

int v19; // eax
int v20; // eax
int v21; // eax
int v22; // eax
int v23; // eax
int v24; // eax
int v25; // eax
int v26; // eax
int v27; // eax
int v28; // eax
int v29; // eax
int v30; // eax
int v31; // eax
int v32; // eax
int v33; // eax
int v34; // eax
int v35; // eax
int v36; // eax
int v37; // eax
int v38; // eax
int v39; // eax
int v40; // eax
int v41; // eax
int v42; // eax
int v43; // eax
int v44; // eax
int v45; // eax
int v46; // eax
int v47; // eax
int v48; // eax
int v49; // eax
int v50; // eax
int v51; // eax
int v52; // eax
int v53; // eax
int v54; // eax
int v55; // eax
int v56; // eax
int v57; // eax
int v58; // eax
int v59; // eax
int v60; // eax
int v61; // eax
int v62; // eax
int v63; // eax
int v64; // eax
int v65; // eax
int v66; // eax
int v67; // eax
int v68; // eax
int v69; // eax

```
int v70; // eax
int v71; // eax
int v72; // eax
int v73; // eax
int v74; // eax
int v75; // eax
int v76; // eax
int v77; // eax
int v78; // eax
int v79; // eax
int v80; // eax
int v81; // eax
int v82; // eax
int v83; // eax
int v84; // eax
int v85; // eax
int v86; // eax
int v87; // eax
int v88; // eax
int v89; // eax
int v90; // eax
int v91; // eax
int v92; // eax
int v93; // eax
int v94; // eax
int v95; // eax
int v96; // eax
int v97; // eax
int v98; // eax
int v99; // eax
int v100; // eax
int v101; // eax
int v102; // eax
int v103; // eax
int v104; // eax
int v105; // eax
int v106; // eax
int v107; // eax
int v108; // eax
int v109; // ST1C_4
int v110; // eax
int v111; // eax
int v112; // ST20_4
int v113; // eax
int v114; // eax
int v115; // ST20_4
int v116; // eax
signed int i; // [esp+4h] [ebp-90h]
signed int j; // [esp+8h] [ebp-8Ch]
int v120[32]; // [esp+Ch] [ebp-88h]
```

```
int v121[32]; // [esp+10h] [ebp-84h]
int v122[32]; // [esp+14h] [ebp-80h]
int _18[32]; // [esp+18h] [ebp-7Ch]
int _1C[32]; // [esp+1Ch] [ebp-78h]
int _20[32]; // [esp+20h] [ebp-74h]
int _24[32]; // [esp+24h] [ebp-70h]
int _28[32]; // [esp+28h] [ebp-6Ch]
int _2C[32]; // [esp+2Ch] [ebp-68h]
int _30[32]; // [esp+30h] [ebp-64h]
int _34[32]; // [esp+34h] [ebp-60h]
int _38[32]; // [esp+38h] [ebp-5Ch]
int _3C[32]; // [esp+3Ch] [ebp-58h]
int _40[32]; // [esp+40h] [ebp-54h]
int _44[32]; // [esp+44h] [ebp-50h]
int _48[32]; // [esp+48h] [ebp-4Ch]
int _4C[32]; // [esp+4Ch] [ebp-48h]
int _50[32]; // [esp+50h] [ebp-44h]
int _54[32]; // [esp+54h] [ebp-40h]
int _58[32]; // [esp+58h] [ebp-3Ch]
int _5C[32]; // [esp+5Ch] [ebp-38h]
int _60[32]; // [esp+60h] [ebp-34h]
int _64[32]; // [esp+64h] [ebp-30h]
int _68[32]; // [esp+68h] [ebp-2Ch]
int _6C[32]; // [esp+6Ch] [ebp-28h]
int _70[32]; // [esp+70h] [ebp-24h]
int _74[32]; // [esp+74h] [ebp-20h]
int _78[32]; // [esp+78h] [ebp-1Ch]
int _7C[32]; // [esp+7Ch] [ebp-18h]
int _80[32]; // [esp+80h] [ebp-14h]
int _84[32]; // [esp+84h] [ebp-10h]
int _88[32]; // [esp+88h] [ebp-Ch]
int v152; // [esp+8Ch] [ebp-8h]
```

```
for ( i = 0; i < 32; ++i )
v120[i] = 1;
v152 = 0;
puts("Your flag is:");
v3 = sub_401100(v120, 1000000000);
v4 = sub_401220(v3, 999999950);
sub_401100(v4, 2);
v5 = sub_401000(v121, 5000000);
v6 = sub_401220(v5, 6666666);
v7 = sub_401000(v6, 1666666);
v8 = sub_401000(v7, 45);
v9 = sub_401100(v8, 2);
sub_401000(v9, 5);
v10 = sub_401100(v122, 1000000000);
v11 = sub_401220(v10, 999999950);
v12 = sub_401100(v11, 2);
sub_401000(v12, 2);
v13 = sub_401000( 18, 55);
```

v14 = sub_401220(v13, 3);
v15 = sub_401000(v14, 4);
sub_401220(v15, 1);
v16 = sub_401100(_1C, 100000000);
v17 = sub_401220(v16, 99999950);
v18 = sub_401100(v17, 2);
sub_401000(v18, 2);
v19 = sub_401220(_20, 1);
v20 = sub_401100(v19, 1000000000);
v21 = sub_401000(v20, 55);
sub_401220(v21, 3);
v22 = sub_401100(_24, 1000000);
v23 = sub_401220(v22, 999975);
sub_401100(v23, 4);
v24 = sub_401000(_28, 55);
v25 = sub_401220(v24, 33);
v26 = sub_401000(v25, 44);
sub_401220(v26, 11);
v27 = sub_401100(_2C, 10);
v28 = sub_401220(v27, 5);
v29 = sub_401100(v28, 8);
sub_401000(v29, 9);
v30 = sub_401000(_30, 0);
v31 = sub_401220(v30, 0);
v32 = sub_401000(v31, 11);
v33 = sub_401220(v32, 11);
sub_401000(v33, 53);
v34 = sub_401000(_34, 49);
v35 = sub_401220(v34, 2);
v36 = sub_401000(v35, 4);
sub_401220(v36, 2);
v37 = sub_401100(_38, 1000000);
v38 = sub_401220(v37, 999999);
v39 = sub_401100(v38, 4);
sub_401000(v39, 50);
v40 = sub_401000(_3C, 1);
v41 = sub_401000(v40, 1);
v42 = sub_401000(v41, 1);
v43 = sub_401000(v42, 1);
v44 = sub_401000(v43, 1);
v45 = sub_401000(v44, 1);
v46 = sub_401000(v45, 10);
sub_401000(v46, 32);
v47 = sub_401100(_40, 10);
v48 = sub_401220(v47, 5);
v49 = sub_401100(v48, 8);
v50 = sub_401000(v49, 9);
sub_401000(v50, 48);
v51 = sub_401220(_44, 1);
v52 = sub_401100(v51, -294967296);

v53 = sub_401000(v52, 55);
sub_401220(v53, 3);
v54 = sub_401000(_48, 1);
v55 = sub_401000(v54, 2);
v56 = sub_401000(v55, 3);
v57 = sub_401000(v56, 4);
v58 = sub_401000(v57, 5);
v59 = sub_401000(v58, 6);
v60 = sub_401000(v59, 7);
sub_401000(v60, 20);
v61 = sub_401100(_4C, 10);
v62 = sub_401220(v61, 5);
v63 = sub_401100(v62, 8);
v64 = sub_401000(v63, 9);
sub_401000(v64, 48);
v65 = sub_401000(_50, 7);
v66 = sub_401000(v65, 6);
v67 = sub_401000(v66, 5);
v68 = sub_401000(v67, 4);
v69 = sub_401000(v68, 3);
v70 = sub_401000(v69, 2);
v71 = sub_401000(v70, 1);
sub_401000(v71, 20);
v72 = sub_401000(_54, 7);
v73 = sub_401000(v72, 2);
v74 = sub_401000(v73, 4);
v75 = sub_401000(v74, 3);
v76 = sub_401000(v75, 6);
v77 = sub_401000(v76, 5);
v78 = sub_401000(v77, 1);
sub_401000(v78, 20);
v79 = sub_401100(_58, 1000000);
v80 = sub_401220(v79, 999999);
v81 = sub_401100(v80, 4);
v82 = sub_401000(v81, 50);
sub_401220(v82, 1);
v83 = sub_401220(_5C, 1);
v84 = sub_401100(v83, -294967296);
v85 = sub_401000(v84, 49);
sub_401220(v85, 1);
v86 = sub_401220(_60, 1);
v87 = sub_401100(v86, 1000000000);
v88 = sub_401000(v87, 54);
v89 = sub_401220(v88, 1);
v90 = sub_401000(v89, 1000000000);
sub_401220(v90, 1000000000);
v91 = sub_401000(_64, 49);
v92 = sub_401220(v91, 1);
v93 = sub_401000(v92, 2);
sub_401220(v93, 1);

```

v94 = sub_401100(_68, 10);
v95 = sub_401220(v94, 5);
v96 = sub_401100(v95, 8);
v97 = sub_401000(v96, 9);
sub_401000(v97, 48);
v98 = sub_401000(_6C, 1);
v99 = sub_401000(v98, 3);
v100 = sub_401000(v99, 3);
v101 = sub_401000(v100, 3);
v102 = sub_401000(v101, 6);
v103 = sub_401000(v102, 6);
v104 = sub_401000(v103, 6);
sub_401000(v104, 20);
v105 = sub_401000(_70, 55);
v106 = sub_401220(v105, 33);
v107 = sub_401000(v106, 44);
v108 = sub_401220(v107, 11);
sub_401000(v108, 42);
sub_401000(_74, _70[0]);
sub_401000(_78, _3C[0]);
v109 = _78[0];
v110 = sub_401220(_7C, 1);
v111 = sub_401000(v110, v109);
sub_401220(v111, 1);
v112 = _68[0];
v113 = sub_401220(_80, 1);
v114 = sub_401100(v113, 1000000);
sub_401000(v114, v112);
v115 = _78[0];
v116 = sub_401000(_84, 1);
sub_401100(v116, v115);
sub_401000(_88, _84[0]);
((void (__cdecl*)(const char *, signed int))sub_401C7F)("CTF{", 1);
for (j = 0; j < 32; ++j )
sub_401C7F("%c", SLOBYTE(v120[j]));
sub_401C7F("}\n");
return 0;
}'

```

发现主要是sub_401100,sub_401000, sub_401220三个函数的反复使用，分别进入这三个函数么，首先是sub_401100

```

_DWORD *__cdecl sub_401100(_DWORD *a1, int a2)
{
    int v2; // ST20_4
    signed int v4; // [esp+Ch] [ebp-1Ch]
    int v5; // [esp+14h] [ebp-14h]
    int v6; // [esp+18h] [ebp-10h]
    int v7; // [esp+1Ch] [ebp-Ch]
    int v8; // [esp+20h] [ebp-8h]

    v5 = *a1;
    v6 = a2;
    v4 = -1;
    v8 = 0;
    v7 = a2 * v5;
}

```

```

while ( a2 )
{
    v2 = v7 * v5;
    sub_401000(&v8, *a1);
    ++v7;
    --a2;
    v6 = v2 - 1;
}
while ( v4 )
{
    ++v7;
    ++*a1;
    --v4;
    --v6;
}
++*a1;
*a1 = v8;
return a1;
}

```

https://blog.csdn.net/weixin_49700990

发现需要先搞定sub_401000这个函数，说干就干

```

1 int *__cdecl sub_401000(int *a1, int a2)
2 {
3     int v2; // edx
4     int v4; // [esp+Ch] [ebp-18h]
5     signed int v5; // [esp+10h] [ebp-14h]
6     int v6; // [esp+18h] [ebp-Ch]
7     signed int v7; // [esp+1Ch] [ebp-8h]
8
9     v5 = -1;
10    v4 = -1 - a2 + 1;
11    v7 = 1231;
12    v2 = *a1;
13    v6 = a2 + 1231;
14    while ( v4 )
15    {
16        ++v7;
17        --*a1;
18        --v4;
19        --v6;
20    }
21    while ( v5 )
22    {
23        --v6;
24        ++*a1;
25        --v5;
26    }
27    ++*a1;
28    return a1;
29 }

```

https://blog.csdn.net/weixin_49700990

看到两个循环，因为计算机存储整数使用的是“补码”的机制，所以导致了“溢出”的情况下，奇怪的现象。

简单的例子

char 一般是一个字节，表示-128~127之间的整数，那么

-128-1——>127

127+1——>-128

利用这个原理，试一试发现第一个循环就是 $a1+a2$ （不考虑指针的格式，以下都是这样表述），第二个循环则是 $a1-1$ ，循环后有一个 $++a1$ ，综上所述`sub_401000(int a,int b)`返回的是 $a+b$ 的值

再回到`sub_401100`函数中，分析后可以得到`sub_401100 (int a,int b)` 返回的是 $a*b$ 的值

同理`sub_401220 (int a, int b)` 返回的是 $a-b$ 的值

写代码

```
//#include<iostream>
#include<stdio.h>
//using namespace std;
int sub_401100(int a, int b) {
    return a * b;
}
int sub_401000(int a, int b) {
    return a + b;
}
int sub_401220(int a, int b) {
    return a - b;
}
```

```
int main() {
    int v3; // eax
    int v4; // eax
    int v5; // eax
    int v6; // eax
    int v7; // eax
    int v8; // eax
    int v9; // eax
    int v10; // eax
    int v11; // eax
    int v12; // eax
    int v13; // eax
    int v14; // eax
    int v15; // eax
    int v16; // eax
    int v17; // eax
    int v18; // eax
    int v19; // eax
    int v20; // eax
    int v21; // eax
    int v22; // eax
    int v23; // eax
    int v24; // eax
    int v25; // eax
    int v26; // eax
    int v27; // eax
    int v28; // eax
    int v29; // eax
    int v30; // eax
    int v31; // eax
    int v32; // eax
    int v33; // eax
    int v34; // eax
    int v35; // eax
    int v36; // eax
    int v37; // eax
}
```

```
int v37; // eax
int v38; // eax
int v39; // eax
int v40; // eax
int v41; // eax
int v42; // eax
int v43; // eax
int v44; // eax
int v45; // eax
int v46; // eax
int v47; // eax
int v48; // eax
int v49; // eax
int v50; // eax
int v51; // eax
int v52; // eax
int v53; // eax
int v54; // eax
int v55; // eax
int v56; // eax
int v57; // eax
int v58; // eax
int v59; // eax
int v60; // eax
int v61; // eax
int v62; // eax
int v63; // eax
int v64; // eax
int v65; // eax
int v66; // eax
int v67; // eax
int v68; // eax
int v69; // eax
int v70; // eax
int v71; // eax
int v72; // eax
int v73; // eax
int v74; // eax
int v75; // eax
int v76; // eax
int v77; // eax
int v78; // eax
int v79; // eax
int v80; // eax
int v81; // eax
int v82; // eax
int v83; // eax
int v84; // eax
int v85; // eax
int v86; // eax
int v87; // eax
int v88; // eax
int v89; // eax
int v90; // eax
int v91; // eax
int v92; // eax
int v93; // eax
int v94; // eax
int v95; // eax
int v96; // eax
```

```

int v97; // eax
int v98; // eax
int v99; // eax
int v100; // eax
int v101; // eax
int v102; // eax
int v103; // eax
int v104; // eax
int v105; // eax
int v106; // eax
int v107; // eax
int v108; // eax
int v109; // ST1C_4
int v110; // eax
int v111; // eax
int v112; // ST20_4
int v113; // eax
int v114; // eax
int v115; // ST20_4
int v116; // eax
signed int i; // [esp+4h] [ebp-90h]
signed int j; // [esp+8h] [ebp-8Ch]
int v120[32]; // [esp+Ch] [ebp-88h]
int v121; // [esp+8Ch] [ebp-8h]
for (i = 0; i < 32; ++i)
    v120[i] = 1;
v121 = 0;
puts("Your flag is:");
v3 = sub_401100(v120[0], 1000000000);
v4 = sub_401220(v3, 999999950);
v120[0]=sub_401100(v4, 2);
v5 = sub_401000(v120[1], 5000000);
v6 = sub_401220(v5, 6666666);
v7 = sub_401000(v6, 1666666);
v8 = sub_401000(v7, 45);
v9 = sub_401100(v8, 2);
v120[1] = sub_401000(v9, 5);
v10 = sub_401100(v120[2], 1000000000);
v11 = sub_401220(v10, 999999950);
v12 = sub_401100(v11, 2);
v120[2] = sub_401000(v12, 2);
v13 = sub_401000(v120[3], 55);
v14 = sub_401220(v13, 3);
v15 = sub_401000(v14, 4);
v120[3] = sub_401220(v15, 1);
v16 = sub_401100(v120[4], 1000000000);
v17 = sub_401220(v16, 999999950);
v18 = sub_401100(v17, 2);
v120[4] = sub_401000(v18, 2);
v19 = sub_401220(v120[5], 1);
v20 = sub_401100(v19, 1000000000);
v21 = sub_401000(v20, 55);
v120[5] = sub_401220(v21, 3);
v22 = sub_401100(v120[6], 1000000);
v23 = sub_401220(v22, 999975);
v120[6] = sub_401100(v23, 4);
v24 = sub_401000(v120[7], 55);
v25 = sub_401220(v24, 33);
v26 = sub_401000(v25, 44);

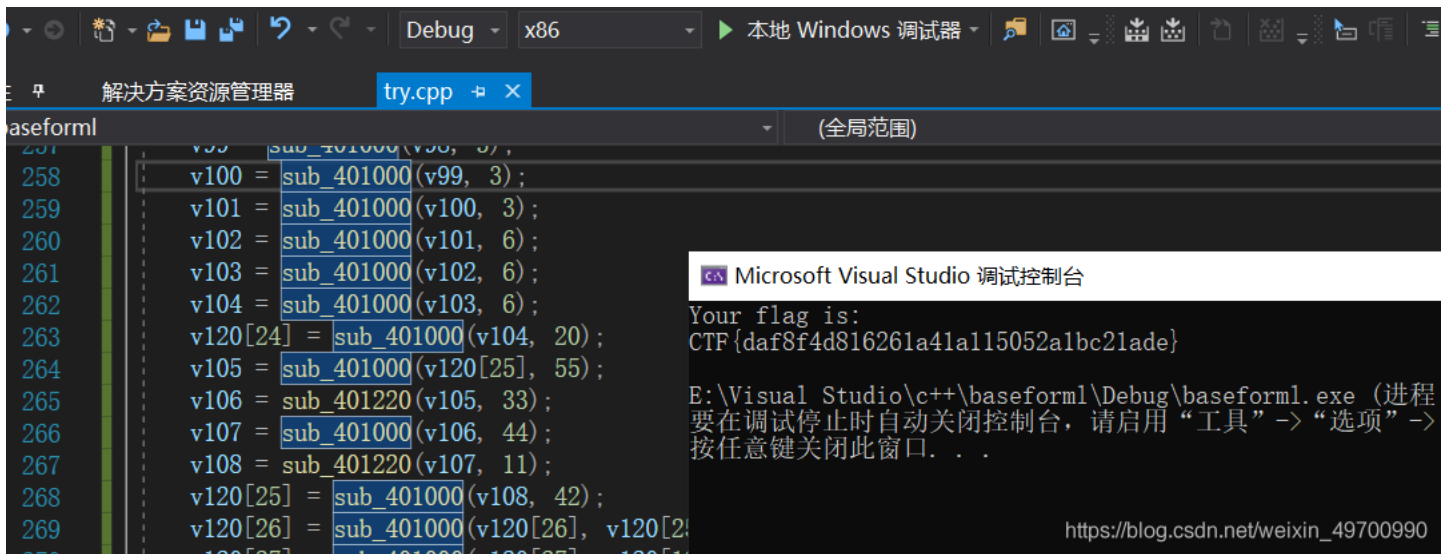
```

```
v120[7] = sub_401220(v26, 11);
v27 = sub_401100(v120[8], 10);
v28 = sub_401220(v27, 5);
v29 = sub_401100(v28, 8);
v120[8] = sub_401000(v29, 9);
v30 = sub_401000(v120[9], 0);
v31 = sub_401220(v30, 0);
v32 = sub_401000(v31, 11);
v33 = sub_401220(v32, 11);
v120[9] = sub_401000(v33, 53);
v34 = sub_401000(v120[10], 49);
v35 = sub_401220(v34, 2);
v36 = sub_401000(v35, 4);
v120[10] = sub_401220(v36, 2);
v37 = sub_401100(v120[11], 1000000);
v38 = sub_401220(v37, 999999);
v39 = sub_401100(v38, 4);
v120[11] = sub_401000(v39, 50);
v40 = sub_401000(v120[12], 1);
v41 = sub_401000(v40, 1);
v42 = sub_401000(v41, 1);
v43 = sub_401000(v42, 1);
v44 = sub_401000(v43, 1);
v45 = sub_401000(v44, 1);
v46 = sub_401000(v45, 10);
v120[12] = sub_401000(v46, 32);
v47 = sub_401100(v120[13], 10);
v48 = sub_401220(v47, 5);
v49 = sub_401100(v48, 8);
v50 = sub_401000(v49, 9);
v120[13] = sub_401000(v50, 48);
v51 = sub_401220(v120[14], 1);
v52 = sub_401100(v51, -294967296);
v53 = sub_401000(v52, 55);
v120[14] = sub_401220(v53, 3);
v54 = sub_401000(v120[15], 1);
v55 = sub_401000(v54, 2);
v56 = sub_401000(v55, 3);
v57 = sub_401000(v56, 4);
v58 = sub_401000(v57, 5);
v59 = sub_401000(v58, 6);
v60 = sub_401000(v59, 7);
v120[15] = sub_401000(v60, 20);
v61 = sub_401100(v120[16], 10);
v62 = sub_401220(v61, 5);
v63 = sub_401100(v62, 8);
v64 = sub_401000(v63, 9);
v120[16] = sub_401000(v64, 48);
v65 = sub_401000(v120[17], 7);
v66 = sub_401000(v65, 6);
v67 = sub_401000(v66, 5);
v68 = sub_401000(v67, 4);
v69 = sub_401000(v68, 3);
v70 = sub_401000(v69, 2);
v71 = sub_401000(v70, 1);
v120[17] = sub_401000(v71, 20);
v72 = sub_401000(v120[18], 7);
v73 = sub_401000(v72, 2);
v74 = sub_401000(v73, 4);
v75 = sub_401000(v74, 3);
```

```
v76 = sub_401000(v75, 6);
v77 = sub_401000(v76, 5);
v78 = sub_401000(v77, 1);
v120[18] = sub_401000(v78, 20);
v79 = sub_401100(v120[19], 1000000);
v80 = sub_401220(v79, 999999);
v81 = sub_401100(v80, 4);
v82 = sub_401000(v81, 50);
v120[19] = sub_401220(v82, 1);
v83 = sub_401220(v120[20], 1);
v84 = sub_401100(v83, -294967296);
v85 = sub_401000(v84, 49);
v120[20] = sub_401220(v85, 1);
v86 = sub_401220(v120[21], 1);
v87 = sub_401100(v86, 1000000000);
v88 = sub_401000(v87, 54);
v89 = sub_401220(v88, 1);
v90 = sub_401000(v89, 1000000000);
v120[21] = sub_401220(v90, 1000000000);
v91 = sub_401000(v120[22], 49);
v92 = sub_401220(v91, 1);
v93 = sub_401000(v92, 2);
v120[22] = sub_401220(v93, 1);
v94 = sub_401100(v120[23], 10);
v95 = sub_401220(v94, 5);
v96 = sub_401100(v95, 8);
v97 = sub_401000(v96, 9);
v120[23] = sub_401000(v97, 48);
v98 = sub_401000(v120[24], 1);
v99 = sub_401000(v98, 3);
v100 = sub_401000(v99, 3);
v101 = sub_401000(v100, 3);
v102 = sub_401000(v101, 6);
v103 = sub_401000(v102, 6);
v104 = sub_401000(v103, 6);
v120[24] = sub_401000(v104, 20);
v105 = sub_401000(v120[25], 55);
v106 = sub_401220(v105, 33);
v107 = sub_401000(v106, 44);
v108 = sub_401220(v107, 11);
v120[25] = sub_401000(v108, 42);
v120[26] = sub_401000(v120[26], v120[25]);
v120[27] = sub_401000(v120[27], v120[12]);
v109 = v120[27];
v110 = sub_401220(v120[28], 1);
v111 = sub_401000(v110, v109);
v120[28] = sub_401220(v111, 1);
v112 = v120[23];
v113 = sub_401220(v120[29], 1);
v114 = sub_401100(v113, 1000000);
v120[29] = sub_401000(v114, v112);
v115 = v120[27];
v116 = sub_401000(v120[30], 1);
v120[30] = sub_401100(v116, v115);
v120[31] = sub_401000(v120[31], v120[30]);
printf("CTF{");
for (j = 0; j < 32; ++j)
    printf("%c", v120[j]);
printf("}\n");
```

```
return 0;
```

```
}
```



运行得出: CTF{daf8f4d816261a41a115052a1bc21ade}