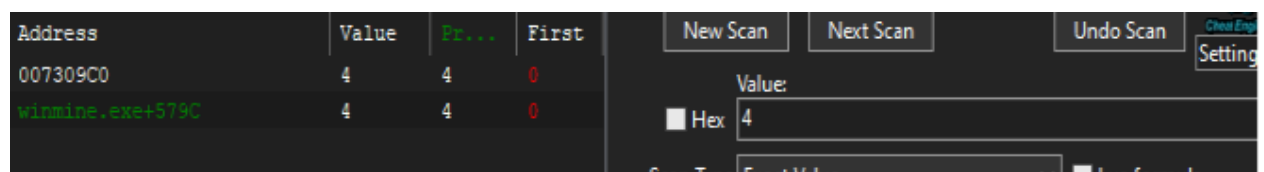


1) Freeze Timer: запустим cheatEngine и запустим поиск по значению в бинаре. Для начала это будет 0, т.к. таймер игры не запущен. После первого удачного нажатия на поле(или нет) таймер будет запущен:

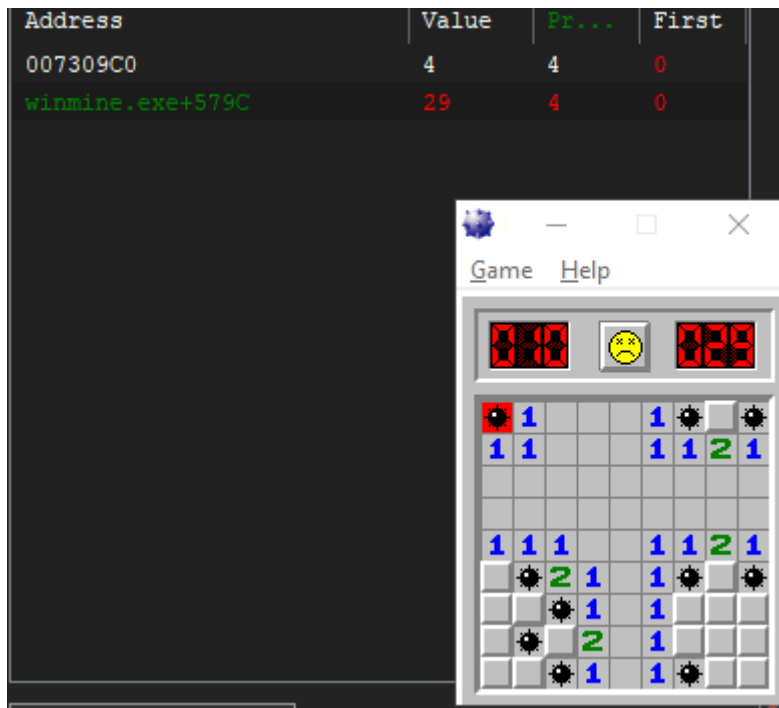


После проигрыша таймер остановит ход времени. По этому значению мы и будем искать область памяти в cheatEngine

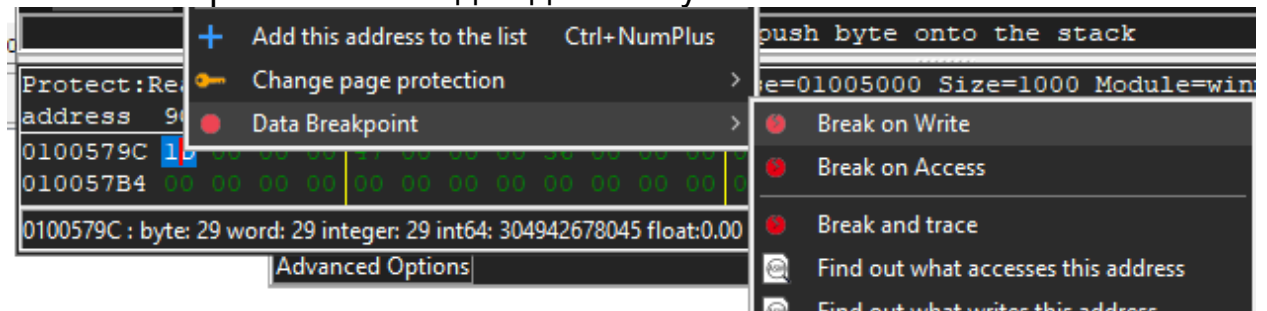


Всего найдено 2 записи со значением 4. Нам нужна вторая.

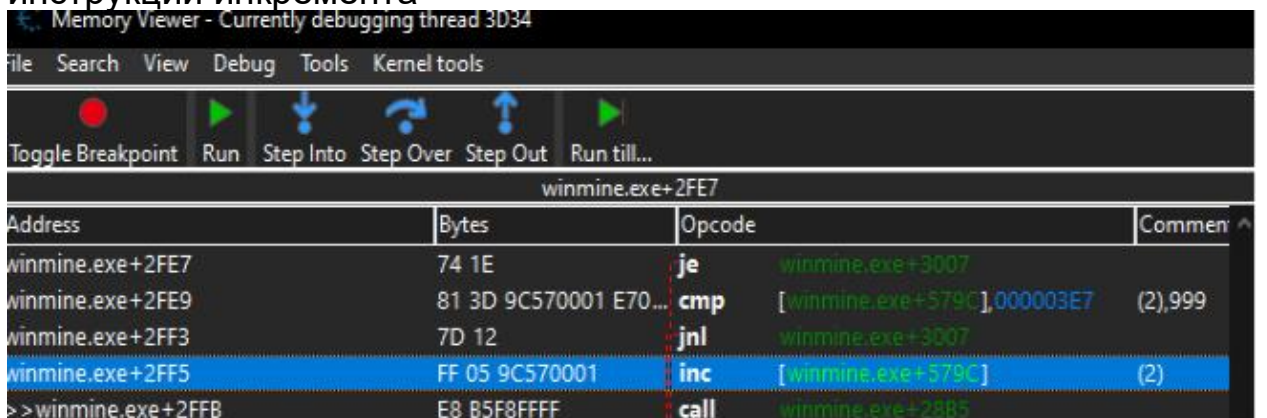
Это легко проверить, запустив игру заново. Будет видно, как значение данного участка памяти изменяется параллельно значению таймера в игре.



Поставим бряк на запись для данного участка памяти



Начнем новую игру и программа остановит своё выполнение на инструкции инкремента



идем в иде по данному адресу и видим такой участок кода








```

int sub_1002FE0()
{
    int result; // eax

    if ( dword_1005164 )
    {
        if ( m_timer < 999 )
        {
            ++m_timer;
            sub_10028B5();
            result = sub_10038ED(1);
        }
    }
    return result;
}

```

и переименовываем переменную в m_timer. Ищем перекрестные ссылки данной переменной

xrefs to m_timer			
Direction	Typ	Address	Text
	Up	r sub_1002825+8	mov edi, m_timer
	Up	r sub_1002FE0+9	cmp m_timer, 3E7h
	w	sub_1002FE0+15	inc m_timer
	Do...	r sub_100347C+73	mov ecx, m_timer
	Do...	w sub_100367A+A0	mov m_timer, edi
	Do...	r sub_10037E1+40	cmp m_timer, 0
	Do...	w sub_10037E1+4F	inc m_timer

Line 7 of 7

OK Cancel Search Help

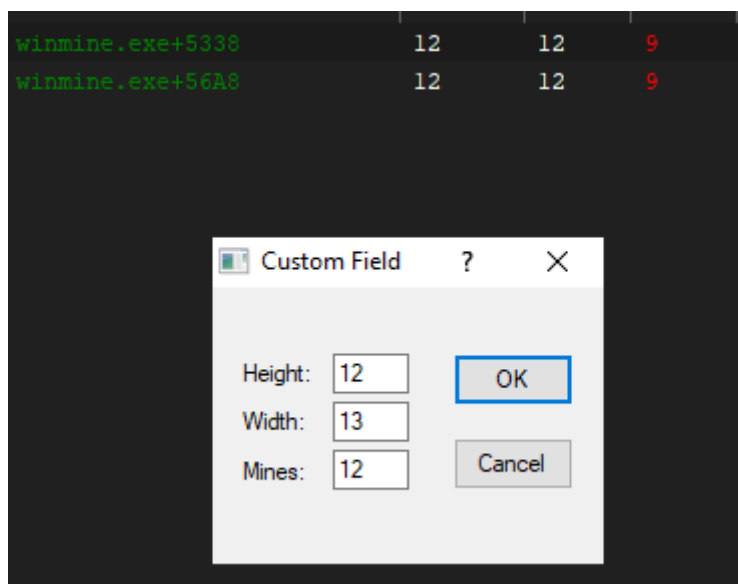
достаточно заНОРать инструкции с инкрементов переменной m_timer, чтобы таймер больше не работал (адреса инструкций: 0x01003830, 0x01002FF5)

Как можем наблюдать, таймер действительно больше на работает:



DONE!

2)Extract Tile Layout: с помощью cheatEngine найдем область памяти, в которой хранится информация о размерах игрового поля. Установим значение 9 и выполним поиск, после чего изменим значение на высоты на 12, ширины на 13 и снова просканируем память



Address	Value	Pr...	First
winmine.exe+5334	13	13	9
winmine.exe+56AC	13	13	9

Custom Field ? X

Height: 12

Width: 13

Mines: 12

OK

Cancel

Посмотрим в ida на этот участок памяти и его перекрестные ссылки. натываемся на один интересный момент....

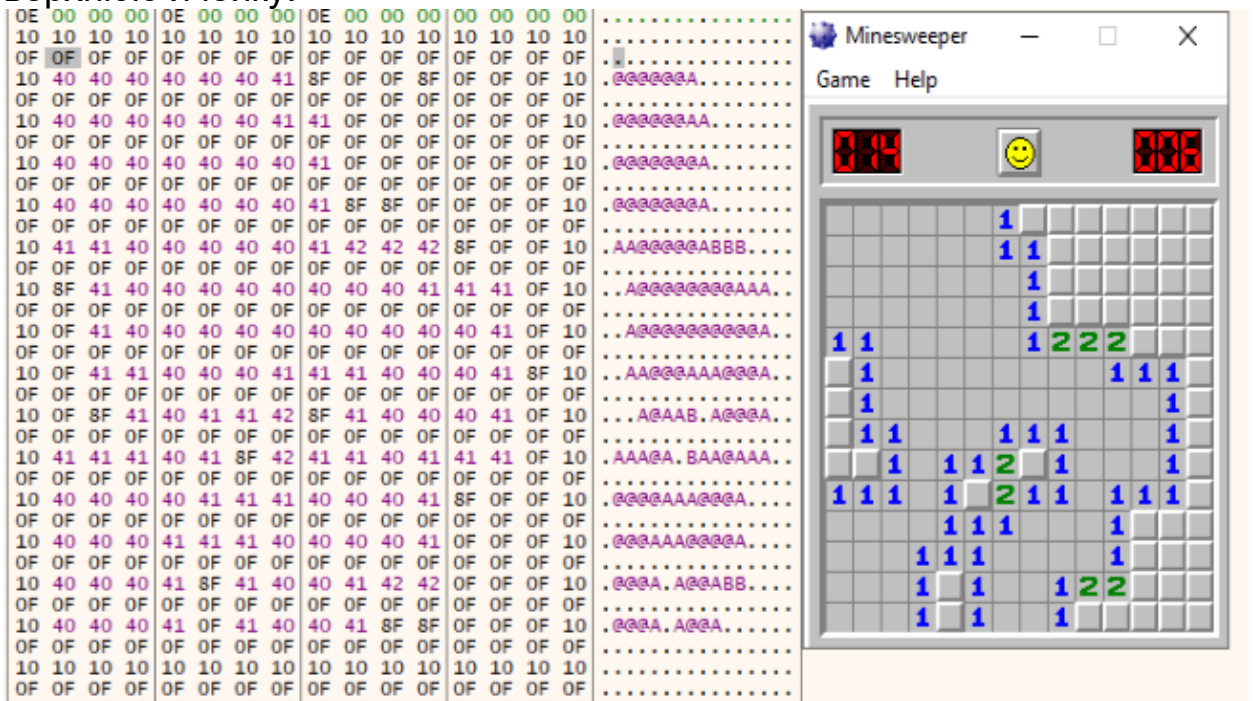
Это использование какого-то массива. Предположительно, в нём и хранится наша карта

```

v2 = &m_map[32 * a2 + a1];
if ( *v2 >= 0 )
{
    sub_1003084(a1, a2);
    LOWORD(v3) = dword_10057A4;
    if ( dword_10057A4 == dword_10057A0 )
        LOWORD(v3) = __lose_func(1);
}
else if ( dword_10057A4 )
{
    sub_1002EAB(a1, a2, 76);
    LOWORD(v3) = __lose_func(0);
}
else
{
    v3 = 1;
    if ( height > 1 )
    {
        for ( i = (char *)&byte_1005360; ; i += 32 )
        {
            v5 = 1;
            if ( width > 1 )
                break;
        }
        LABEL_8:
        if ( ++v3 >= height )
            return v3;
    }
    while ( i[v5] < 0 )
    {
        if ( ++v5 >= width )
            goto LABEL_8;
    }
    *v2 = 15;
    m_map[32 * v3 + v5] |= 0x80u;
    LOWORD(v3) = sub_1003084(a1, a2);
}

```

проверим это запустив сапера под отладчиком и перейдя по нужному адресу в памяти. (Размер поля 14x14) выберем крайнюю левую верхнюю ячейку.



Получим такие значения:

0x40 – проверенная пустая ячейка

0x41 – ячейка со значением 1

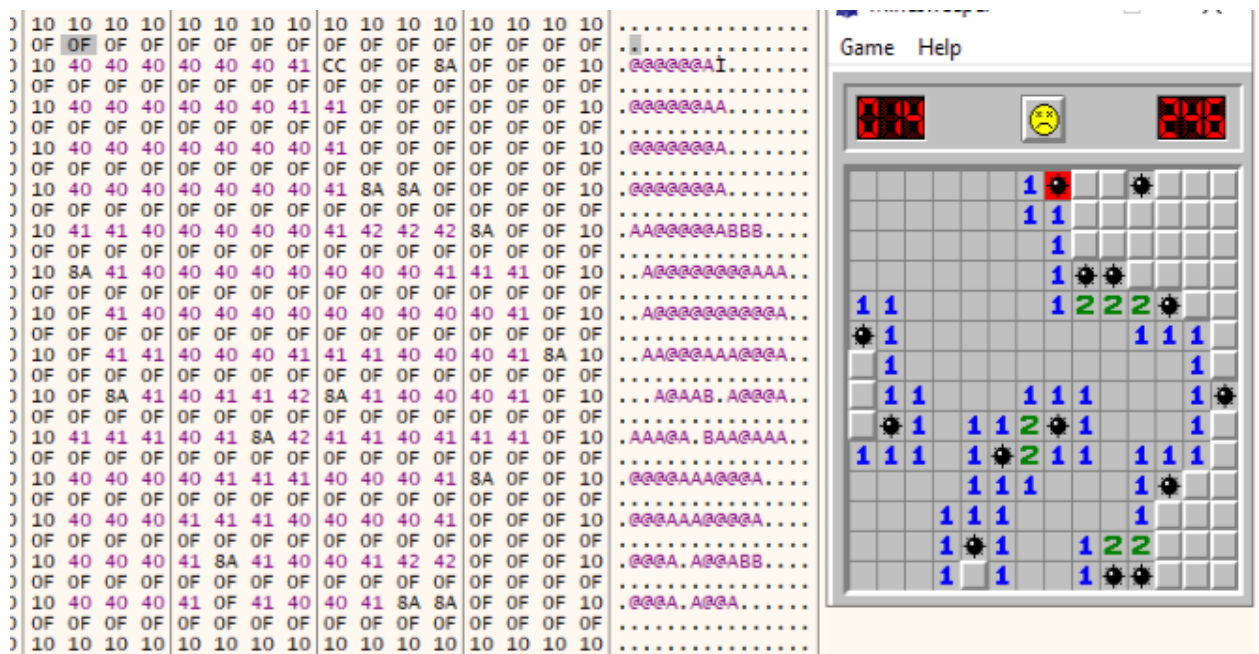
0x42 – ячейка со значением 2

0x43 – ячейка со значением 3

0x8F – мина

0x0F – пустая невыбранная ячейка

Если специально выбрать мину, массив примет дополнительные значения:

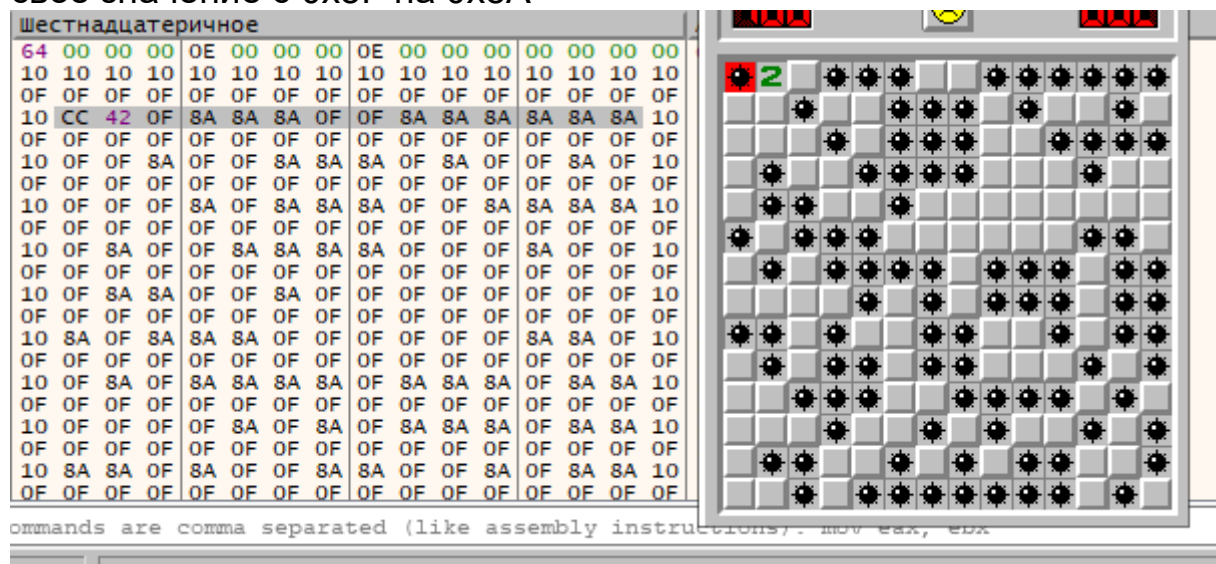


0xCC – мина, на которой мы «подорвались»

0x8A – помеченная мина при проигрыше

3) Inert Mine Tiles достаточно изменить значение байта 0x8F (означает мину) на 0x8E (помеченная мина), чтобы ячейка не сработала

4) Show Mines заметим, что после проигрыша мины в памяти меняют своё значение с 0x8F на 0x8A



Поставим бряк на запись на тот байт, в котором находится мина и специально проиграем. Отладчик остановится на адресе 0x01002FCA в функции sub_01002F80. Проанализировав её в иде мы поймем, что она пробегается по игровому полю и помечает мины значением 0x8A, после чего вызывается функция sub_0100272E, которая и отвечает за вывод мин на экран. Проанализировав данную

функу мы поймем, что используется GDI и и как написать функции вывода мин на игровое поле.