

# Ank83's KeyGenMe #2 - Keygen'd

Author: Napalm  
Tools: IDA and a Brain

## Introduction

A nice and easy DOS Keygen. After full analysis with IDA we can see its created with C++. i.e.) Standard cin/cout usage. Again with the last one UPX 1.93 was used so you will need to download this version from upx.sf.net and decompress the exe with the -d option.

## Serial Generating Code

Here is a IDA dump of a this segment of the code full with comments. I've removed some of the junk lines not needed to understand the serial generation.

```
00401145 ; -----
00401145      push    offset "\n So let's begin by entering ID (0...999): ";
0040114F      call    ostream::operator<<(char const *)
00401154      lea    eax, [ebp+nID]
00401157      push    eax                ; push a pointer to nID for cin to read into
0040115D      call    istream::operator>>(int &)
00401162      mov    ecx, [ebp+nID]      ; \
00401165      add    ecx, 1              ; > nID++;
00401168      mov    [ebp+nID], ecx     ; /
0040116B      mov    edx, [ebp+nID]     ; \
0040116E      add    edx, [ebp+nID]     ; > nID *= 2;
00401171      mov    [ebp+nID], edx     ; /
00401174      mov    eax, [ebp+nID]     ; \
00401177      sub    eax, 1              ; > nID -= 1;
0040117A      mov    [ebp+nID], eax     ; /
0040117D      mov    ecx, [ebp+nID]     ; \
00401180      imul  ecx, [ebp+nID]     ; > nID *= nID;
00401184      mov    [ebp+nID], ecx     ; /
00401187      mov    [ebp+nCount], 1    ; Set loop counter to 1
0040118E      jmp    short local_loopstart
00401190 ; -----
00401190 local_loopreturn:
00401190      mov    edx, [ebp+nCount]  ; \
00401193      add    edx, 1              ; > nCount++;
00401196      mov    [ebp+nCount], edx ; /
00401199
00401199 local_loopstart:
00401199      cmp    [ebp+nCount], 20   ; \_ if(nCount >= 20) goto local_loopend
0040119D      jge    short local_loopend ; /
0040119F      mov    eax, [ebp+nID]     ; \
004011A2      sub    eax, [ebp+nCount]  ; > nID -= nCount;
004011A5      mov    [ebp+nID], eax     ; /
004011A8      jmp    short local_loopreturn
004011AA
004011AA ; -----
004011AA local_loopend:
004011AA      push    offset "\n Now enter the serial: ";
004011B4      call    ostream::operator<<(char const *)
004011B9      lea    ecx, [ebp+nSerial]
004011BC      push    ecx                ; push a pointer to nSerial for cin to read
into
004011C2      call    istream::operator>>(int &)
004011C7      mov    edx, [ebp+nID]     ; \
004011CA      cmp    edx, [ebp+nSerial] ; > if(nID != nSerial) goto serial_incorrect
004011CD      jnz    short serial_incorrect ; /
004011CF      push    offset "\n\n      Wow ! You did it. Congratulation.\n";
004011D9      call    ostream::operator<<(char const *)
004011DE ; -----
004011DE serial_incorrect:
004011DE      mov    eax, [ebp+nID]     ; \
004011E1      cmp    eax, [ebp+nSerial] ; > if(nID == nSerial) goto main_exit
004011E4      jz     short main_exit    ; /
004011E6      push    offset "\n\n      Please try again !\n";
004011F0      call    ostream::operator<<(char const *)
004011F5
004011F5 main_exit:
004011F5      lea    ecx, [ebp+nIgnored]
004011F8      push    ecx                ; push a pointer to nIgnored for cin to read into
004011FE      call    istream::operator>>(int &)
```

Hopefully my cheap ascii-art and comments have helped you understand it..  
Heres the way we could rejoin this all in to a high level language like C++.

```
#include <iostream>
using namespace std;

int nID, nCount, nSerial, nIgnored;
cout << "\n So let's begin by entering ID (0...999): ";
cin >> nID;
nID = (((nID + 1) * 2) - 1);
nID *= nID;
for(nCount = 1; nCount < 20; nCount++) nID -= nCount;
cout << "\n Now enter the serial: ";
cin >> nSerial;
if(nID == nSerial) {
    cout << "\n\n      Wow ! You did it. Congratulation";
} else {
    cout << "\n\n      Please try again ! It's";
}
cin >> nIgnored;
```

## Solution

So as a keygen goes we can simply do this.. or we can patch the keygenme to output a valid serial.

```
#include <iostream>
using namespace std;

int nID, nCount, nIgnored;
cout << "\n Enter ID (0...999): ";
cin >> nID;
nID = (((nID + 1) * 2) - 1);
nID *= nID;
for(nCount = 1; nCount < 20; nCount++) nID -= nCount;
cout << "\n Serial Code: " << nID;
cin >> nIgnored;
```

Or you can patch the following addresses.

004011AA	push	offset "\n Now enter the serial: ";
004011B4	call	ostream::operator<<(char const *)
004011B9	lea	ecx, [ebp+nSerial]

We can change the push to push the stack offset of *nID* and then change the call to *ostream::operator<<(int &)* and the since "*lea ecx, [ebp+X]*" is 5 bytes there's plenty of room to make a short jump to *main\_exit*.

## Final Words

Many thanks to Ank83 for a great beginners keygenme. I hope this text file will help many people.  
Best of luck to all!

**Napalm**  
Team ICU