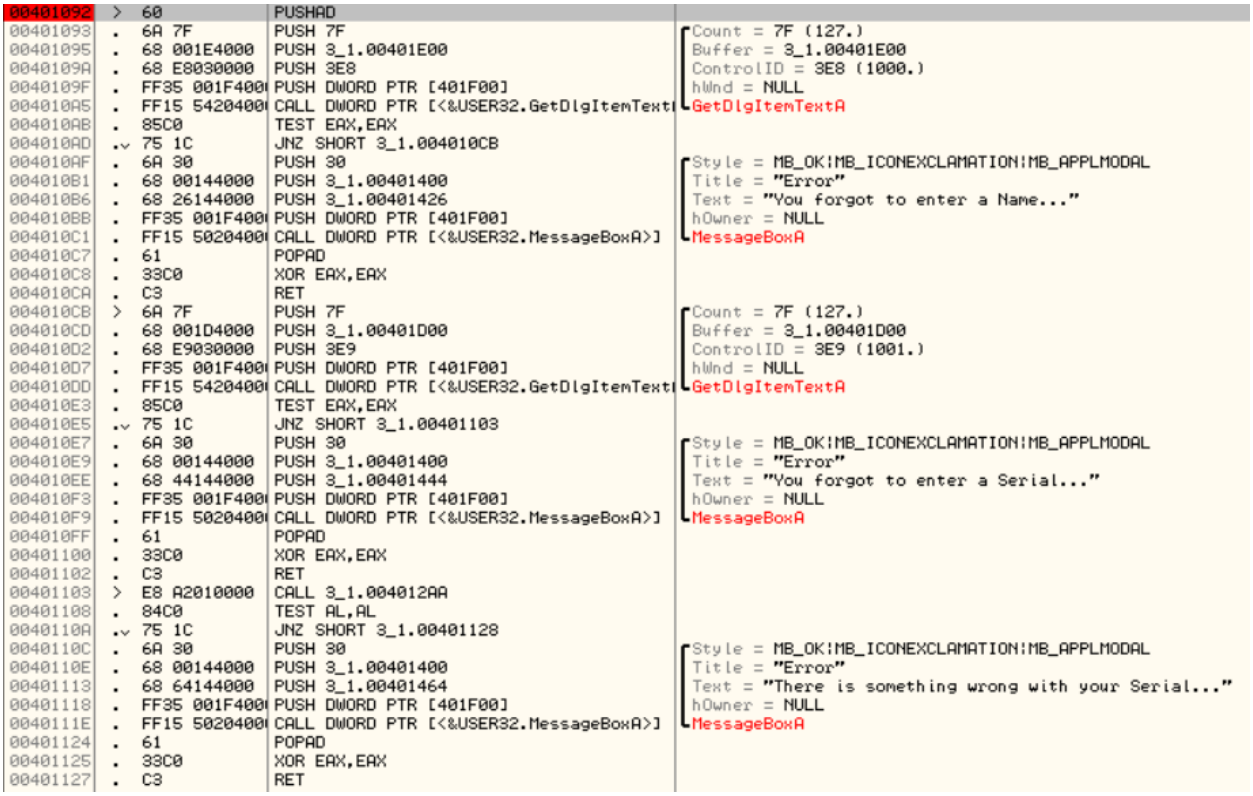
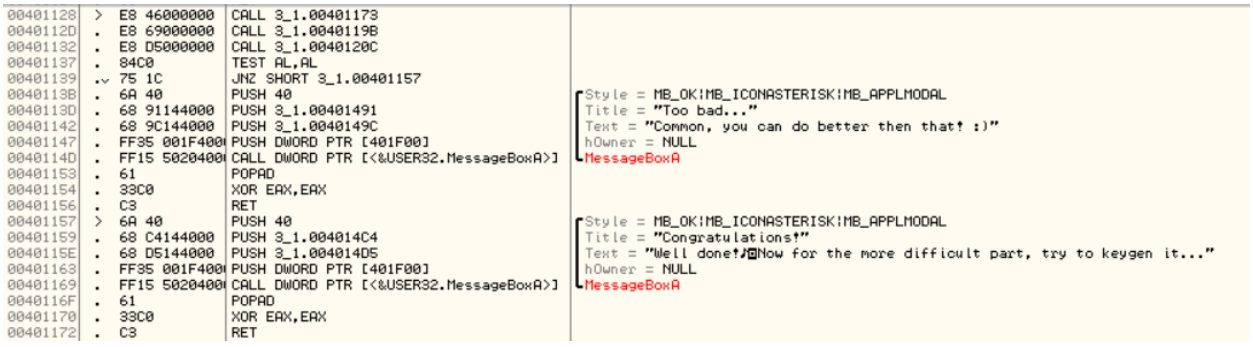
Câu 3.1.exe

-Từ 00401092 đến 00401172 là nơi chứa tất cả các hàm và lệnh thực hiện các công việc nhập xuất và xử lý name và serial.





004010A5 . FF15 54204000 CALL DWORD PTR [<&USER32.GetDlgItemTextA>; \GetDlgItemTextA

=> Nhập name

004010C1 . FF15 50204000 CALL DWORD PTR [<&USER32.MessageBoxA>] ; \MessageBoxA

=> Báo lỗi nếu name rỗng

004010DD . FF15 54204000 CALL DWORD PTR [<&USER32.GetDlgItemTextA>; \GetDlgItemTextA

=> Nhập serial

004010F9 . FF15 50204000 CALL DWORD PTR [<&USER32.MessageBoxA>] ; \MessageBoxA

=> Báo lỗi nếu serial rỗng

Tại 00401103 > E8 A2010000 CALL 3_1.004012AA

```
004012AA  $  50            PUSH EAX
004012AB  .  56            PUSH ESI
004012AC  .  BE 00174000   MOV ESI,3_1.00401700
004012B1  >  8B06          MOV EAX,DWORD PTR [ESI]
004012B3  .  8326 00       AND DWORD PTR [ESI],0
004012B6  .  83C6 04       ADD ESI,4
004012B9  .  85C0          TEST EAX,EAX
004012BB  .^ 75 F4         JNZ SHORT 3_1.004012B1
004012BD  .  BE 001D4000   MOV ESI,3_1.00401D00
004012C2  >  8A06          MOV AL,BYTE PTR [ESI]
004012C4  .  84C0          TEST AL,AL
004012C6  .v 74 18         JE SHORT 3_1.004012E0
004012C8  .  3C 30         CMP AL,30
004012CA  .v 72 0C         JB SHORT 3_1.004012D8
004012CC  .  3C 3A         CMP AL,3A
004012CE  .v 72 0D         JB SHORT 3_1.004012DD
004012D0  .  3C 41         CMP AL,41
004012D2  .v 72 04         JB SHORT 3_1.004012D8
004012D4  .  3C 47         CMP AL,47
004012D6  .v 72 05         JB SHORT 3_1.004012DD
004012D8  >  5E            POP ESI
004012D9  .  58            POP EAX
004012DA  .  B0 00         MOV AL,0
004012DC  .  C3            RET
004012DD  >  46            INC ESI
004012DE  .^ EB E2         JMP SHORT 3_1.004012C2
004012E0  >  5E            POP ESI
004012E1  .  58            POP EAX
004012E2  .  B0 01         MOV AL,1
004012E4  L. C3            RET
```

=> Kiểm tra các ký tự trong serial có phải là một trong các ký tự: ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘A’, ‘B’, ‘C’, ‘D’, ‘E’, ‘F’ hay không?

Nếu có ký tự không thuộc các ký tự trên thì báo lỗi:

```
0040110C  .  6A 30         PUSH 30                                  Style = MB_OK|MB_ICONEXCLAMATION|MB_APPLMODAL
0040110E  .  68 00144000   PUSH 3_1.00401400                          Title = "Error"
00401113  .  68 64144000   PUSH 3_1.00401464                        Text = "There is something wrong with your Serial..."
00401118  .  FF35 001F400( PUSH DWORD PTR [401F00]                  hOwner = NULL
0040111E  .  FF15 5020400( CALL DWORD PTR [<&USER32.MessageBoxA>]   MessageBoxA
```

Còn nếu quá trình nhập name và serial thuận lợi thì ta sẽ có good boy và bad boy là:

-Good boy

```
00401157  >  6A 40         PUSH 40                                  Style = MB_OK|MB_ICONASTERISK|MB_APPLMODAL
00401159  .  68 C4144000   PUSH 3_1.004014C4                        Title = "Congratulations!"
0040115E  .  68 D5144000   PUSH 3_1.004014D5                        Text = "Well done!♪■Now for the more difficult part, try to keygen it..."
00401163  .  FF35 001F400( PUSH DWORD PTR [401F00]                  hOwner = NULL
00401169  .  FF15 5020400( CALL DWORD PTR [<&USER32.MessageBoxA>]   MessageBoxA
```

-Bad boy

```
0040113B  .  6A 40         PUSH 40                                  Style = MB_OK|MB_ICONASTERISK|MB_APPLMODAL
0040113D  .  68 91144000   PUSH 3_1.00401491                       Title = "Too bad..."
00401142  .  68 9C144000   PUSH 3_1.0040149C                       Text = "Common, you can do better then that! :)"
00401147  .  FF35 001F400( PUSH DWORD PTR [401F00]                  hOwner = NULL
0040114D  .  FF15 5020400( CALL DWORD PTR [<&USER32.MessageBoxA>]   MessageBoxA
```

-Ta xét 3 hàm quan trọng:

```
00401128  >  E8 46000000   CALL 3_1.00401173
0040112D  .  E8 69000000   CALL 3_1.0040119B
00401132  .  E8 D5000000   CALL 3_1.0040120C
```

-Đầu tiên, 00401128 > E8 46000000 CALL 3_1.00401173 là hàm tạo ma trận 16x16 và 2 hàng rào chắn

```
00401173  $  50            PUSH EAX
00401174  .  51            PUSH ECX
00401175  .  57            PUSH EDI
00401176  .  BF F01A4000   MOV EDI,3_1.00401AF0
0040117B  .  FC            CLD
0040117C  .  B9 10000000   MOV ECX,10
00401181  .  B0 FF         MOV AL,0FF
00401183  .  F3:AA         REP STOS BYTE PTR ES:[EDI]
00401185  .  B9 00010000   MOV ECX,100
0040118A  .  B0 00         MOV AL,0
0040118C  .  F3:AA         REP STOS BYTE PTR ES:[EDI]
0040118E  .  B9 10000000   MOV ECX,10
00401193  .  B0 FF         MOV AL,0FF
00401195  .  F3:AA         REP STOS BYTE PTR ES:[EDI]
00401197  .  5F            POP EDI
00401198  .  59            POP ECX
00401199  .  58            POP EAX
0040119A  .  C3            RET
```

00401183 |. F3:AA REP STOS BYTE PTR ES:[EDI]

⇨ Đây là một dạng như memset trong C++, lưu EAX vào nơi EDI trỏ đến.

0040117C |. B9 10000000 MOV ECX,10

00401181 |. B0 FF MOV AL,0FF

00401183 |. F3:AA REP STOS BYTE PTR ES:[EDI]

⇨ Cho 16 ô của hàng đầu mang giá trị FF

00401185 |. B9 00010000 MOV ECX,100 ; 256 bytes - 16x16 grid

0040118A |. B0 00 MOV AL,0

0040118C |. F3:AA REP STOS BYTE PTR ES:[EDI]

⇨ Cho liên tục 16 ô của 16 hàng kế tiếp là 00 (16x16=256=100h)

0040118E |. B9 10000000 MOV ECX,10

00401193 |. B0 FF MOV AL,0FF

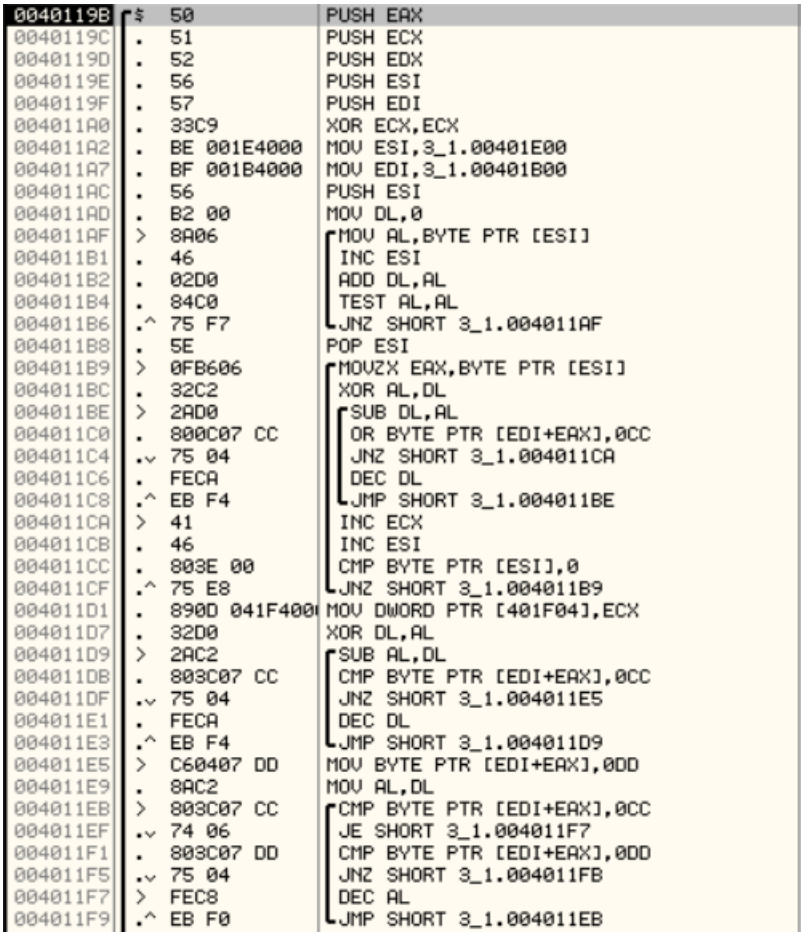
00401195 |. F3:AA REP STOS BYTE PTR ES:[EDI]

⇨ Cho 16 ô của hàng cuối mang giá trị FF

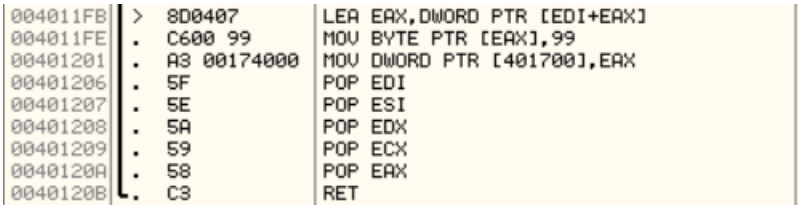
Ta có dạng ma trận sau khi gọi hàm như sau:

```
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

-Tiếp theo, 0040112D . E8 69000000 CALL 3_1.0040119B: là hàm xử lý name để phát sinh các giá trị cần thiết cho ma trận.



```
0040119B  $  50            PUSH EAX
0040119C  .  51            PUSH ECX
0040119D  .  52            PUSH EDX
0040119E  .  56            PUSH ESI
0040119F  .  57            PUSH EDI
004011A0  .  33C9          XOR ECX,ECX
004011A2  .  BE 001E4000   MOV ESI,3_1.00401E00
004011A7  .  BF 001B4000   MOV EDI,3_1.00401B00
004011AC  .  56            PUSH ESI
004011AD  .  B2 00         MOV DL,0
004011AF  >  8A06          MOV AL,BYTE PTR [ESI]
004011B1  .  46            INC ESI
004011B2  .  02D0          ADD DL,AL
004011B4  .  84C0          TEST AL,AL
004011B6  .^ 75 F7         JNZ SHORT 3_1.004011AF
004011B8  .  5E            POP ESI
004011B9  >  0FB606        MOVZX EAX,BYTE PTR [ESI]
004011BC  .  32C2          XOR AL,DL
004011BE  >  2AD0          SUB DL,AL
004011C0  .  800C07 CC     OR BYTE PTR [EDI+EAX],0CC
004011C4  .v 75 04         JNZ SHORT 3_1.004011CA
004011C6  .  FECA          DEC DL
004011C8  .^ EB F4         JMP SHORT 3_1.004011BE
004011CA  >  41            INC ECX
004011CB  .  46            INC ESI
004011CC  .  803E 00       CMP BYTE PTR [ESI],0
004011CF  .^ 75 E8         JNZ SHORT 3_1.004011B9
004011D1  .  890D 041F400  MOV DWORD PTR [401F04],ECX
004011D7  .  32D0          XOR DL,AL
004011D9  >  2AC2          SUB AL,DL
004011DB  .  803C07 CC     CMP BYTE PTR [EDI+EAX],0CC
004011DF  .v 75 04         JNZ SHORT 3_1.004011E5
004011E1  .  FECA          DEC DL
004011E3  .^ EB F4         JMP SHORT 3_1.004011D9
004011E5  >  C60407 DD     MOV BYTE PTR [EDI+EAX],0DD
004011E9  .  8AC2          MOV AL,DL
004011EB  >  803C07 CC     CMP BYTE PTR [EDI+EAX],0CC
004011EF  .v 74 06         JE SHORT 3_1.004011F7
004011F1  .  803C07 DD     CMP BYTE PTR [EDI+EAX],0DD
004011F5  .v 75 04         JNZ SHORT 3_1.004011FB
004011F7  >  FEC8          DEC AL
004011F9  .^ EB F0         JMP SHORT 3_1.004011EB
```



```
004011FB  >  8D0407        LEA EAX,DWORD PTR [EDI+EAX]
004011FE  .  C600 99       MOV BYTE PTR [EAX],99
00401201  .  A3 00174000   MOV DWORD PTR [401700],EAX
00401206  .  5F            POP EDI
00401207  .  5E            POP ESI
00401208  .  5A            POP EDX
00401209  .  59            POP ECX
0040120A  .  58            POP EAX
0040120B  .  C3            RET
```

Ta xét các bước xử lý đáng chú ý của hàm:

004011A2 |. BE 001E4000 MOV ESI,Snake.00401E00 ;

⇨ gán name vào ESI

004011A7 |. BF 001B4000 MOV EDI,Snake.00401B00 ;

⇨ gán ma trận 16x16 vào EDI

004011C0 |. 800C07 CC ||OR BYTE PTR DS:[EDI+EAX],0CC ;

⇨ mỗi ký tự trong name được ánh xạ tới một vị trí nào đó trong ma trận và thể hiện bằng “CC”

004011E5 |> C60407 DD MOV BYTE PTR DS:[EDI+EAX],0DD ;

⇨ "DD" đại diện cho điểm kết thúc

004011FE |. C600 99 MOV BYTE PTR DS:[EAX],99 ;

⇨ "99" đại diện cho Snake

Thực chất, đây là một dạng mô phỏng lại game Snake.

Dưới đây là code C++ mô tả lại hàm phát sinh ma trận và các giá trị cần thiết của nó:

```
void generate()
{
    vector<string> temp;
    for (int i = 0; i < 256; i++)
    {
        temp.push_back("00");
    }
    string s = textToHex(name);
    QInt x("00", "16");
    QInt y("00", "16");
    string AL = "", DL = "00";
    for (int i = 0; i < 2 * name.length(); i += 2)
    {
        AL = s.substr(i, 2);
        x.setData(AL, "16");
        y.setData(DL, "16");
        DL = (y + x).toString("16");
        DL = getLastChars(DL, 1);
    }
```

```
    int pos;
    for (int i = 0; i < 2 * name.length(); i += 2)
    {
        AL = s.substr(i, 2);
        x.setData(AL, "16");
        y.setData(DL, "16");
        AL = (x ^ y).toString("16");
        AL = getLastChars(AL, 1);

        Label0:
        x.setData(AL, "16");
        y.setData(DL, "16");
        DL = (y - x).toString("16");
        DL = getLastChars(DL, 1);

        x.setData(AL, "16");
        pos = atoi(x.toString("10").c_str());

        if (temp[pos] == "CC")
        {
            y.setData(DL, "16");
            x.setData("01", "16");
            DL = (y - x).toString("16");
            DL = getLastChars(DL, 1);
            goto Label0;
        }

        }
        else temp[pos] = "CC";
    }
    x.setData(AL, "16");
    y.setData(DL, "16");
    DL = getLastChars((y^x).toString("16"), 1);

Label1:
    x.setData(AL, "16");
    y.setData(DL, "16");
    AL = getLastChars((x - y).toString("16"), 1);
    x.setData(AL, "16");
    pos = atoi(x.toString("10").c_str());

    if (temp[pos] == "CC")
    {
        y.setData(DL, "16");
        x.setData("01", "16");
        DL = getLastChars((y - x).toString("16"), 1);
        goto Label1;
    }
```

```
    x.setData(AL, "16");
    pos = atoi(x.toString("10").c_str());

    temp[pos] = "DD";
    AL = DL;

Label2:
    x.setData(AL, "16");
    pos = atoi(x.toString("10").c_str());

    if (temp[pos] == "CC" || temp[pos] == "DD")
    {
        x.setData(AL, "16");
        y.setData("01", "16");
        AL = getLastChars((x - y).toString("16"), 1);
        goto Label2;
    }

    x.setData(AL, "16");
    pos = atoi(x.toString("10").c_str());

    temp[pos] = "99";

    int i = 0, j = 0;
    for (int k = 0; k < 256; k++)
    {
        if (temp[k] == "DD")
        {
            dest.x = i;
            dest.y = j;
            grid[i][j] = 'D';
        }
        if (temp[k] == "CC")
        {
            Point2D food;
            food.x = i;
            food.y = j;
            foods.push_back(food);
            grid[i][j] = 'C';
        }
        if (temp[k] == "99")
        {
            snake.x = i;
            snake.y = j;
            grid[i][j] = '9';
        }
        j++;
```

```
        if ((k+1) % 16 == 0)
        {
            j = 0;
            i++;
        }
    }
}
```

Đoạn code đơn giản là sử dụng QInt để mô tả lại quá trình tính toán và phát sinh ma trận. Sau đó lưu vào một ma trận để xử lý về sau. Để việc lưu ma trận đơn giản, “FF” -> ‘F’, “00” -> ‘0’, “CC” -> ‘C’, ”DD” -> ‘D’, “99” -> ‘9’.

Giả sử name nhập vào là: abcde

Ta có ma trận mô phỏng như sau:

```
F F F F F F F F F F F F F F F F
0 0 0 C 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 D
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 C 0 0
0 0 0 0 0 C 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 C 0 0
0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 C 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
F F F F F F F F F F F F F F F F
```

-Cuối cùng, 00401132 . E8 D5000000 CALL 3_1.0040120C là hàm kiểm tra serial. Thực chất serial là một chuỗi các ký tự hướng dẫn đường đi cho snake để có thể đi đến đích cuối cùng.

```
0040120C  $  60              PUSHAD
0040120D  .  BE 001D4000     MOV ESI,3_1.00401D00
00401212  .  BF 00174000     MOV EDI,3_1.00401700
00401217  >  0FB606          MOVZX EAX,BYTE PTR [ESI]
0040121A  .  84C0            TEST AL,AL
0040121C  .v 74 42           JE SHORT 3_1.00401260
0040121E  .  2C 30           SUB AL,30
00401220  .  3C 0A           CMP AL,0A
00401222  .v 72 02           JB SHORT 3_1.00401226
00401224  .  2C 07           SUB AL,7
00401226  >  8BC8            MOV ECX,EAX
00401228  .  24 03           AND AL,3
0040122A  .  C0E9 02         SHR CL,2
0040122D  .  BA 10000000     MOV EDX,10
00401232  .  84C0            TEST AL,AL
00401234  .v 74 0B           JE SHORT 3_1.00401241
00401236  .  48              DEC EAX
00401237  .v 74 06           JE SHORT 3_1.0040123F
00401239  .  C1EA 04         SHR EDX,4
0040123C  .  48              DEC EAX
0040123D  .v 75 02           JNZ SHORT 3_1.00401241
0040123F  >  F7DA            NEG EDX
00401241  >  8B07            MOV EAX,DWORD PTR [EDI]
00401243  .  03C2            ADD EAX,EDX
00401245  .  8A00            MOV AL,BYTE PTR [EAX]
00401247  .  84C0            TEST AL,AL
00401249  .v 74 2F           JE SHORT 3_1.0040127A
0040124B  .  3C 99           CMP AL,99
0040124D  .v 74 11           JE SHORT 3_1.00401260
0040124F  .  3C CC           CMP AL,0CC
00401251  .v 74 15           JE SHORT 3_1.00401268
00401253  .  3C DD           CMP AL,0DD
00401255  .v 75 09           JNZ SHORT 3_1.00401260
00401257  .  833D 041F400(   CMP DWORD PTR [401F04],0
0040125E  .v 74 04           JE SHORT 3_1.00401264
00401260  >  61              POPAD
00401261  .  B0 00           MOV AL,0
00401263  .  C3              RET
00401264  >  61              POPAD
00401265  .  B0 01           MOV AL,1
00401267  .  C3              RET
```

```
00401268  >  FF0D 041F400(   DEC DWORD PTR [401F04]
0040126E  .  E8 16000000     CALL 3_1.00401289
00401273  .  8918            MOV DWORD PTR [EAX],EBX
00401275  .  C603 99         MOV BYTE PTR [EBX],99
00401278  .v EB 05           JMP SHORT 3_1.0040127F
0040127A  >  E8 0A000000     CALL 3_1.00401289
0040127F  >  85C9            TEST ECX,ECX
00401281  .v 74 03           JE SHORT 3_1.00401286
00401283  .  49              DEC ECX
00401284  .^ EB BB           JMP SHORT 3_1.00401241
00401286  >  46              INC ESI
00401287  .^ EB 8E           JMP SHORT 3_1.00401217
00401289  $  57              PUSH EDI
0040128A  .  8B07            MOV EAX,DWORD PTR [EDI]
0040128C  .  8BD8            MOV EBX,EAX
0040128E  .  03C2            ADD EAX,EDX
00401290  >  8907            MOV DWORD PTR [EDI],EAX
00401292  .  C600 99         MOV BYTE PTR [EAX],99
00401295  .  C603 00         MOV BYTE PTR [EBX],0
00401298  .  83C7 04         ADD EDI,4
0040129B  .  833F 00         CMP DWORD PTR [EDI],0
0040129E  .v 74 06           JE SHORT 3_1.004012A6
004012A0  .  8BC3            MOV EAX,EBX
004012A2  .  8B1F            MOV EBX,DWORD PTR [EDI]
004012A4  .^ EB EA           JMP SHORT 3_1.00401290
004012A6  >  8BC7            MOV EAX,EDI
004012A8  .  5F              POP EDI
004012A9  .  C3              RET
```

Ta xét các bước xử lý đáng chú ý của hàm:

00401228 |. 24 03 |AND AL,3 ;

⇨ Nhận chỉ đường để di chuyển

Mỗi ký tự trong serial được giải mã thành một lệnh di chuyển theo lệnh "AND AL, 3”:

0 --> address + 10 --> đi xuống

1 --> address - 10 --> đi lên

2 --> address - 1 --> qua trái

3 --> address + 1 --> qua phải

0040122A |. C0E9 02 |SHR CL,2 ;

⇨ Số di chuyển để thực hiện theo một hướng nhất định

Xác định số lượng các di chuyển để thực hiện theo một hướng nhất định từ lệnh "SHR CL, 2”:

0 đến 3 --> 1 di chuyển

4 đến 7 --> 2 di chuyển

8 đến B --> 3 di chuyển

C đến F --> 4 di chuyển

Ta có bảng cách di chuyển dựa vào serial:

Serial Char	Hướng	Số lần di chuyển
0	0	1
1	1	
2	2	
3	3	
4	0	2
5	1	
6	2	
7	3	
8	0	3
9	1	
A	2	
B	3	
C	0	4
D	1	
E	2	
F	3	

0040124B |. 3C 99 |CMP AL,99 ;

⇨ Kiểm tra snake có ăn chính nó không

0040124F |. 3C CC |CMP AL,0CC ;

⇨ Kiểm tra snake có ăn thức ăn “CC” không

00401253 |. 3C DD |CMP AL,0DD ;

⇨ Kiểm tra snake có chạm điểm đích “DD” hay không

00401257 |. 833D 041F4000 00 |CMP DWORD PTR DS:[401F04],0 ;

⇨ Phải ăn hết số lượng thức ăn tương ứng với số ký tự của name, nếu không sẽ game over

Từ name, đã có hàm để phát sinh ma trận và tính toán để đưa các giá trị cần thiết vào ma trận. Sau đó, sẽ có hàm để kiểm tra serial.

Ta có thể mô tả luật chơi của game snake trong 3.1 như sau: Snake có đại diện là “99” trong ma trận, để đi đến chiến thắng, snake phải ăn hết các thức ăn “CC” và sau khi ăn hết thức ăn thì đi đến đích “DD”. Mỗi khi ăn một thức ăn “CC” snake sẽ dài ra thêm một đơn vị “99” , snake không được ăn chính nó và snake không được chạm “DD” khi chưa ăn hết thức ăn “CC”.

-Bài 3.1 có keygen, ý tưởng viết keygen là việc tìm đường đi từ đầu của snake đến 1 thức ăn, không được đụng điểm đích, cũng không được ăn chính nó (lưu thân snake vào vector). Thực hiện vòng lặp với số lần lặp bằng với số ký tự của name. Sau đó, tìm đường đi từ đầu của snake đến điểm đích, không được ăn chính nó. Cứ mỗi lần đi như vậy sẽ lưu hướng di chuyển vào một chuỗi. Sau khi kết thúc, ta dựa vào bảng cách di chuyển để chuyển thành hướng đi cho đúng quy định của 3.1. Dĩ nhiên, trong quá trình tìm đường, mỗi lần như vậy sẽ có hàm để tối ưu đường đi để cắt bớt đường đi thừa.

Một ví dụ về keygen với:

Name: abcde

Serial:
DD1BCCC83DDD9E2CCC83DDD92C82D9ACCC82DDD92CCC83DD53D1FF3CCC83DDD5