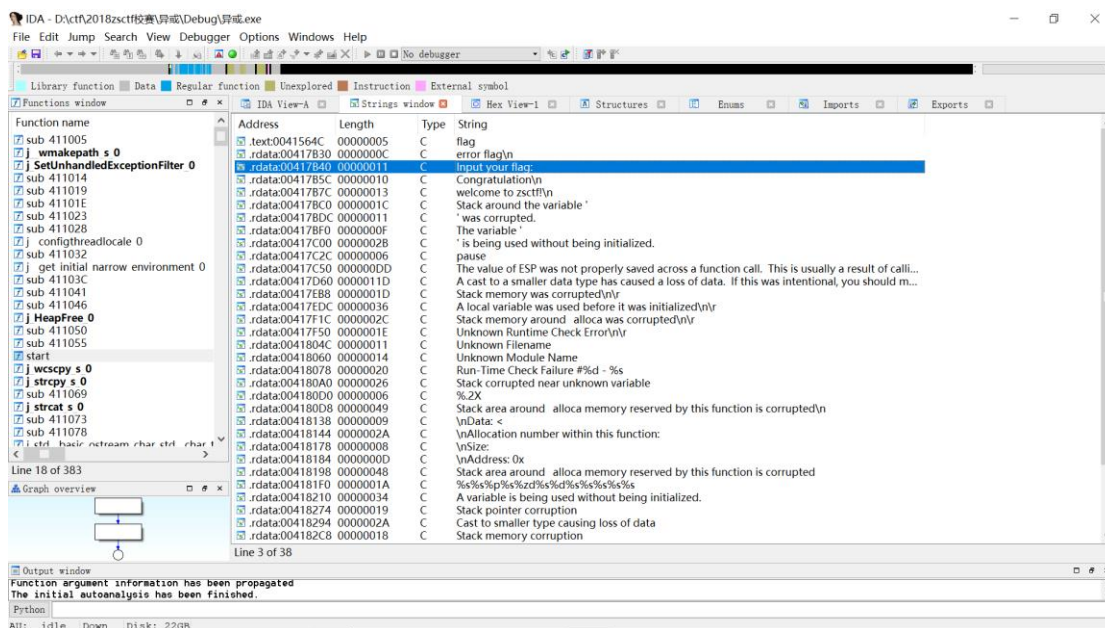




从程序可以看出是让我们输入 flag 然后判断是否正确

拖进 IDA，在字符串(Shift+F12)里看到 *Input your flag:*



跟过去 f5 反出 c 代码，基本可以判断这就是 main 函数

```

4 char v5; // [sp+0h] [bp-118h]@1
5 char v6; // [sp+Ch] [bp-10Ch]@1
6 unsigned int i; // [sp+D0h] [bp-48h]@1
7 char input[56]; // [sp+DCh] [bp-3Ch]@1
8 unsigned int v9; // [sp+114h] [bp-4h]@1
9 int savedregs; // [sp+118h] [bp+0h]@1
10
11 memset(&v6, 0xCCu, 0x10Cu);
12 v9 = (unsigned int)&savedregs ^ __security_cookie;
13 printf("welcome to zscf!\n", v5);
14 printf("Input your flag:", v5);
15 scanf_s("%s", input, 50);
16 for ( i = 0; ; ++i )
17 {
18     v3 = j_strlen_0(input);
19     if ( i >= v3 )
20         break;
21     if ( input[i] < 'A' || input[i] > '}' ) // 判断输入的字符是否在 A 到 } 之间
22     {
23         printf("error flag\n", v5); // 如果不是则输出error flag, 程序结束
24         system("pause");
25         sub_41114F();
26         goto LABEL_11;
27     }
28 }
29 sub_4113B6(input, (int)&unk_41A050);
30 if ( (unsigned __int8)sub_41118B(input) ) |
31     printf("Congratulation\n", v5);
32 else
33     printf("error flag\n", v5);
34     system("pause");
35     sub_41114F();
36 LABEL_11:
37 sub_41129E(&savedregs, &dword_415638);
38 sub_4112B2();
39 return sub_41114F();
40 }

```

查看 main 函数，发现输入后先判断输入字符是否在 A 到 } 之间，之后调用函数 sub_4113B6 对输入进行处理，最后再通过函数 sub_41118B 与已有数据进行比较判断是否正确。

```

1 int __cdecl sub_4153D0(char *input, int a2)
2 {
3     size_t v2; // eax@5
4     char v4; // [sp+Ch] [bp-D8h]@1
5     int i; // [sp+D0h] [bp-14h]@1
6     size_t v6; // [sp+DCh] [bp-8h]@1
7
8     memset(&v4, 0xCCu, 0xD8u);
9     v6 = 0;
10    for ( i = j_strlen_0(input) - 1; (signed int)v6 < i; --i )
11    {
12        input[v6] ^= input[i]; // 通过三次异或将数组头尾依次对调，将数组颠倒
13        input[i] ^= input[v6];
14        input[v6] ^= input[i];
15        ++v6;
16    }
17    v6 = 0;
18    for ( i = 0; ; ++i )
19    {
20        v2 = j_strlen_0(input);
21        if ( v6 >= v2 )
22            break;
23        if ( i >= 3 )
24            i = 0;
25        input[v6] ^= *(_BYTE *) (i + a2); // 讲输入与三个数依次进行异或
26        ++v6;
27    }
28    return sub_41114F();
29 }

```

在函数 sub_4153D0 里先将输入前后颠倒, 再与传参进来的三个字节依次进行异或, 在 main 里查看了一下传进来的 unk_41A050 地址所在出的数据就是 0xAA,0xBB,0xCC 三个字节。

```
.data:0041A050 unk_41A050      db  0AAh ;  
.data:0041A051              db  0BBh ;  
.data:0041A052              db  0CCh ;  
.data:0041A053              db   0 ;  
.data:0041A054              db   0 ;
```

在函数 sub_4117A0 中与循环 24 次与 byte_41A038[]进行比较

```
1 char __cdecl sub_4117A0(int a1)  
2 {  
3     char v2; // [sp+Ch] [bp-CCh]@1  
4     int i; // [sp+D0h] [bp-8h]@1  
5  
6     memset(&v2, 0xCCu, 0xCCu);  
7     for ( i = 0; i < 24; ++i )  
8     {  
9         if ( *(_BYTE *)(i + a1) != byte_41A038[i] )  
10            return 0;  
11     }  
12     return 1;  
13 }
```

```

.data:0041A038 ; char byte_41A038[]
.data:0041A038 byte_41A038      db 0D7h
.data:0041A039      db 0D5h
.data:0041A03A      db 0A3h
.data:0041A03B      db 0D9h
.data:0041A03C      db 0DAh
.data:0041A03D      db 0A9h
.data:0041A03E      db 0D8h
.data:0041A03F      db 0E4h
.data:0041A040      db 0ADh
.data:0041A041      db 0F5h
.data:0041A042      db 0C8h
.data:0041A043      db 0A5h
.data:0041A044      db 0F5h
.data:0041A045      db 0DEh
.data:0041A046      db 0BEh
.data:0041A047      db 0CFh
.data:0041A048      db 0D3h
.data:0041A049      db 98h
.data:0041A04A      db 0D1h
.data:0041A04B      db 0DDh
.data:0041A04C      db 0B8h
.data:0041A04D      db 0C9h
.data:0041A04E      db 0C8h
.data:0041A04F      db 0B6h

```

至此可看出程序要求输入 24 个在 A 到 } 之间的字符，之后前后颠倒并与 0xAA,0xBB,0xCC 三个字节依次异或，最后与地址 0x41A038 处数据进行对比判断正误。

那么直接对 0x41A038 位置的 24 个字符串处理即可得到 flag

IDC 脚本如下(Shift+F2):

```

1  auto i,c;
2  for(i=0,c=0;i<24;i++,c++)
3      Message(Byte(0x41a04f-i)^Byte(0x41a052-c%3));
4

```

```

Output window
4117A0: using guessed type _DWORD __cdecl sub_4117A0(_DWORD);
zsctf{There_is_a_reason}
Python

```

得到 flag zsctf{There_is_a_reason}