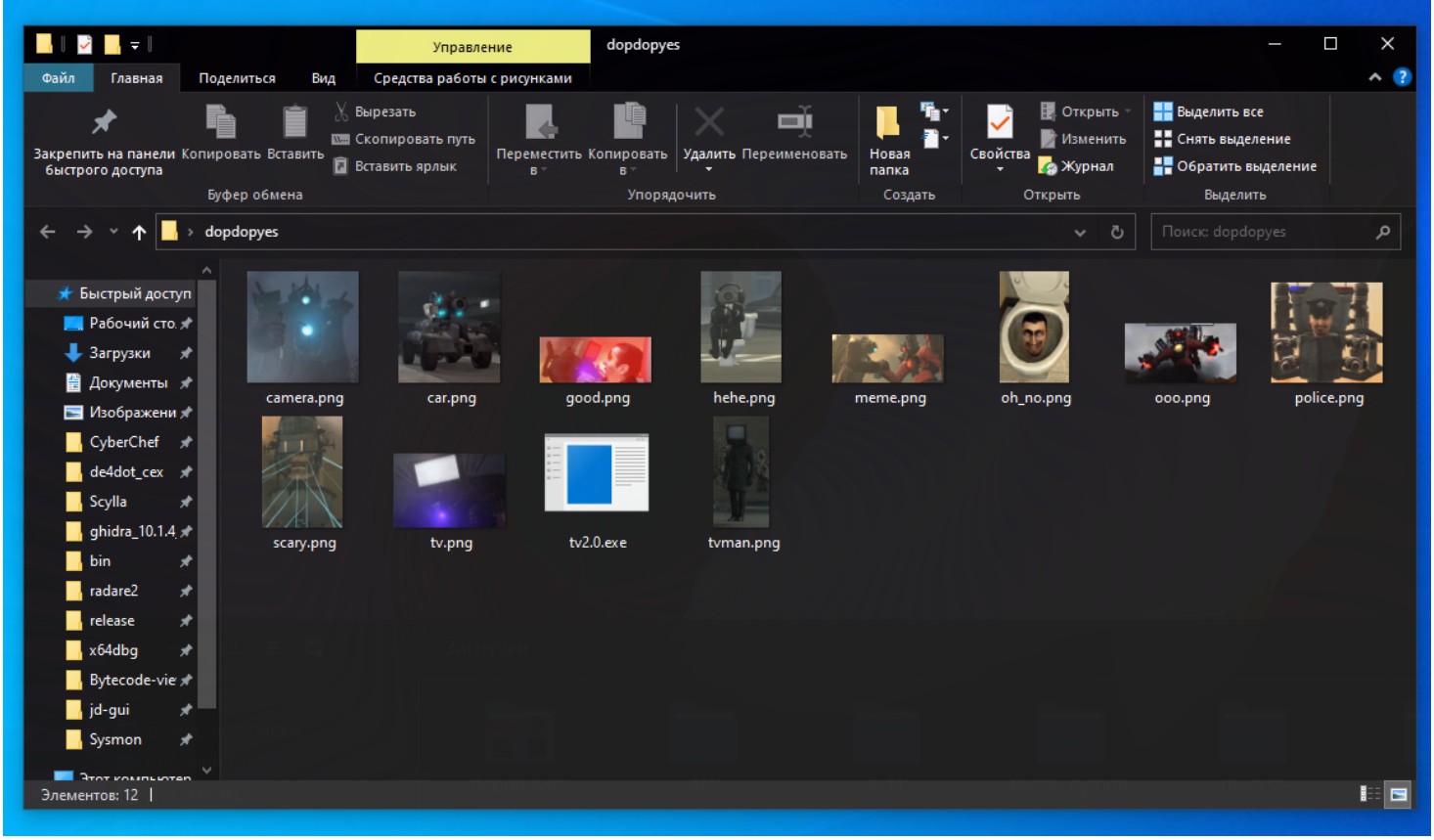
доп доп ec

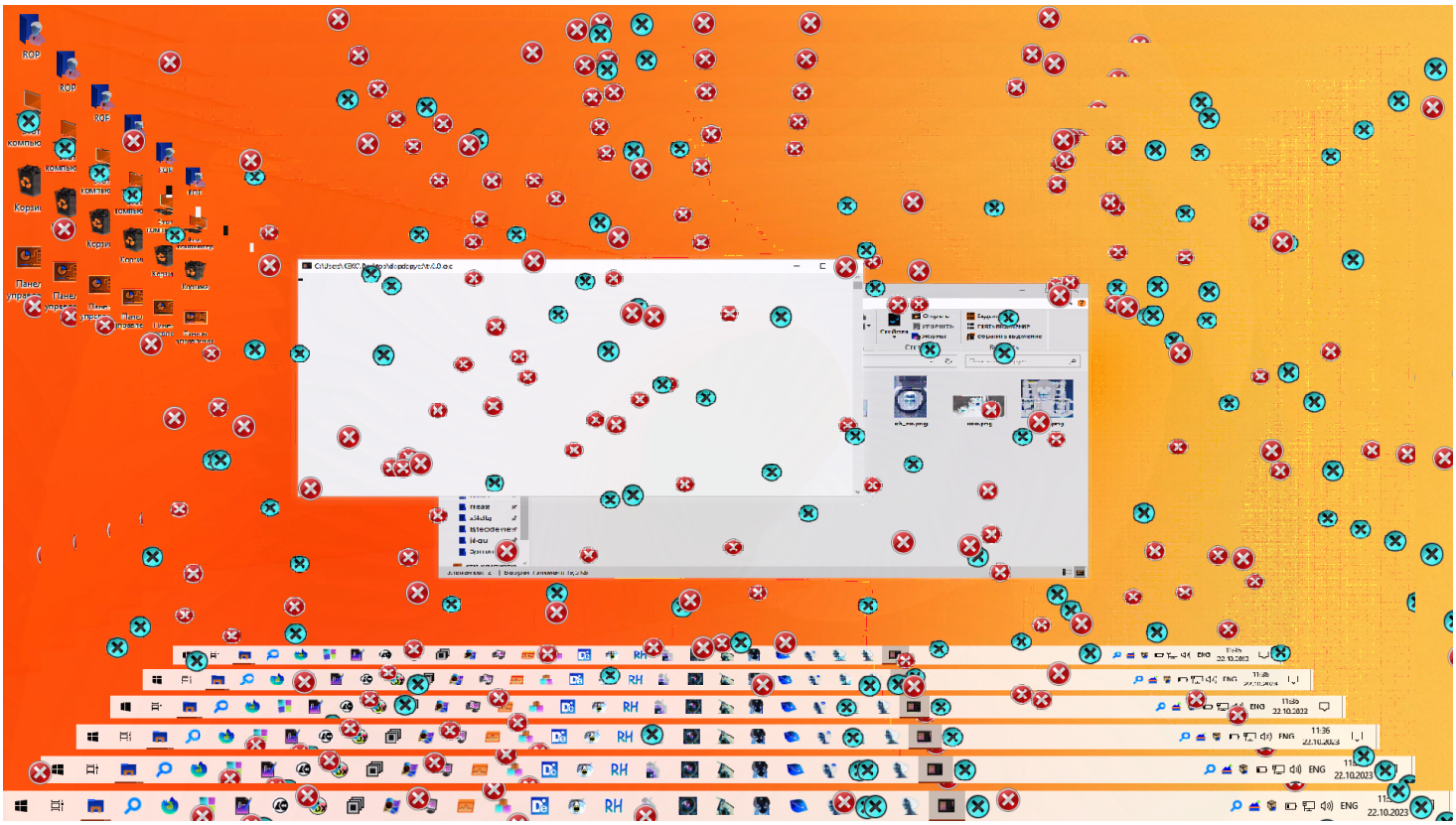
Краткая информация
Название: доп доп ec
Описание: Я нашёл это у своего внука на компьютере. Очень не понимаю, почему там какие-то туалеты и роботы. Может, вы знаете?
Категория: Реверс-инжиниринг
Сложность: Легко
Очки: 400
Подсказка: ALT+F4 спасёт мир

Райтап

1. Распакуем архив, и видим картинки с EXE-файлом.



2. Попробуем запустить.



Ох, зря.

3. Закроем программу и изучим её в IDA. Можем использовать декомпилятор.

```
61          Buffer[i] ^= v12 ^ 0x3C;
62          Buffer[i] += (i + v13) ^ (v19 + 52);
63          Buffer[i] ^= (unsigned __int8)(v12 + v19) ^ 0x19;
64          Buffer[i] += v13 ^ 0x50;
65          Buffer[i] -= 52;
66          Buffer[i] -= i ^ 0xAB;
67          Buffer[i] ^= i + v19 + 116;
68          Buffer[i] ^= v19;
69          Buffer[i] -= v19;
70          Buffer[i] ^= 0xCCu;
71          Buffer[i] += i;
72          Buffer[i] ^= 0xAAu;
73          Buffer[i] += 49;
74          Buffer[i] ^= 0x44u;
75          v19 = v11;
76        }
77        Stream = fopen("flag.txt", "wb");
78        if ( Stream )
79        {
80          fwrite(Buffer, 1ui64, 0x9B9ui64, Stream);
81          fclose(Stream);
82        }
83        ReleaseDC(hWnd, hDC);
84        --v22;
85      }
86    }
87    return 0;
88  }
```

Если мы сделаем всё, как нужно, то появится файл `flag.txt` с флагом.

```
28    hWnd = GetDesktopWindow();
29    hDC = GetWindowDC(hWnd);
30    GetWindowRect(hWnd, &Rect);
31    while ( v21 <= 49 )
32    {
33      v3 = rand();
34      X = v3 % GetSystemMetrics(0);
35      v4 = rand();
36      Y = v4 % GetSystemMetrics(1);
37      IconA = LoadIconA(0i64, (LPCSTR)0x7F01);
38      DrawIcon(hDC, X, Y, IconA);
39      ++v21;
40    }
41    v21 = 0;
42    Sleep(0x44Cu);
43    GetCursorPos(&Point);
44    BitBlt(hDC, 0, 0, Rect.right - Rect.left, Rect.bottom - Rect.top, hDC, 0, 0, 0x330008u);
45    StretchBlt(hDC, 50, 50, Rect.right - 100, Rect.bottom - 100, hDC, 0, 0, Rect.right, Rect.bottom, 0xCC0020u);
46    if ( Point.x == Point.y
47      && (Point.x ^ 0x13371337) > 322376637
48      && (Point.y ^ 0x13371336) <= 322376641
49      && Point.y % 121 == 16 )
50    {
51      memcpy(Buffer, &unk_404020, 0x9B9ui64);
52      v14 = 2489;
53      v19 = 0;
54      v13 = Point.x;
55      v12 = Point.y;
```

А нужно, чтобы выполнились эти условия:

```
if ( Point.x == Point.y
  && (Point.x ^ 0x13371337) > 322376637
  && (Point.y ^ 0x13371336) <= 322376641
```

```
      && Point.y % 121 == 16 )
    {
```

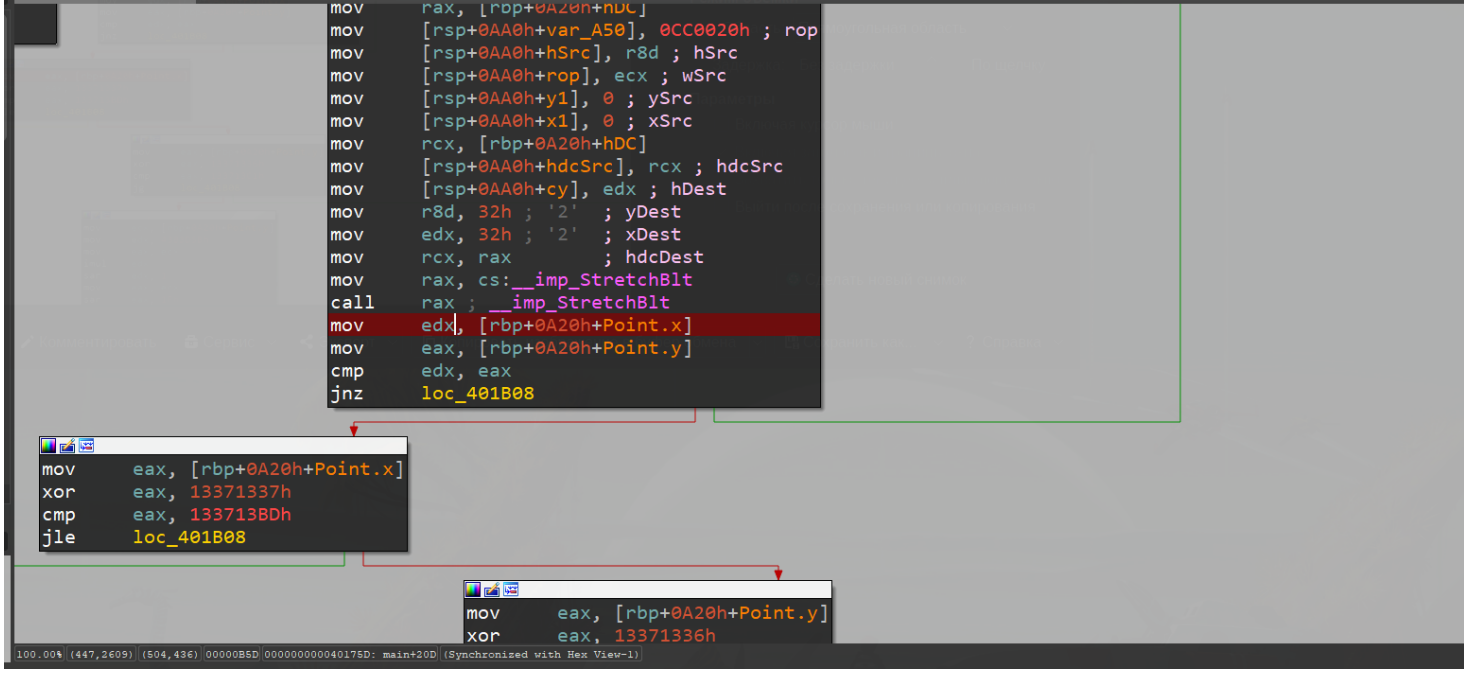
Тут `Point.x` и `Point.y` - координаты курсора на экране.

4. Напишем программу для решения этого условия.

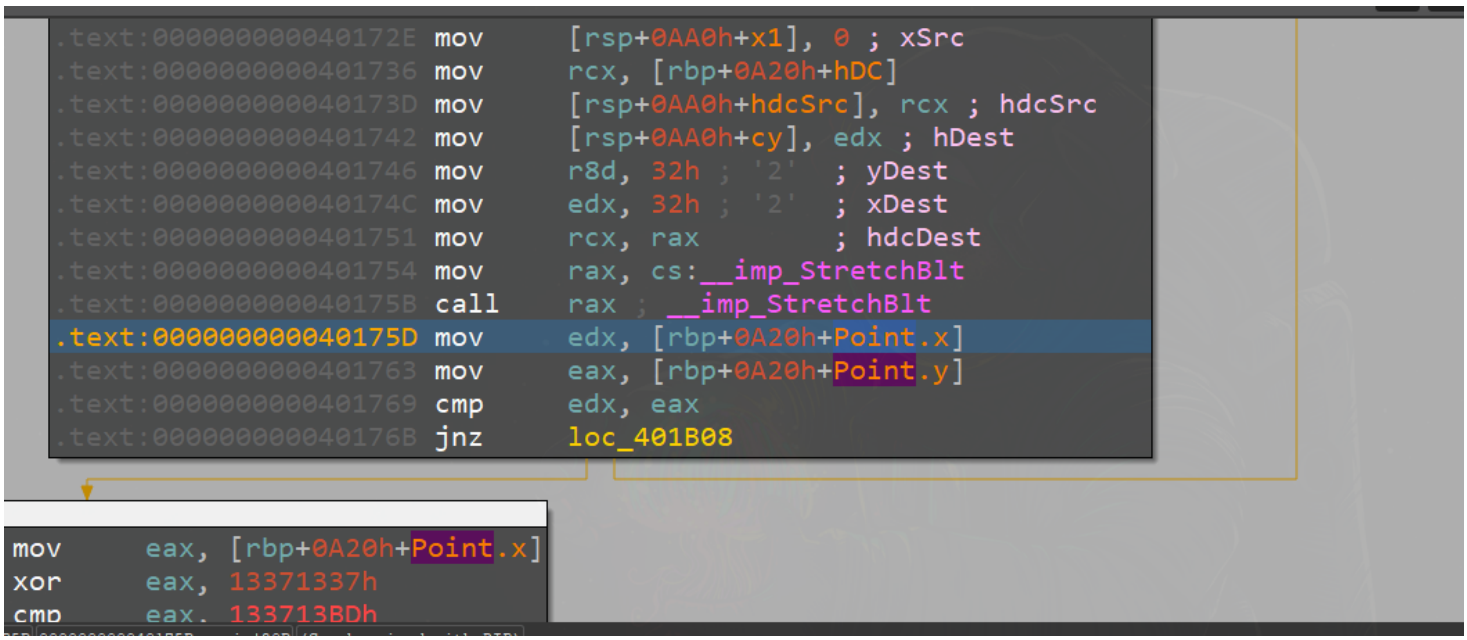
```
#include <stdio.h>

int main() {

        unsigned int x;
        unsigned int y;


        for (x = 0; x < 1000000; x++) {
                y = x;
                if ( x == y
                  && (x ^ 0x13371337) > 322376637
                  && (y ^ 0x13371336) <= 322376641
                  && y % 121 == 16 )
                {
                        printf("(%d, %d)", x, y);
                }
        }

        return 0;
}
```

Вывод: `(137, 137)`

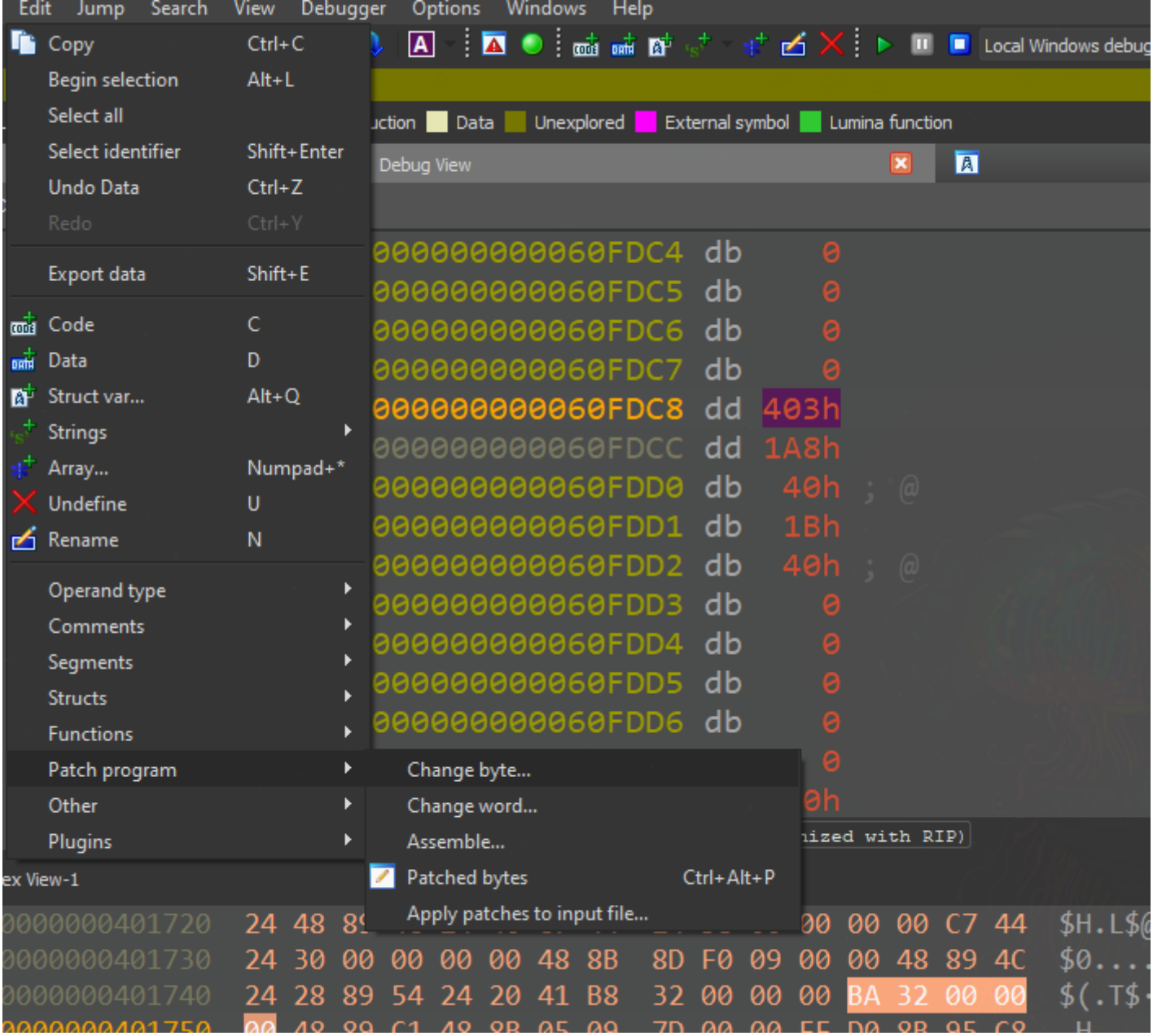
5. Так как вручную наводить курсор на эту координату неудобно, то поменяем просто значение `Point.x` и `Point.y` при отладке.

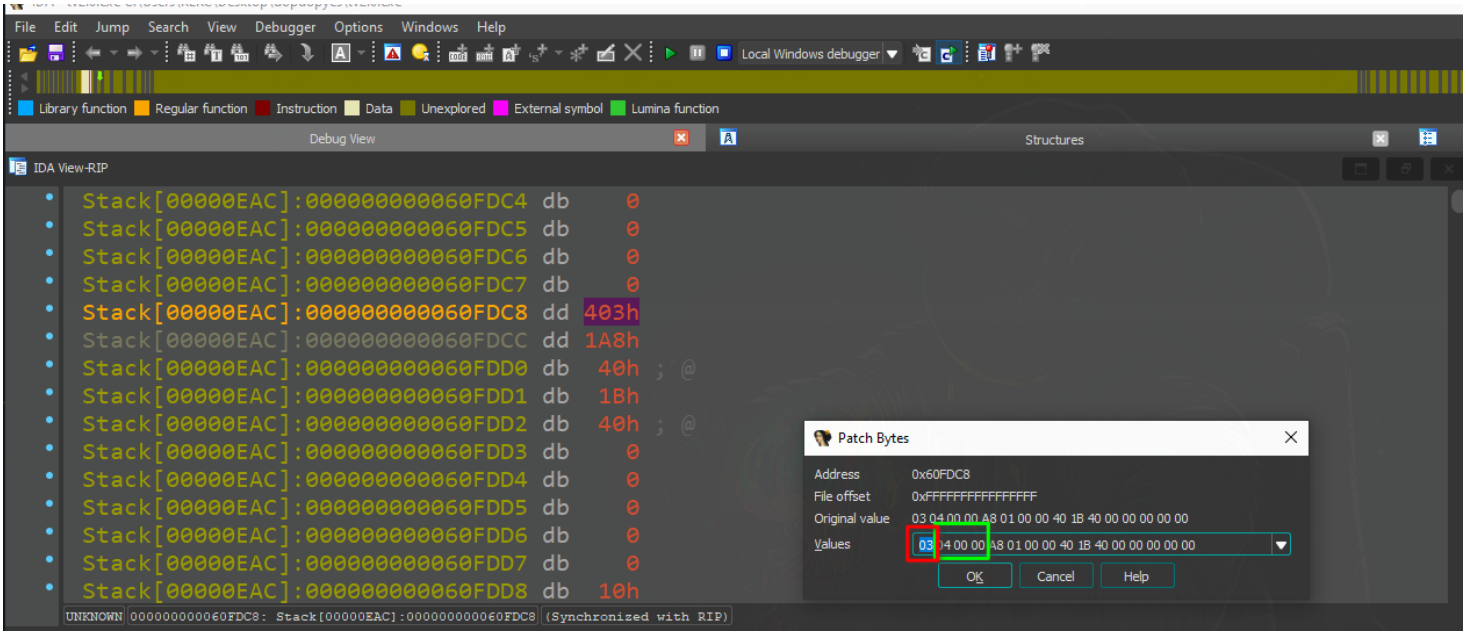


Ставим точку останова сюда. Запускаем отладчик.

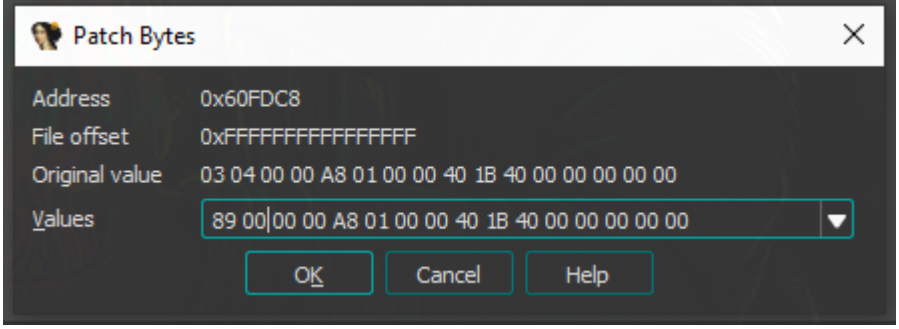


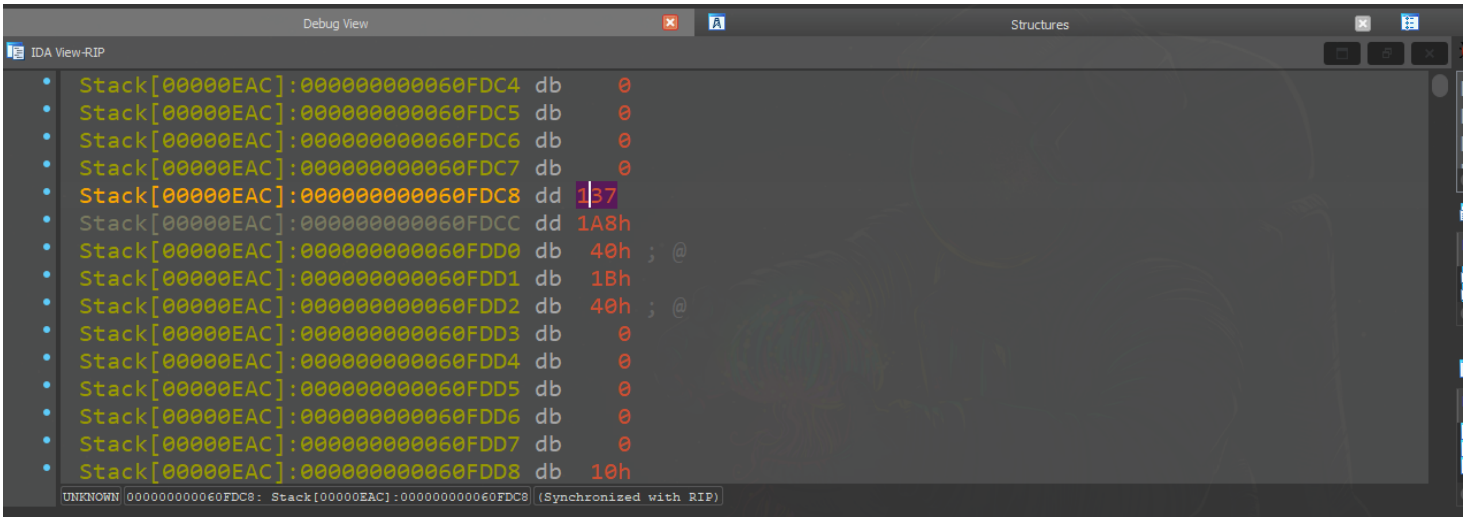
Нажимаем 2 раза ЛКМ на `Point.x`. Через Change byte поменяем байты.





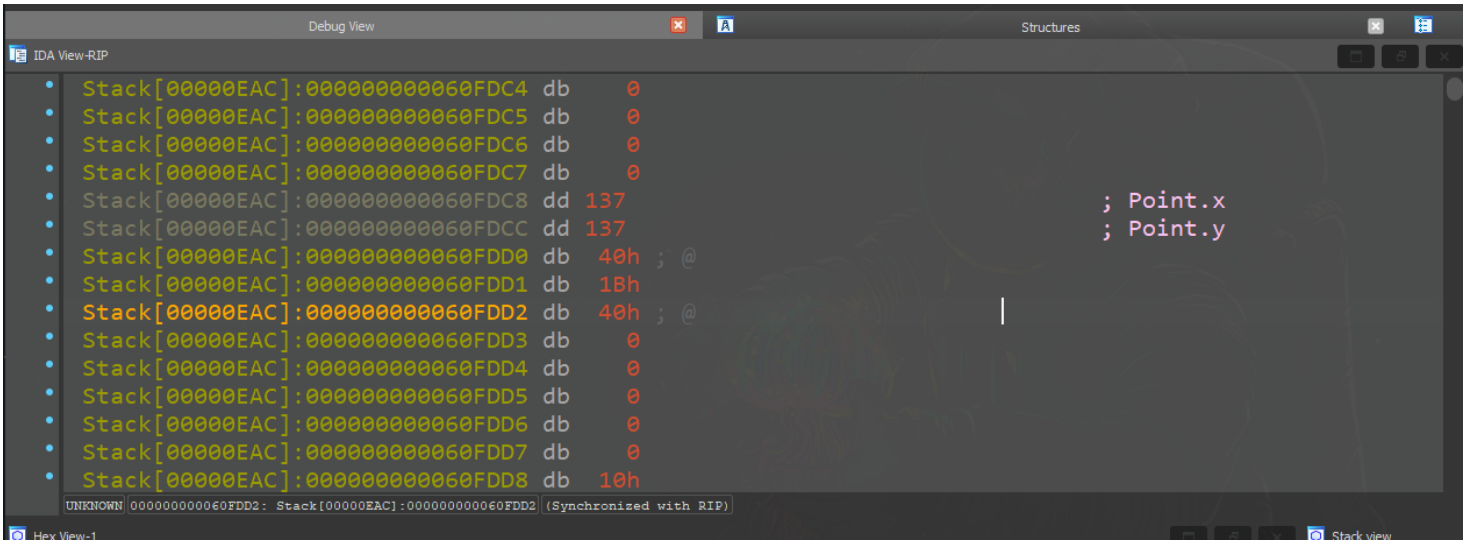
137 - это 0x89. Поэтому красное выделение меняем на 0x89, а в зелёном сделаем 0.



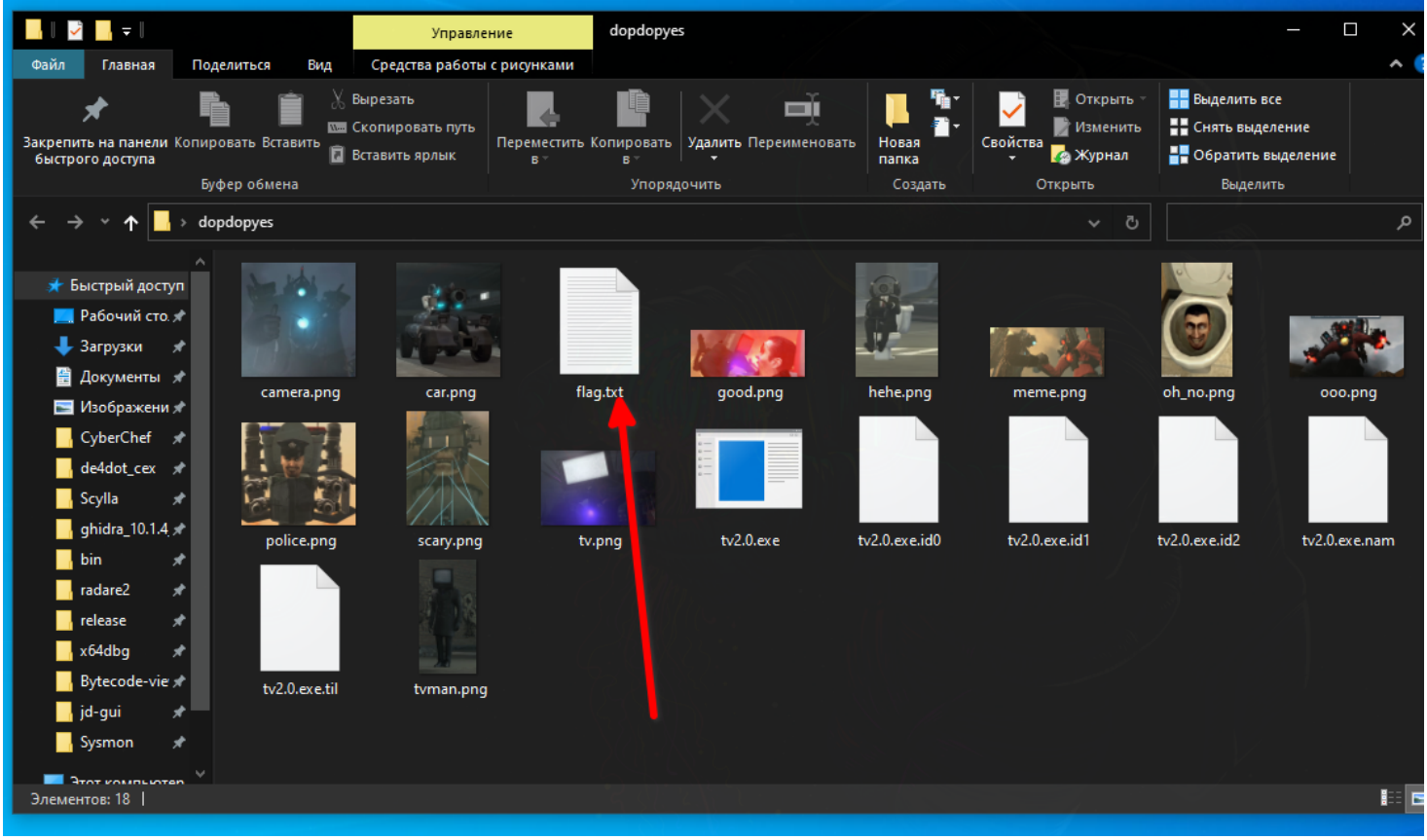


Отлично!

Сделаем то же самое с `Point.y`.



Дальше просто выполняем программу.



Появился флаг.

Его содержимое:

```
01010101 01000111 01111000 01101100 01011001
01011000 01001110 01101100 01001001 01000111
01010010 01110110 01001001 01000111 00110101
01110110 01100100 01000011 01000010 01111001
01100100 01010111 00110100 01100111 01100100
01010111 00110101 01101101 01011001 01010111
00110001 01110000 01100010 01000111 01101100
01101000 01100011 01101001 01000010 01101101
01100001 01010111 01111000 01101100 01100011
01111001 01000010 01110110 01100010 01101001
01000010 00110101 01100010 00110011 01010110
01111001 01001001 01001000 01001110 00110101
01100011 00110011 01010010 01101100 01100010
01010011 00110100 01100111 01010001 01010111
01111000 00110011 01011001 01011000 01101100
01111010 01001001 01000111 01000110 01110101
```

01011001 01010111 01111000 00110101 01100101
01101101 01010101 01100111 01100100 01000111
01101000 01101100 01100010 01010011 01000010
01101001 01011010 01010111 01011010 01110110
01100011 01101101 01010101 01100111 01100011
01101110 01010110 01110101 01100010 01101101
01101100 01110101 01011010 01111001 01000010
00110000 01100001 01000111 01010110 01110100
01001100 01000011 01000010 01101000 01100011
01111001 01000010 00110000 01100001 01000111
01010101 01100111 01011010 01000111 01000110
01110100 01011001 01010111 01100100 01101100
01001001 01000111 00111001 01101101 01001001
01000111 01111000 01101000 01100101 01101101
01101100 01110101 01011010 01011000 01001110
01111010 01001001 01000111 00110001 01101000
01100101 01010011 01000010 01101001 01011010
01010011 01000010 01110100 01100010 00110011
01001010 01101100 01001001 01001000 01010010
01101111 01011001 01010111 00110100 01100111
01100100 00110010 01000110 01111010 01100100
01000111 01101100 01110101 01011010 01111001
01000010 01111010 01100010 00110010 00110001
01101100 01001001 01001000 01010010 01110000
01100010 01010111 01010101 01110101 01000011
01100111 01110000 01000100 01010100 00110000
01010010 01000110 01010001 01101100 01101100
00110111 01011001 01010111 01111000 00110011
01010001 01011000 01101100 01010100 01011000
00110000 01001110 01101111 01001101 01101101
01001110 01110010 01011000 00110011 01010010
01001001 01011010 01010110 00111001 01101101
01001101 01010111 01111000 01101100 01010101
00110001 00111001 01101001 01011010 01010111

```
01011001 01110111 01100011 01101011 01010110
01100110 01100011 00110001 01010010 01101000
01100011 01101110 01010001 01111000 01100010
01101101 01100100 00111001 00001010 00001010
01000010 01100001 01110011 01100101 00110110
00110100

Bin
```

6. Декодируем всё из двоичного кода.

```
UGxlYXNlIGRvIG5vdCBydW4gdW5mYW1pbGhciBmaWxlcyBvbiB5
b3VyIHN5c3RlbS4gQWx3YXlzIGFuYWx5emUgdGhlbSBiZWZvcmUg
cnVubmluZyB0aGVtLCBhcyB0aGUgZGFtYWdlIG9mIGxhemluZXNz
IG1heSBiZSBtb3JlIHRoYW4gd2FzdGluZyBzb21lIHRpbWUuCgpD
T0RFQll7YWx3QXlTX0NoMmNrX3RIZV9mMWxlU19iZWYwckVfc1Rh
cnQxbmd9

Base64
```

Теперь из Base64.

```
Please do not run unfamiliar files on your system.
Always analyze them before running them, as the
damage of laziness may be more than wasting some
time.

CODEBY{alwAyS_Ch2ck_tHe_f1leS_bef0rE_sTart1ng}
```

Переводим на русский:

```
Пожалуйста, не запускайте незнакомые файлы в вашей
системе. Всегда анализируйте их перед запуском, так
```

```
как ущерб от лени может быть больше, чем пустая
трата времени.

CODEBY{alwAyS_Ch2ck_tHe_f1leS_bef0rE_sTart1ng}
```

Флаг: `CODEBY{alwAyS_Ch2ck_tHe_f1leS_bef0rE_sTart1ng}`