream@DU?$char_traits@0@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; std::basic_ostream<char,std::chu
00007FF6C4BC157B    lea     rcx, Command ; "pause"
00007FF6C4BC1582    call    cs:system
00007FF6C4BC1588    xor     eax, eax
00007FF6C4BC158A    jmp     short loc_7FF6C4BC15C1

```

- Если не сходится значение, то из этого цикла:

```
[tr], xmm1
00000000]

00007FF6C4BC14E0
00007FF6C4BC14E0 check_key_serial: ; Val
00007FF6C4BC14E0 movsx   edx, [rsp+rbx+260h+serial_buffer]
00007FF6C4BC14E5 lea     rcx, [rsp+260h+Str] ; Str
00007FF6C4BC14EA call    cs:strchr
00007FF6C4BC14F0 movzx   ecx, byte ptr [rax+00h]
00007FF6C4BC14F4 [rbp+rbx+160h+key_buffer], cl ; сравнение 2 операндов (вводимое значение, правильное значение - их код в ASCII-таблице)
00007FF6C4BC14F8 cmp     display_wrong ; если сравнение не равно (JNZ) оно переносит к Activation failed,
00007FF6C4BC14F8 jnz     ; а если равно, то начинается проверка каждого символа (зациклено)
```

Переходит в этот:

```
00007FF6C4BC158C
00007FF6C4BC158C display_wrong:
00007FF6C4BC158C      mov     rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<char,
00007FF6C4BC1593      lea     rdx, aActivationFail ; "Activation failed =(. Exit"
00007FF6C4BC159A      call    sub_7FF6C4BC16F0
00007FF6C4BC159F      mov     rcx, rax
00007FF6C4BC15A2      lea     rdx, sub_7FF6C4BC1A80
00007FF6C4BC15A9      call    cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; std::l
00007FF6C4BC15AF      lea     rcx, Command ; "pause"
00007FF6C4BC15B6      call    cs:system
00007FF6C4BC15BC      mov     eax, 0FFFFFFFFh
```

- Вышеперечисленные адреса помечены разборчивыми комментариями в дизассемблере.
- В дизассемблере переменным, хранящим адреса вводимого ключа и серийного номера, присвоены имена “key_buffer” и “serial_buffer”.

С помощью отладки:

```
Moscow Polytechnic University
Reverse Engineering Exam 2020-2021

Serial: 970-290_
```

```
lea     edx, [rbx+7] ; SizeInBytes
lea     rcx, [rsp+260h+Dst] ; Dst
call    cs:strncpy_s
lea     r8, [rsp+260h+Src+3] ; Src
mov     [rsp+260h+var_20D], 2Dh
lea     edx, [rbx+4] ; SizeInBytes
lea     rcx, [rsp+260h+var_20C] ; Dst
call    cs:strncpy_s
mov     rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<char,std::char_trai
lea     rdx, aSerial ; "Serial: "
call    sub_7FF6C4BC16F0
lea     rcx, [rsp+260h+Dst] ; Format
call    printf
mov     rcx, cs:?cout@std@@[rsp+260h+Dst]=[Stack[00003888]:0000002DD6CFF910] ; std::basic_ostream<char,std::char_trai
lea     rdx, sub_7FF6C4BC1A80 ; db 39h ; 9
call    cs:??6?$basic_ostre ; db 37h ; 7
xor     edx, edx ; db 30h ; 0
lea     rcx, [rbp+160h+var_ ; db 2Dh ; -
mov     r8d, 100h ; S ; db 32h ; 2
call    memset ; db 39h ; 9
mov     rcx, cs:?cout@std@@ ; db 30h ; 0
lea     rdx, aEnterKey ; " ; db 0
call    sub_7FF6C4BC16F0 ; db 0
mov     rcx, cs:?cin@std@@3 ; db 0
lea     r8, [rbp+160h+var_110] ; db 0
call    input_key

.A 00007FF6C4BC144A: sub_7FF6C4BC1270+1DA (Synchronized with RIP)
```

Dst - это и есть serial_buffer.

```

lea rcx, [rsp+260h+serial_buffer] ; Dst
call cs:strncpy_s
lea r8, [rsp+260h+Src+3] ; Src
mov [rsp+260h+var_20D], 2Dh
lea edx, [rbx+4] ; SizeInBytes
lea rcx, [rsp+260h+var_20C] ; Dst
call cs:strncpy_s
mov rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@
lea rdx, aSerial ; "Serial: "
call sub_7FF6C4BC16F0
lea rcx, [rsp+260h+serial_buffer] ; Format
call printf
mov rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@
lea rdx, sub_7FF6C4BC1A80
call cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAEAV0
xor edx, edx ; Val
lea rcx, [rbp+160h+var_110] ; Dst
mov r8d, 100h ; Size
call memset
mov rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@
lea rdx, aEnterKey ; "Enter key:\t"
call sub_7FF6C4BC16F0

```

Также с помощью отладки:

```

Moscow Polytechnic University
Reverse Engineering Exam 2020-2021
Serial: 020-460
Enter key: 111222333

```

The screenshot shows a debugger window with the following assembly code and memory dump:

```

lea rdx, aSerial ; "Serial: "
call sub_7FF6C4BC16F0
lea rcx, [rsp+260h+serial_buffer] ; Format
call printf
mov rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<
lea rdx, sub_7FF6C4BC1A80
call cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; s
xor edx, edx ; Val
lea rcx, [rbp+160h+var_110] ; Dst
mov r8d, 100h ; Size
call memset
mov rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<
lea rdx, aEnterKey ; "Enter key:\t"
call sub_7FF6C4BC16F0
mov rcx, cs:?cin@std@@3V?$basic_istream@DU?$char_traits@D@std@@@1@A ; std::basic_istream<ch
lea r8, [rbp+160h+var_110]
call input_key
mov eax, cs:dword_7FF6C4BC1A80
xorps xmm0, xmm0
movups xmm1, cs:xmmword_7FF6C4BC1A80
mov [rsp+260h+var_220], eax
movzx eax, cs:word_7FF6C4BC1A80
mov [rsp+260h+var_21C], eax
movzx eax, cs:byte_7FF6C4BC1A80
mov [rsp+260h+var_21A], eax
movsd [rsp+260h+var_240], xmm1
mov [rbp+160h+var_11], byte_7FF6C4BC1A80
movups xmmword ptr [rsp+260h+var_210], xmm1

```

The memory dump shows the contents of [rbp+160h+var_110] as a stack of 10 bytes:

db	31h	; 1
db	31h	; 1
db	31h	; 1
db	32h	; 2
db	32h	; 2
db	32h	; 2
db	33h	; 3
db	33h	; 3
db	33h	; 3
db	0	

FF6C4BC1493: sub_7FF6C4BC1270+223 (Synchronized with RIP)

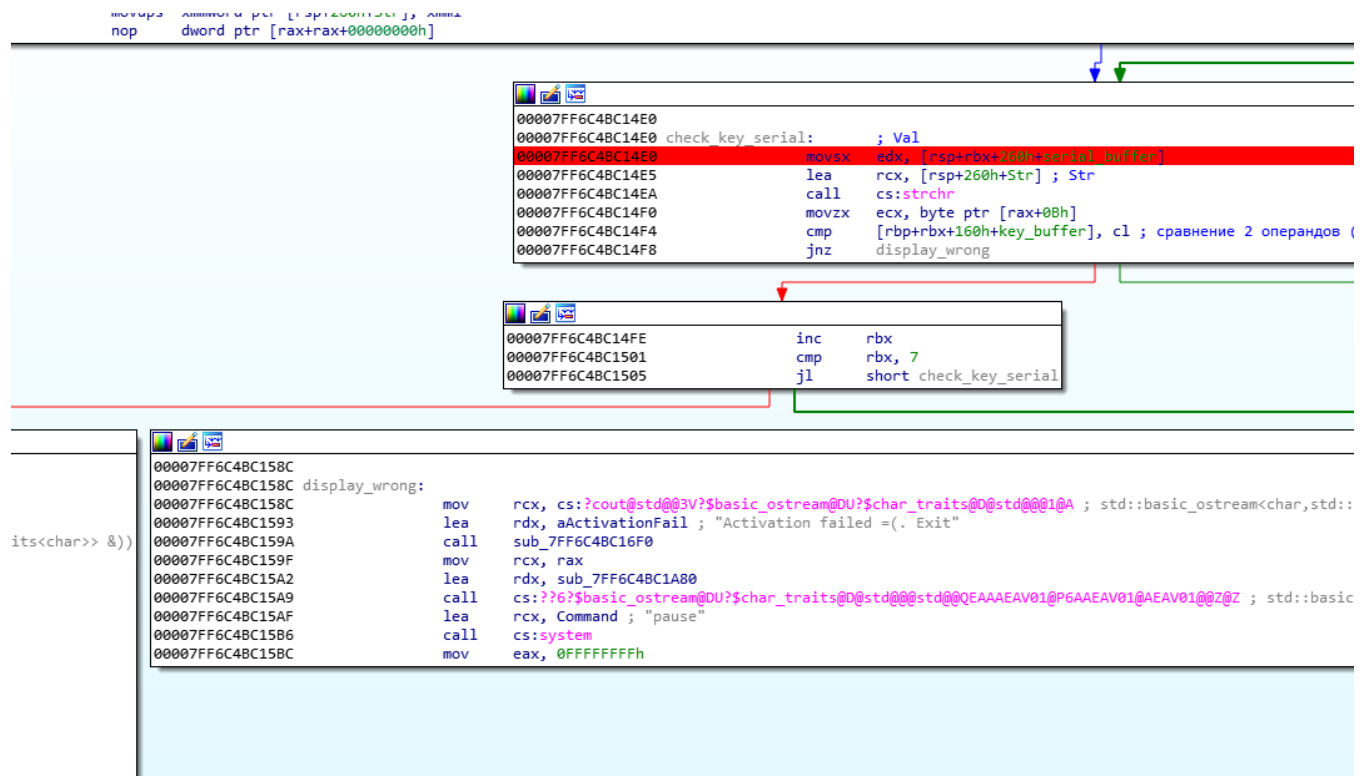

```

lea     rcx, [rsp+260h+var_20C] ; Dst
call    cs:strcpy_s
mov     rcx, cs:?cout@std@@@3V?$basic_ostream@DU?$char
lea     rdx, aSerial ; "Serial: "
call    sub_7FF6C4BC16F0
lea     rcx, [rsp+260h+serial_buffer] ; Format
call    printf
mov     rcx, cs:?cout@std@@@3V?$basic_ostream@DU?$char
lea     rdx, sub_7FF6C4BC1A80
call    cs:??6?$basic_ostream@DU?$char_traits@D@std@@
xor     edx, edx ; Val
lea     rcx, [rbp+160h+key_buffer] ; Dst
mov     r8d, 100h ; Size
call    memset
mov     rcx, cs:?cout@std@@@3V?$basic_ostream@DU?$char
lea     rdx, aEnterKey ; "Enter key:\t"
call    sub_7FF6C4BC16F0
mov     rcx, cs:?cin@std@@@3V?$basic_istream@DU?$char_
lea     r8, [rbp+160h+key_buffer]
call    input_key
mov     eax, cs:dword_7FF6C4BC3440
xorps   xmm0, xmm0
movups  xmm1, cs:xmmword_7FF6C4BC3430
mov     [rsp+260h+var_220], eax
movzx   eax, cs:word_7FF6C4BC3444
mov     [rsp+260h+var_21C], ax
movzx   eax, cs:byte_7FF6C4BC3446
mov     [rsp+260h+var_21A], al

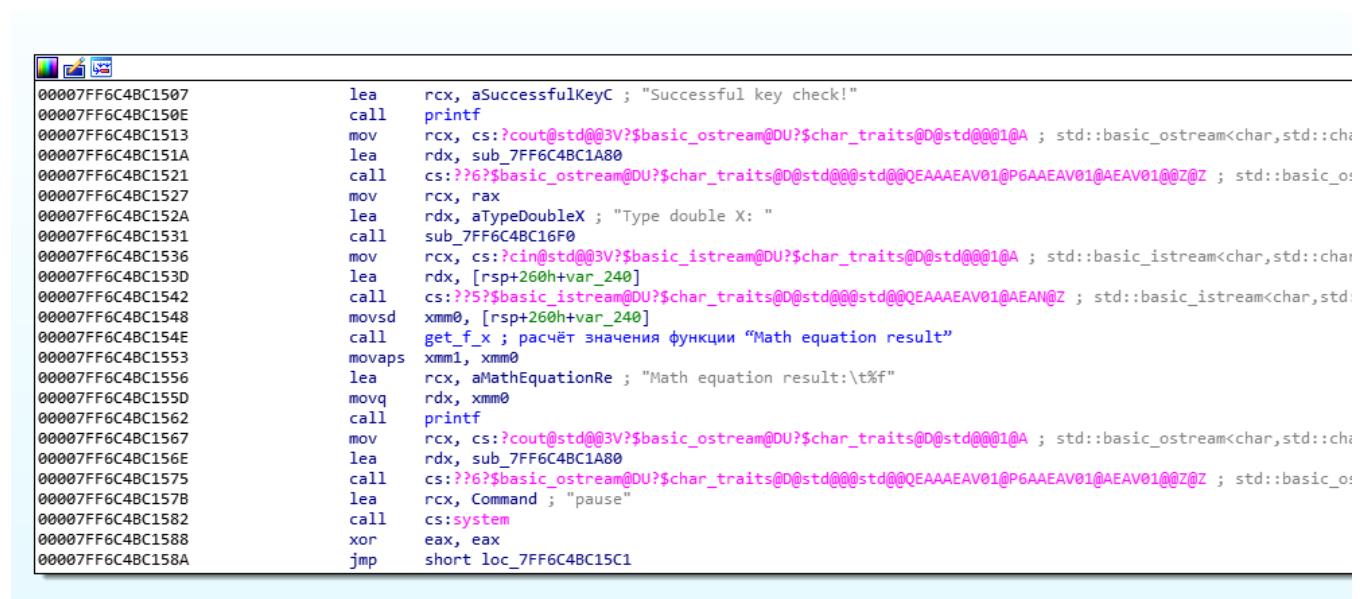
```

- Функциям, производящим 1) проверку корректности введённых ключа и серийного номера, 2) расчёт значения функции “Math equation result” 3) отображение сообщения о верном и неверном вводе ключа и серийного номера присвоены имена соответственно check_key_serial, get_f_x, display_valid, display_wrong.

check_key_serial с помощью отладки и по смыслу:



get_f_x по ее содержимому (умножение, вычитание и тд):



```

00007FF6C4BC1D40
00007FF6C4BC1D40
00007FF6C4BC1D40
00007FF6C4BC1D40 get_f_x      proc near
00007FF6C4BC1D40
00007FF6C4BC1D40 var_18      = qword ptr -18h
00007FF6C4BC1D40 arg_0       = qword ptr 8
00007FF6C4BC1D40
00007FF6C4BC1D40 movsd      [rsp+arg_0], xmm0
00007FF6C4BC1D46 sub        rsp, 38h
00007FF6C4BC1D4A movsd      xmm0, cs:qword_7FF6C4BC3938
00007FF6C4BC1D52 mulsd      xmm0, [rsp+38h+arg_0] ; X
00007FF6C4BC1D58 call       log10
00007FF6C4BC1D5D movsd      [rsp+38h+var_18], xmm0
00007FF6C4BC1D63 movsd      xmm1, cs:qword_7FF6C4BC3930
00007FF6C4BC1D6B addsd      xmm1, [rsp+38h+arg_0]
00007FF6C4BC1D71 movaps     xmm0, xmm1 ; X
00007FF6C4BC1D74 call       tan
00007FF6C4BC1D79 movsd      xmm1, [rsp+38h+var_18]
00007FF6C4BC1D7F divsd      xmm1, xmm0
00007FF6C4BC1D83 movaps     xmm0, xmm1 ; X
00007FF6C4BC1D86 call       exp
00007FF6C4BC1D8B add        rsp, 38h
00007FF6C4BC1D8F retn
00007FF6C4BC1D8F get_f_x      endp
00007FF6C4BC1D8F

```

display_wrong по ключевым словам "Activation failed =(Exit":

```

00007FF6C4BC158C
00007FF6C4BC158C display_wrong:
00007FF6C4BC158C mov        rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<char,sti
00007FF6C4BC1593 lea        rdx, aActivationFail ; "Activation failed =( Exit"
00007FF6C4BC159A call       sub_7FF6C4BC16F0
00007FF6C4BC159F mov        rcx, rax
00007FF6C4BC15A2 lea        rdx, sub_7FF6C4BC1A80
00007FF6C4BC15A9 call       cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@@QEAEEAV01@P6AAEAV01@AEAV01@@Z@Z ; std::ba
00007FF6C4BC15AF lea        rcx, Command ; "pause"
00007FF6C4BC15B6 call       cs:system
00007FF6C4BC15BC mov        eax, 0FFFFFFFh

```

display_valid (нет названия функции):

```

00007FF6C4BC1507    lea     rcx, aSuccessfulKeyC ; "Successful key check!", display_valid
00007FF6C4BC150E    call   printf
00007FF6C4BC1513    mov     rcx, cs:cout@std@3V?basic_ostream@DU?char_traits@D@std@@@1@A ; std::basic_ostream<char, std::char_traits<char>>::operator<<(...>>
00007FF6C4BC151A    lea     rdx, sub_7FF6C4BC1A80
00007FF6C4BC1521    call   cs:??6?basic_ostream@DU?char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; std::basic_ostream<char, std::char_traits<char>>::operator<<(...>>
00007FF6C4BC1527    mov     rcx, rax
00007FF6C4BC152A    lea     rdx, aTypeDoubleX ; "Type double X: "
00007FF6C4BC1531    call   sub_7FF6C4BC16F0
00007FF6C4BC1536    mov     rcx, cs:cin@std@3V?basic_istream@DU?char_traits@D@std@@@1@A ; std::basic_istream<char, std::char_traits<char>>::operator>>(...>>
00007FF6C4BC153D    lea     rdx, [rsp+260h+var_240]
00007FF6C4BC1542    call   cs:??5?basic_istream@DU?char_traits@D@std@@@std@@QEAAAEAV01@AEAV01@@Z@Z ; std::basic_istream<char, std::char_traits<char>>::operator>>(...>>
00007FF6C4BC1548    movsd   xmm0, [rsp+260h+var_240]
00007FF6C4BC154E    call   get_f_x ; расчёт значения функции "Math equation result"
00007FF6C4BC1553    movaps  xmm1, xmm0
00007FF6C4BC1556    lea     rcx, aMathEquationRe ; "Math equation result: %f"
00007FF6C4BC155D    movq    rdx, xmm0
00007FF6C4BC1562    call   printf
00007FF6C4BC1567    mov     rcx, cs:cout@std@3V?basic_ostream@DU?char_traits@D@std@@@1@A ; std::basic_ostream<char, std::char_traits<char>>::operator<<(...>>
00007FF6C4BC156E    lea     rdx, sub_7FF6C4BC1A80
00007FF6C4BC1575    call   cs:??6?basic_ostream@DU?char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; std::basic_ostream<char, std::char_traits<char>>::operator<<(...>>
00007FF6C4BC157B    lea     rcx, Command ; "pause"
00007FF6C4BC1582    call   cs:system
00007FF6C4BC1588    xor     eax, eax
00007FF6C4BC158A    jmp     short loc_7FF6C4BC15C1

```

В случае, если пункты задания можно проиллюстрировать одним снимком экрана дизассемблера (инструкции расположены рядом) - помещать в отчёте один снимок и указывать на нём в графическом редакторе, либо средствами Word, либо комментариями в дизассемблере необходимые по заданию фрагменты.

Прокомментировать в отчёте, на основе каких данных сделаны те или иные заключения и выводы.

Блок 1.3

Изменить копию приложения с помощью патча машинных команд таким образом, чтобы при вводе произвольного ключа:

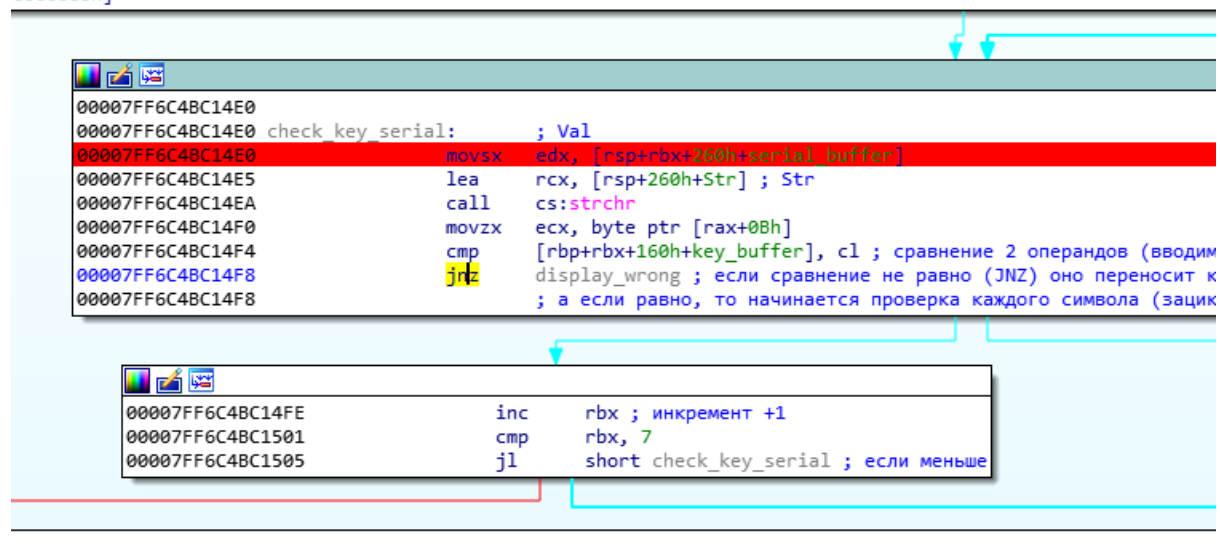
- отображалось сообщение о правильности ключа;
- разблокировалась возможность ввода числового аргумента и получения результата арифметической функции (с момента отображения строки “Type double X:”).

В отчёте приведён:

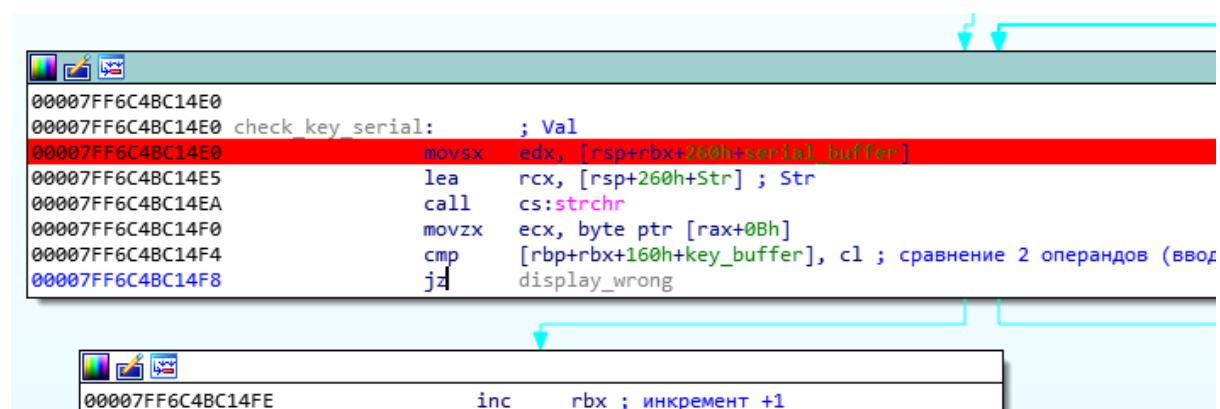
- Снимки фрагмента исправленного кода до и после изменения с пометкой, какие фрагменты кода подвергались изменению.

Было:

0000000h]



Стало:



- Снимки, иллюстрирующие процесс патча в используемом ПО для реверс-инжиниринга (отображены необходимые меню и диалоговые окна).

IDA - Var8_Reverse_Exam — копия.i64 (Var8_Reverse_Exam.exe) C:\Users\anton\Downloads\Var8_Reverse_Exam — копия.i64

File Edit Jump Search View Debugger Options Windows Help

Copy Ctrl+C
Begin selection Alt+L
Select all
Select identifier Shift+Enter
Export data Shift+E

Code C
Data D
Struct var... Alt+Q
Strings
Array... Numpad+
Undefine U
Rename N

Operand type
Comments
Segments
Structs
Functions

Patch program
Other
Plugins

get_f_x
std::_Lockit::~_Lockit(void)
sub_7FF6C4BC1D98
__security_check_cookie
__raise_securityfailure
__report_gsfailure
capture_previous_context
j_i_free
sub_7FF6C4BC1F88
sub_7FF6C4BC1F84

IDA View-A Strings window Hex View-1 Structures

action Data Unexplored External symbol

by, [rbp+160h+key_buffer]
input_key
ax, cs:dword_7FF6C4BC3440 ; 49484746h
xmm0, xmm0
mm1, cs:xmmword_7FF6C4BC3430 ; 30h, 31h, 32h, 33h, 34h, 35h, 36h, 37h, 38h, 39h, :
rsp+260h+var_220], eax
ax, cs:word_7FF6C4BC3444 ; 4Ah, 78h
rsp+260h+var_21C], ax
ax, cs:byte_7FF6C4BC3446
rsp+260h+var_21A], al
rsp+260h+var_240], xmm0
rbp+160h+var_11], bl
mmword ptr [rsp+260h+Str], xmm1
word ptr [rax+rax+00000000h]

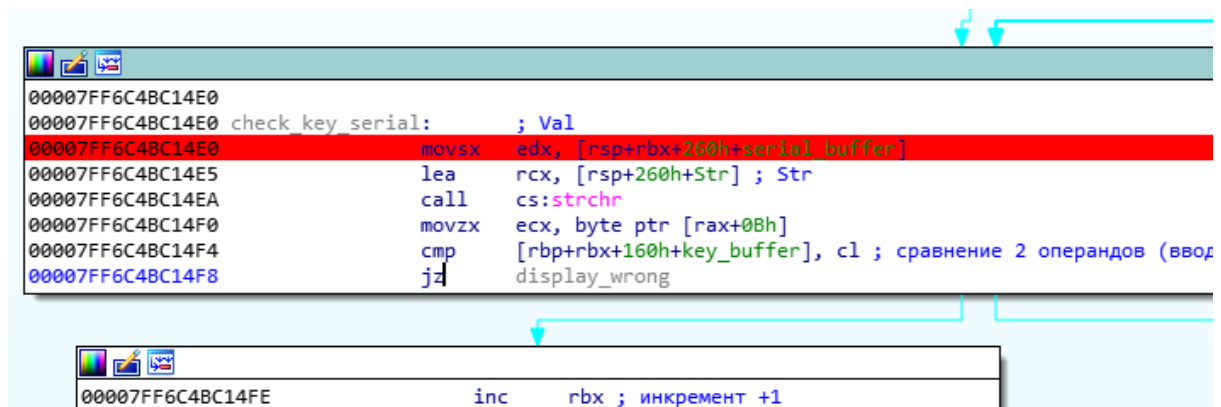
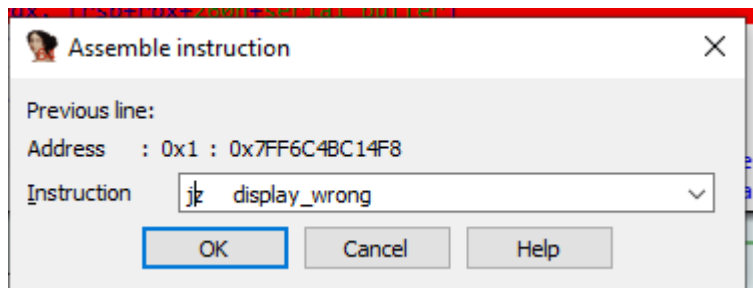
00007FF6C4BC14E0
00007FF6C4BC14E0 check_key_serial: ; Val
00007FF6C4BC14E0 movsx edx, [rsp+rbx+260h+var_21A]
00007FF6C4BC14E5 lea rcx, [rsp+260h+Str]
00007FF6C4BC14EA call cs:strchr
00007FF6C4BC14F0 movzx ecx, byte ptr [rax+0Bh]
00007FF6C4BC14F4 cmp [rbp+rbx+160h+key_buffer], cl ; сравнение 2 операндов (вводим
00007FF6C4BC14F8 jnz display_wrong ; а если равно,
00007FF6C4BC14F8 ; а если равно,

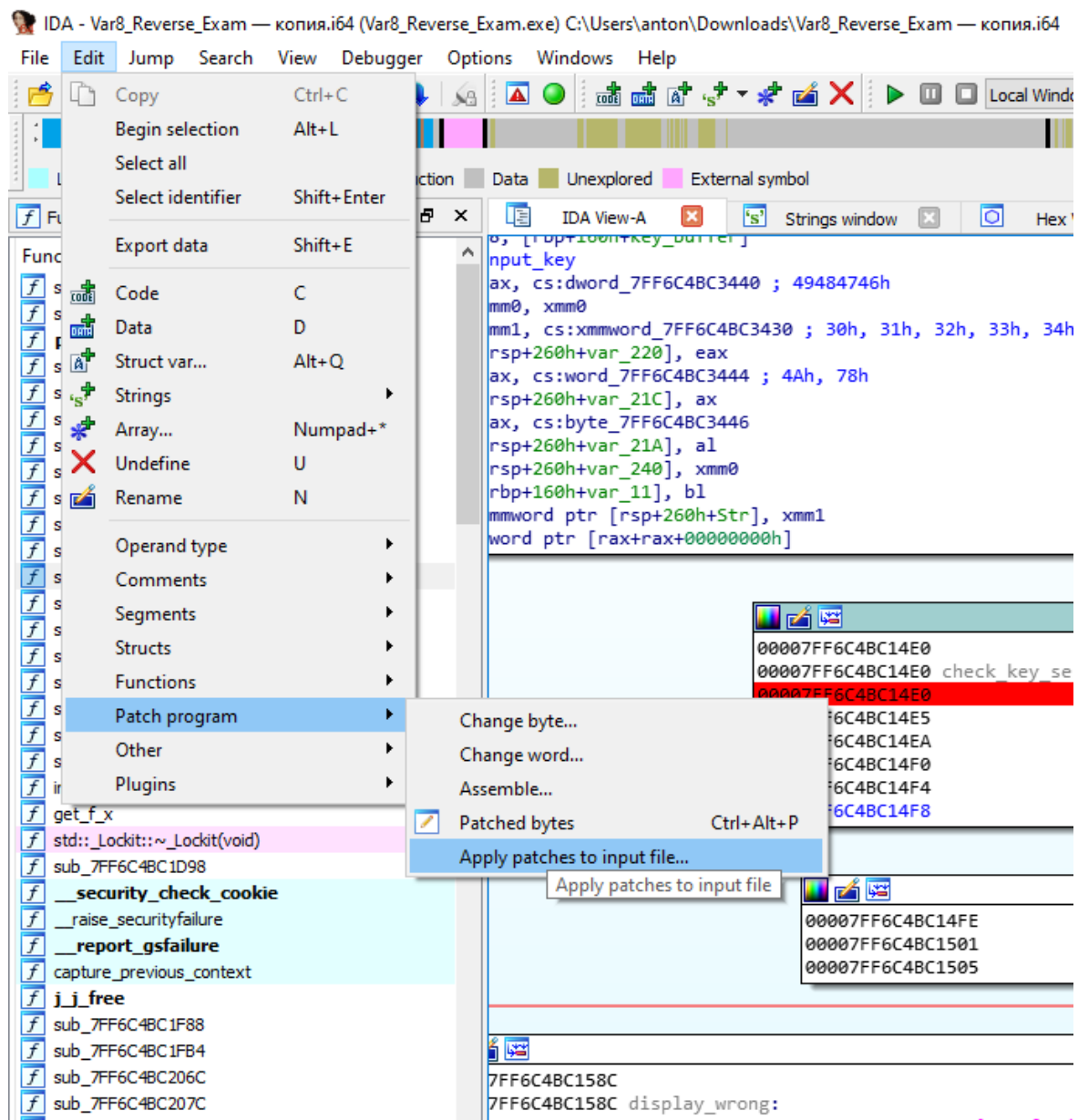
00007FF6C4BC14FE inc rbx ; инкремент +1
00007FF6C4BC1501 cmp rbx, 7
00007FF6C4BC1505 jl short check_key_serial ; если меньше

rsp+260h+var_21A]
00000000h]

00007FF6C4BC14E0
00007FF6C4BC14E0 check_key_serial: ; Val
00007FF6C4BC14E0 movsx edx, [rsp+rbx+260h+serial_buffer]
00007FF6C4BC14E5 lea rcx, [rsp+260h+Str] ; Str
00007FF6C4BC14EA call cs:strchr
00007FF6C4BC14F0 movzx ecx, byte ptr [rax+0Bh]
00007FF6C4BC14F4 cmp [rbp+rbx+160h+key_buffer], cl ; сравнение 2 операндов (вводим
00007FF6C4BC14F8 jnz display_wrong ; если сравнение не равно (JNZ) оно переносит к
00007FF6C4BC14F8 ; а если равно, то начинается проверка каждого символа (зацик

00007FF6C4BC14FE inc rbx ; инкремент +1
00007FF6C4BC1501 cmp rbx, 7
00007FF6C4BC1505 jl short check_key_serial ; если меньше





- Комментарий, почему замена команд производилась именно таким образом.

Замена на противоположную команду.

Работа приложения с произвольным введённым ключом продемонстрирована в виде снимка экрана.


```
-----
Moscow Polytechnic University
Reverse Engineering Exam 2020-2021

Serial: 683-190
Enter key:      111111111
Successful key check!
Type double X: _
```

Блок 2

Блок 2.1

Найден фрагмент кода, в котором производится вычисление математического выражения “Math equation result”, выполнение которой начинается с печати строки “Type double X:” и заканчивается выводом строки “Math equation result”.

```
00007FF6C4BC1507    lea     rcx, aSuccessfulKeyC ; "Successful key check!", display_valid
00007FF6C4BC150E    call   printf
00007FF6C4BC1513    mov     rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_
00007FF6C4BC151A    lea     rdx, sub_7FF6C4BC1A80
00007FF6C4BC1521    call   cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV0
00007FF6C4BC1527    mov     rcx, rcx
00007FF6C4BC152A    lea     rdx, aTypeDoubleX ; "Type double X: "
00007FF6C4BC1531    call   sub_7FF6C4BC16F0
00007FF6C4BC1536    mov     rcx, cs:?cin@std@@3V?$basic_istream@DU?$char_traits@D@std@@@1@A ; std::basic_
00007FF6C4BC153D    lea     rdx, [rsp+260h+var_240]
00007FF6C4BC1542    call   cs:??5?$basic_istream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@AEAV01@AEAV0
00007FF6C4BC1548    movsd   xmm0, [rsp+260h+var_240]
00007FF6C4BC154E    call   get_f_x ; расчёт значения функции "Math equation result"
00007FF6C4BC1553    movaps  xmm1, xmm0
00007FF6C4BC1556    lea     rcx, aMathEquationRe ; "Math equation result:\t%f"
00007FF6C4BC155D    movq    rdx, xmm0
00007FF6C4BC1562    call   printf
00007FF6C4BC1567    mov     rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_
00007FF6C4BC156E    lea     rdx, sub_7FF6C4BC1A80
00007FF6C4BC1575    call   cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV0
00007FF6C4BC157B    lea     rcx, Command ; "pause"
00007FF6C4BC1582    call   cs:system
00007FF6C4BC1588    xor     eax, eax
00007FF6C4BC158A    jmp     short loc_7FF6C4BC15C1
```

В дизассемблере переменной, хранящей аргумент x и результат расчёта присвоены идентификаторы “x” и “math_result”.

Идентификатор X:

Было:

```
var_18= qword ptr -18h
arg_0= qword ptr 8
```

Стало:

```

00007FF6C4BC1D40 get_f_x      proc near
00007FF6C4BC1D40
00007FF6C4BC1D40 log_432.1xX    = qword ptr -18h
00007FF6C4BC1D40 X          = qword ptr 8
00007FF6C4BC1D40

```

Идентификатор math_result:

Было:

```

00007FF6C4BC1507      lea     rcx, aSuccessfulKeyC ; "Successful key check!", display_valid
00007FF6C4BC150E      call    printf
00007FF6C4BC1513      mov     rcx, cs:cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<
00007FF6C4BC151A      lea     rdx, sub_7FF6C4BC1A80
00007FF6C4BC1521      call    cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; s
00007FF6C4BC1527      mov     rcx, rax
00007FF6C4BC152A      lea     rdx, aTypeDoubleX ; "Type double X: "
00007FF6C4BC1531      call    sub_7FF6C4BC16F0
00007FF6C4BC1536      mov     rcx, cs:cin@std@@3V?$basic_istream@DU?$char_traits@D@std@@@1@A ; std::basic_istream<ch
00007FF6C4BC153D      lea     rdx, [rsp+260h+var_240]
00007FF6C4BC1542      call    cs:??5?$basic_istream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@AEAN@Z ; std::basic_istre
00007FF6C4BC1548      movsd   xmm0, [rsp+260h+var_240]
00007FF6C4BC154E      call    get_f_x ; расчёт значения функции "Math equation result"
00007FF6C4BC1553      movaps  xmm1, xmm0 ; процессорная оптимизация
00007FF6C4BC1556      lea     rcx, aMathEquationRe ; "Math equation result:\t%f"
00007FF6C4BC155D      movq    rdx, xmm0
00007FF6C4BC1562      call    printf
00007FF6C4BC1567      mov     rcx, cs:cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<
00007FF6C4BC156E      lea     rdx, sub_7FF6C4BC1A80
00007FF6C4BC1575      call    cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; s
00007FF6C4BC157B      lea     rcx, Command ; "pause"
00007FF6C4BC1582      call    cs:system
00007FF6C4BC1588      xor     eax, eax
00007FF6C4BC158A      jmp     short loc_7FF6C4BC15C1

```

Стало:

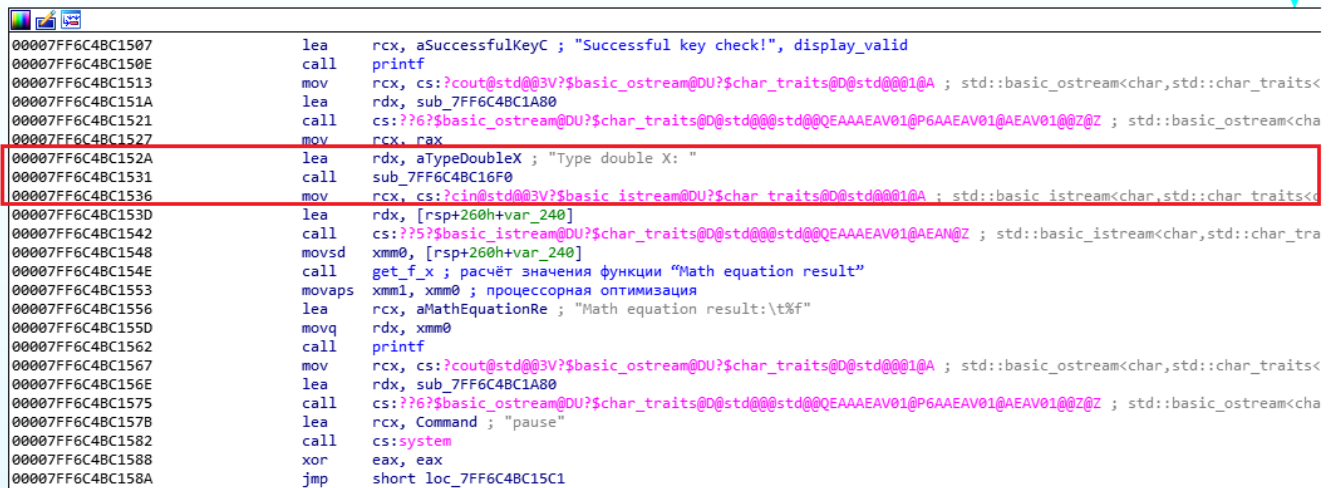
```

00007FF6C4BC1507      lea     rcx, aSuccessfulKeyC ; "Successful key check!", display_valid
00007FF6C4BC150E      call    printf
00007FF6C4BC1513      mov     rcx, cs:cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<cha
00007FF6C4BC151A      lea     rdx, sub_7FF6C4BC1A80
00007FF6C4BC1521      call    cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; std
00007FF6C4BC1527      mov     rcx, rax
00007FF6C4BC152A      lea     rdx, aTypeDoubleX ; "Type double X: "
00007FF6C4BC1531      call    sub_7FF6C4BC16F0
00007FF6C4BC1536      mov     rcx, cs:cin@std@@3V?$basic_istream@DU?$char_traits@D@std@@@1@A ; std::basic_istream<char
00007FF6C4BC153D      lea     rdx, [rsp+260h+var_240]
00007FF6C4BC1542      call    cs:??5?$basic_istream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@AEAN@Z ; std::basic_istream
00007FF6C4BC1548      movsd   math_result, [rsp+260h+var_240]
00007FF6C4BC154E      call    get_f_x ; расчёт значения функции "Math equation result"
00007FF6C4BC1553      movaps  xmm1, math_result ; процессорная оптимизация
00007FF6C4BC1556      lea     rcx, aMathEquationRe ; "Math equation result:\t%f"
00007FF6C4BC155D      movq    rdx, math_result
00007FF6C4BC1562      call    printf
00007FF6C4BC1567      mov     rcx, cs:cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<cha
00007FF6C4BC156E      lea     rdx, sub_7FF6C4BC1A80
00007FF6C4BC1575      call    cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; std
00007FF6C4BC157B      lea     rcx, Command ; "pause"
00007FF6C4BC1582      call    cs:system
00007FF6C4BC1588      xor     eax, eax
00007FF6C4BC158A      jmp     short loc_7FF6C4BC15C1

```

При этом в отчёте указаны (с иллюстрациями):

- Адреса (указать числа) и фрагменты кода (снимок), в котором производится считывание числового аргумента из интерфейса программы в память приложения.



```

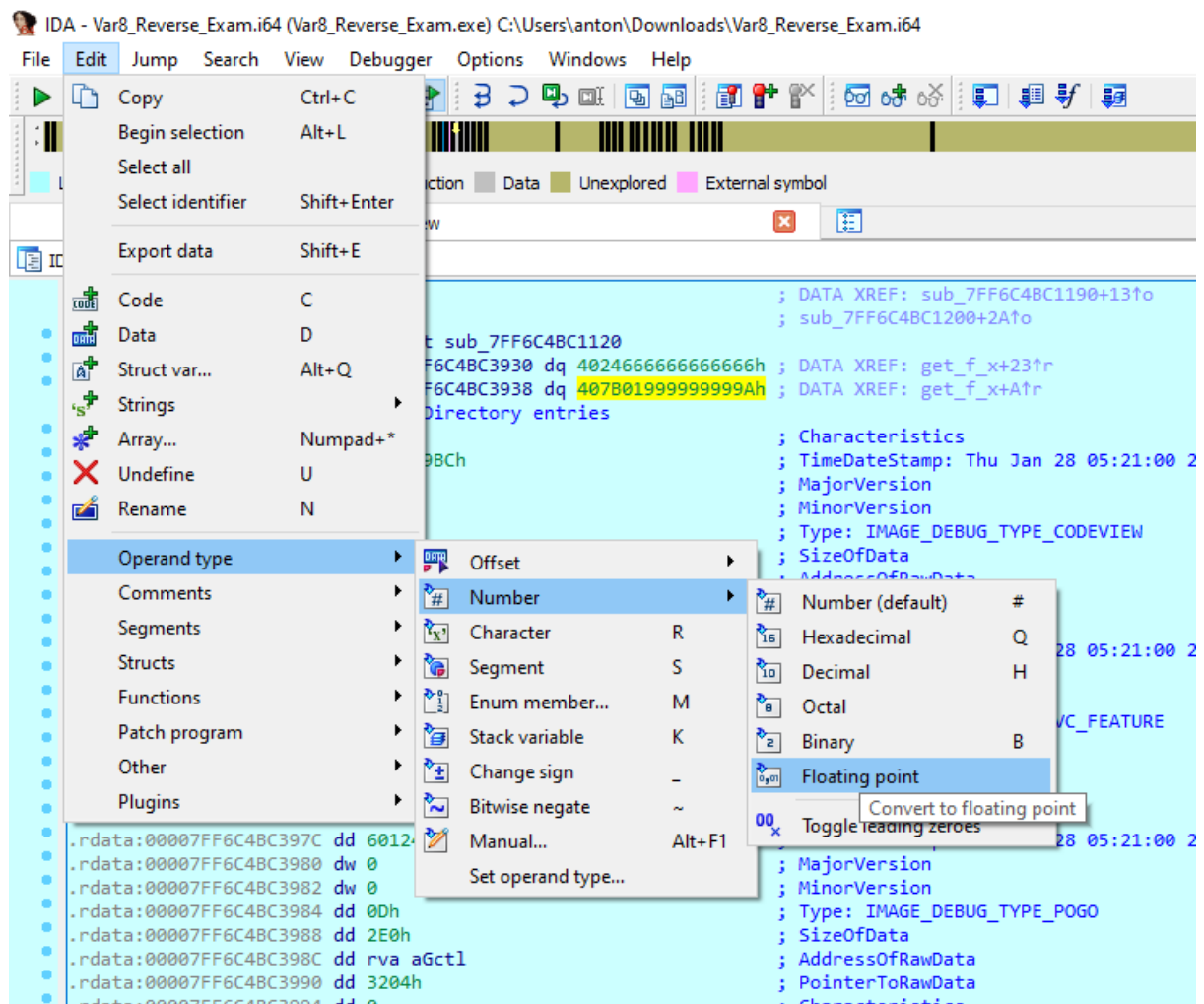
00007FF6C4BC1507    lea     rcx, aSuccessfulKeyC ; "Successful key check!", display_valid
00007FF6C4BC150E    call   printf
00007FF6C4BC1513    mov     rcx, cs:??cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<char,std::char_traits<
00007FF6C4BC151A    lea     rdx, sub_7FF6C4BC1A80
00007FF6C4BC1521    call   cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@Z@Z ; std::basic_ostream<cha
00007FF6C4BC1527    mov     rcx, rcx
00007FF6C4BC152A    lea     rdx, aTypeDoubleX ; "Type double X: "
00007FF6C4BC1531    call   sub_7FF6C4BC16F0
00007FF6C4BC1536    mov     rcx, cs:??cin@std@@3V?$basic_istream@DU?$char_traits@D@std@@@1@A ; std::basic_istream<char,std::char_traits<
00007FF6C4BC153D    lea     rdx, [rsp+260h+var_240]
00007FF6C4BC1542    call   cs:??5?$basic_istream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@AEAN@Z ; std::basic_istream<char,std::char_tra
00007FF6C4BC1548    movsd   xmm0, [rsp+260h+var_240]
00007FF6C4BC154E    call   get_f_x ; расчёт значения функции "Math equation result"
00007FF6C4BC1553    movaps  xmm1, xmm0 ; процессорная оптимизация
00007FF6C4BC1556    lea     rcx, aMathEquationRe ; "Math equation result:\t%f"
00007FF6C4BC155D    movq    rdx, xmm0
00007FF6C4BC1562    call   printf
00007FF6C4BC1567    mov     rcx, cs:??cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::basic_ostream<char,std::char_traits<
00007FF6C4BC156E    lea     rdx, sub_7FF6C4BC1A80
00007FF6C4BC1575    call   cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@Z@Z ; std::basic_ostream<cha
00007FF6C4BC157B    lea     rcx, Command ; "pause"
00007FF6C4BC1582    call   cs:system
00007FF6C4BC1588    xor     eax, eax
00007FF6C4BC158A    jmp     short loc_7FF6C4BC15C1
  
```

- Адреса (указать числа) и фрагменты кода (снимок), по которому получено окончательное значение результата математической функции в памяти программы (которое будет отображено в интерфейсе приложения).

Вышеперечисленные адреса помечены комментариями в дизассемблере.

Блок 2.2

Математический алгоритм записан в отчёте в виде арифметического выражения (формулы) и соответствует варианту. С помощью комментариев в дизассемблере указано, как соотносится каждая машинная команда с записанной формулой (иначе говоря, какая математическая операция из формулы выполняется в строке кода). Снимок экрана дизассемблера, соответствующий коду математического выражения с комментариями приведён в отчёте.



result = exp(log10(X*432.1)/tg(X+10.2))

```

00007FF6C4BC1D40
00007FF6C4BC1D40
00007FF6C4BC1D40
00007FF6C4BC1D40 get_f_x      proc near
00007FF6C4BC1D40
00007FF6C4BC1D40 log_432.1xX    = qword ptr -18h
00007FF6C4BC1D40 X              = qword ptr 8
00007FF6C4BC1D40
00007FF6C4BC1D40      movsd    [rsp+X], xmm0
00007FF6C4BC1D46      sub     rsp, 38h ; вычитание
00007FF6C4BC1D4A      movsd    xmm0, cs:qword_7FF6C4BC3938 ; xmm0 = 432.1
00007FF6C4BC1D52      mulsd    xmm0, [rsp+38h+X] ; (X*432.1)- умножение
00007FF6C4BC1D58      call     log10 ; exp(
00007FF6C4BC1D5D      movsd    [rsp+38h+log_432.1xX], xmm0 ; xmm0 выгружается в переменную
00007FF6C4BC1D63      movsd    xmm1, cs:qword_7FF6C4BC3930 ; xmm1=10.2
00007FF6C4BC1D6B      addsd    xmm1, [rsp+38h+X] ; (X+10.2) - сложение
00007FF6C4BC1D71      movaps    xmm0, xmm1 ; xmm0 = (X+10.2)
00007FF6C4BC1D74      call     tan ; xmm0 = tg(X+10.2)
00007FF6C4BC1D79      movsd    xmm1, [rsp+38h+log_432.1xX] ; xmm1 = log10(X*432.1)
00007FF6C4BC1D7F      divsd    xmm1, xmm0 ; xmm1/xmm0 = log10(X*432.1)/tg(X+10.2)
00007FF6C4BC1D83      movaps    xmm0, xmm1 ; xmm0 = xmm1
00007FF6C4BC1D86      call     exp ; exp(log10(X*432.1)/tg(X+10.2))
00007FF6C4BC1D8B      add     rsp, 38h ; сложение
00007FF6C4BC1D8F      retn     ; xmm0=math_result
00007FF6C4BC1D8F get_f_x      endp
00007FF6C4BC1D8F

```

```

-----
Moscow Polytechnic University
Reverse Engineering Exam 2020-2021

Serial: 648-480
Enter key:      GEIXEIA
Successful key check!
Type double X: 9
Math equation result: 18413.370182
Для продолжения нажмите любую клавишу . . .

```

Блок 3

Блок 3.1

Алгоритм преобразования серийного номера и ключа исследован, описан в отчёте и соответствует варианту. При этом исходный исследуемый РЕ-файл не должен быть модифицирован.

Учащийся продемонстрировал комиссии работу приложения с вводом ключа и выводом сообщения об успешной проверке как минимум трижды.

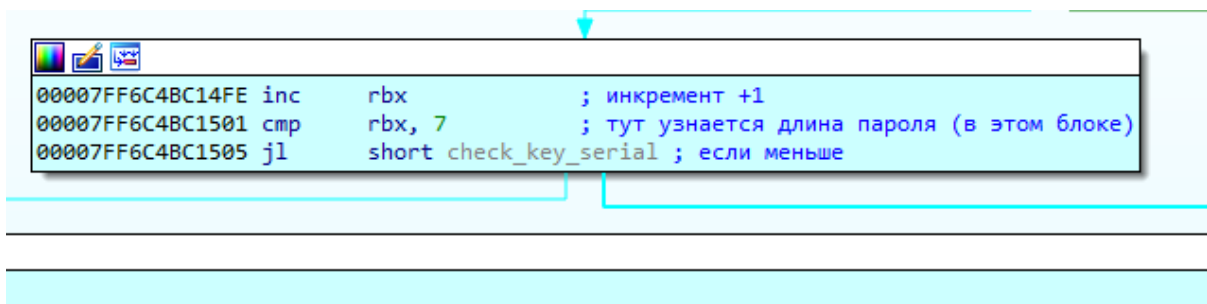
Соответствующие снимки экрана приведены в отчёте.

```

.rdata:00007FF6C4BC3418 ; DATA XREF: sub_7FF6C4BC2900+5to
.rdata:00007FF6C4BC3420 align 10h
.rdata:00007FF6C4BC3430 xmmword_7FF6C4BC3430 db 30h, 31h, 32h, 33h, 34h, 35h, 36h, 37h, 38h, 39h, 20h, 41h, 42h, 43h
.rdata:00007FF6C4BC3430 ; DATA XREF: sub_7FF6C4BC1270+235tr
.rdata:00007FF6C4BC3430 db 44h, 45h
.rdata:00007FF6C4BC3440 dword_7FF6C4BC3440 dd 49484746h ; DATA XREF: sub_7FF6C4BC1270+22Ctr
.rdata:00007FF6C4BC3444 word_7FF6C4BC3444 db 4Ah, 78h ; DATA XREF: sub_7FF6C4BC1270+240tr
.rdata:00007FF6C4BC3446 byte_7FF6C4BC3446 db 0 ; DATA XREF: sub_7FF6C4BC1270+24Ctr
.rdata:00007FF6C4BC3447 align 8
.rdata:00007FF6C4BC3448 aUnknownExcepti db 'Unknown exception',0

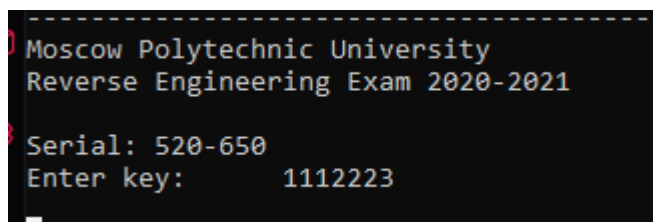
```

Вижу счетчик и в нем 7 => в пароле 7 символов:



```
00007FF6C4BC14FE inc     rbx           ; инкремент +1
00007FF6C4BC1501 cmp     rbx, 7       ; тут узнается длина пароля (в этом блоке)
00007FF6C4BC1505 jl      short check_key_serial ; если меньше
```

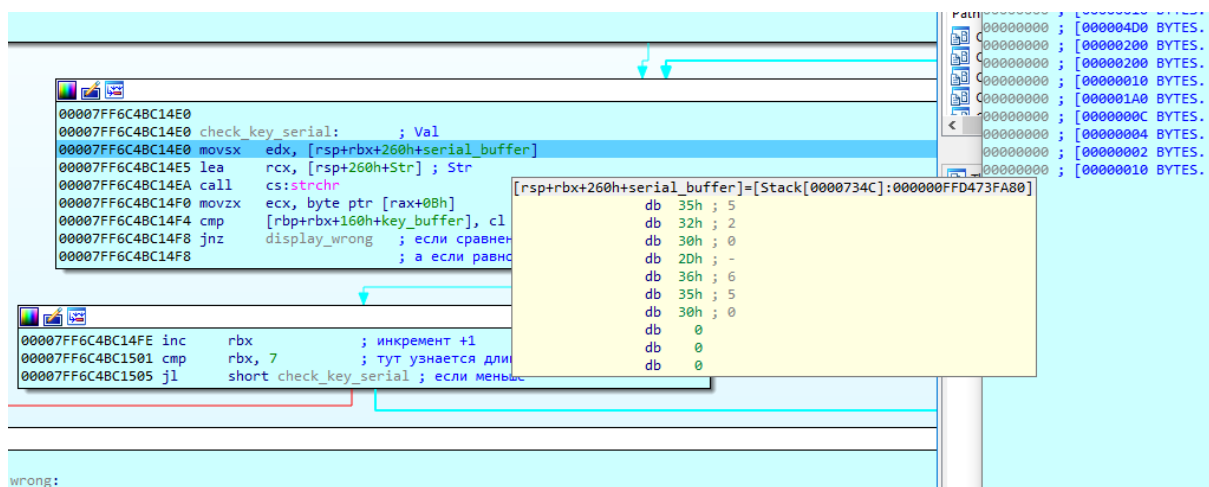
Запускаем отладку и вводим контрастное значение:



```
Moscow Polytechnic University
Reverse Engineering Exam 2020-2021

Serial: 520-650
Enter key:      1112223
```

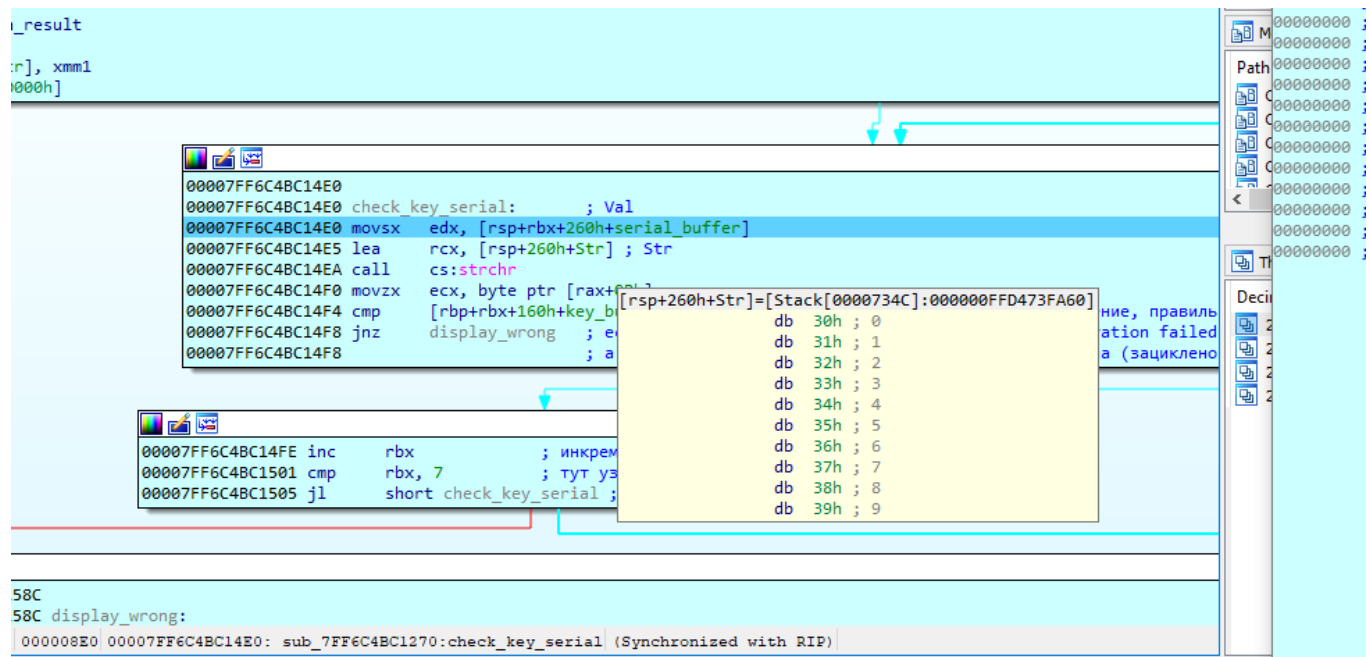
Смотрим check_key_serial:



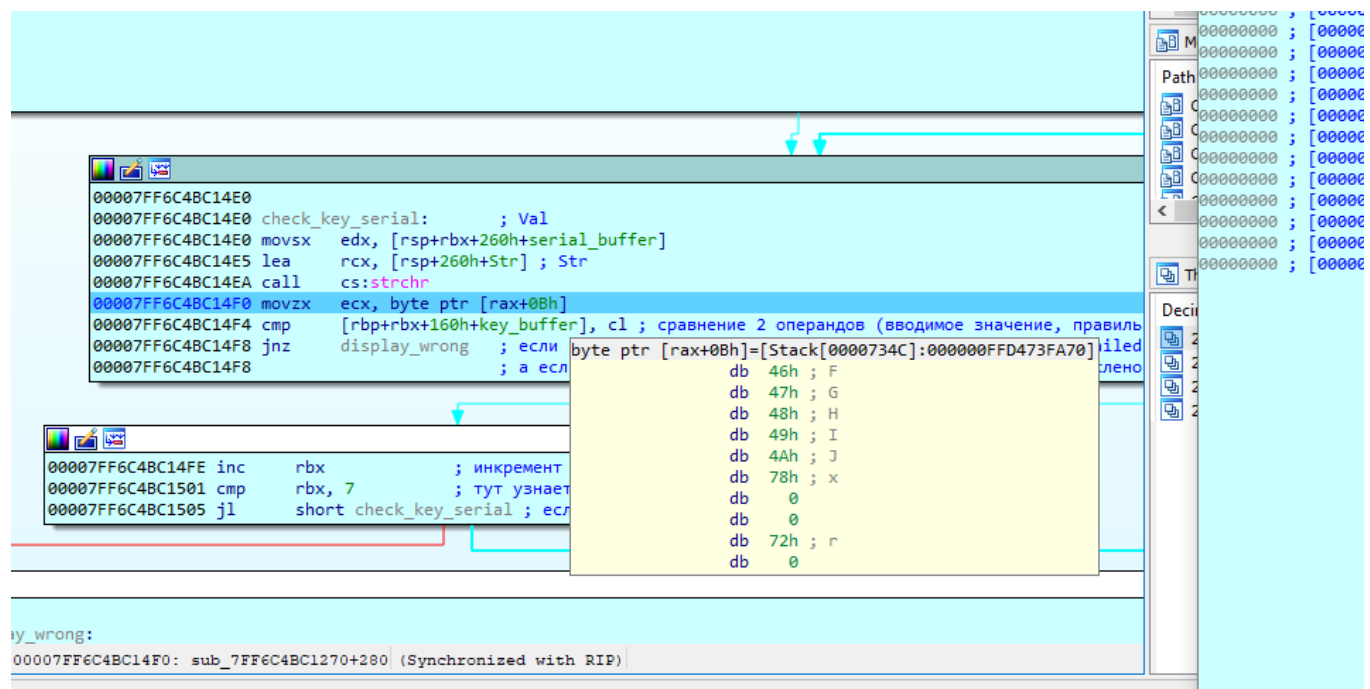
```
00007FF6C4BC14E0 check_key_serial: ; Val
00007FF6C4BC14E0 movsx   edx, [rsp+rbx+260h+serial_buffer]
00007FF6C4BC14E5 lea     rcx, [rsp+260h+Str] ; Str
00007FF6C4BC14EA call    cs:strchr
00007FF6C4BC14F0 movzx   ecx, byte ptr [rax+00h]
00007FF6C4BC14F4 cmp     [rbp+rbx+160h+key_buffer], cl
00007FF6C4BC14F8 jnz     display_wrong ; если сравнен
                                ; а если равен

[rsp+rbx+260h+serial_buffer]=[Stack[0000734C]:00000FFD473FA80]
db 35h ; 5
db 32h ; 2
db 30h ; 0
db 2Dh ; -
db 36h ; 6
db 35h ; 5
db 30h ; 0
db 0
db 0
db 0
```

Используются цифры и их ASCII-символы:



Тут вообще что-то странное, пока пропускаю:



И тут символ сравнивается с cl, значение которого показывает 46:

```

00007FF6C4BC14E0
00007FF6C4BC14E0 check_key_serial: ; Val
00007FF6C4BC14E0 movsx   edx, [rsp+rbx+260h+serial_buffer]
00007FF6C4BC14E5 lea     rcx, [rsp+260h+Str] ; Str
00007FF6C4BC14EA call    cs:strcmp
00007FF6C4BC14F0 movzx   ecx, byte ptr [rax+0Bh]
00007FF6C4BC14F4 cmp     [rbp+rbx+160h+key_buffer], cl ; сравнение 2 операндов (вводимое значение, правильное значение)
00007FF6C4BC14F8 jnz     display_wrong ; если сравнение не равно (JNZ) оно переносит к Activation failed,
00007FF6C4BC14F8 ; а если равно, cl=0000000000000046 проверка каждого символа (зациклено)

```

Смотрю ASCII-таблицу и это F. Первая цифра serial = 5. Предполагаю, что

48	30	0		65	41	A	
49	31	1		66	42	B	
50	32	2		67	43	C	
51	33	3		68	44	D	
52	34	4		69	45	E	
53	35	5		70	46	F	
54	36	6		71	47	G	
55	37	7		72	48	H	
56	38	8		73	49	I	
57	39	9		74	4A	J	

Перезапускаю:

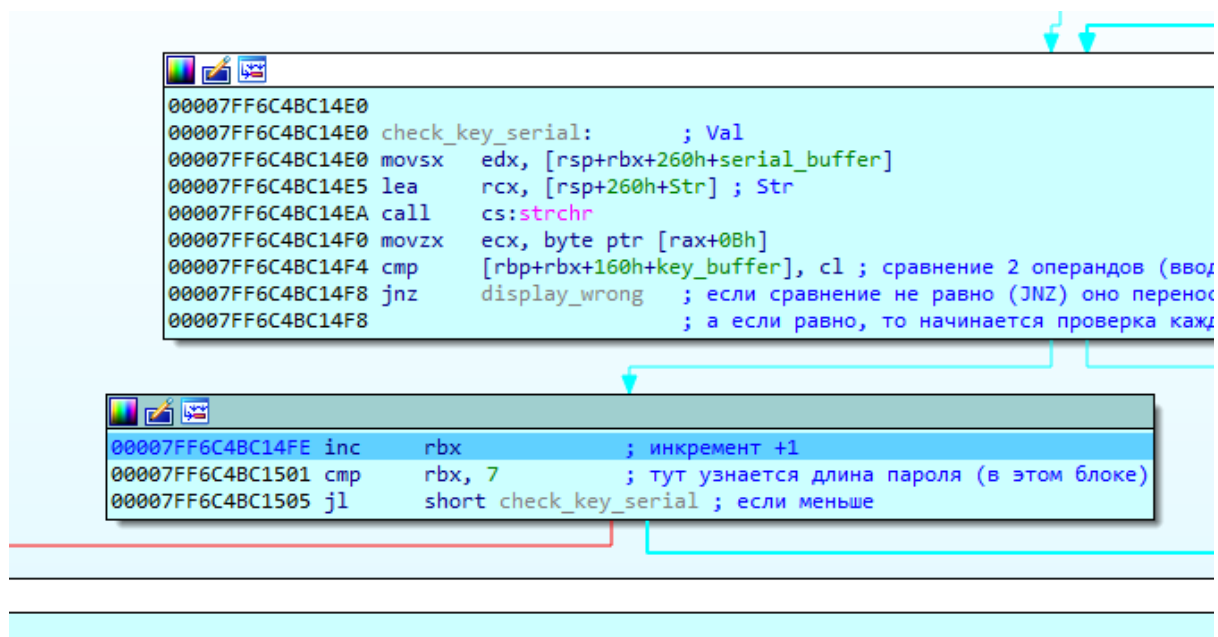
```

Moscow Polytechnic University
Reverse Engineering Exam 2020-2021

Serial: 600-300
Enter key:      G1111111

```

И 1 символ действительно проходит проверку.



Однако нужно как преобразовать такое '-', потому что

45	2D	-		62	3E	>	
----	----	---	--	----	----	---	--

такая замена не сработала.

Обращаюсь опять к:

```

.rdata:00007FF6C4BC3440 dword_7FF6C4BC3440 dd 49484746h
.rdata:00007FF6C4BC3444 word_7FF6C4BC3444 db 4Ah, 78h
.rdata:00007FF6C4BC3446 byte_7FF6C4BC3446 db 0

```

, смотрю в

ASCII-таблицу

120	78	x	
-----	----	---	--

Это x (другие пробовала, но не подошло).

Пробую с x и все выходит.

Алгоритм: key - это ASCII-символы шестнадцатеричного serial со сдвигом 11h для цифр, и со сдвигом 4Bh для '-'. Каждая цифра извлекается отдельно.

```

-----
Moscow Polytechnic University
Reverse Engineering Exam 2020-2021

Serial: 359-520
Enter key:      DFJxFCA
Successful key check!

```

```
-----  
Moscow Polytechnic University  
Reverse Engineering Exam 2020-2021
```

```
Serial: 384-510  
Enter key:      DIExFBA  
Successful key check!  
Type double X: _
```

```
-----  
Moscow Polytechnic University  
Reverse Engineering Exam 2020-2021
```

```
Serial: 388-800  
Enter key:      DIIxIAA  
Successful key check!  
Type double X:
```