gravity_ball

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v4; // [esp+13h] [ebp-75h] BYREF
  _DWORD v5[24]; // [esp+17h] [ebp-71h] BYREF
  char v6; // [esp+77h] [ebp-11h]
  int v7; // [esp+78h] [ebp-10h]
  char v8; // [esp+7Fh] [ebp-9h]

  __main();
  puts("welcome to ez Grivity Ball game!");
  puts("really really ez,i wont cheat u");
  v4 = 0;
  v5[0] = 0;
  v5[23] = 0;
  memset((char *)v5 + 1, 0, 4 * ((((char *)v5 - ((char *)v5 + 1) + 96) & 0xFFFFFFFC) >> 2));
  v8 = 0;
  v7 = 1;
  while ( 1 )
  {
    do
      v6 = getchar();
    while ( v6 == '\n' );
    *((_BYTE *)&v4 + v8++) = v6;
    v7 = inputCheck(v6, v7);
    if ( !v7 )
      break;
    move(v7);
    if ( success )
    {
      printMD5Flag(&v4);
      return 0;
    }
  }
```

先进入main函数。先puts两个字符串，然后进入while循环，一个一个字符读取（跳过换行符），接着就是一个inputcheck函数

```
int __cdecl inputCheck(char a1, int a2)
{
  int v3; // [esp+34h] [ebp+Ch]

  v3 = a2 - 1;
  if ( a1 == 'e' )
    return (v3 + 3) % 4 + 1;
  if ( a1 == 'q' )
    return (v3 + 1) % 4 + 1;
  printf("Illegal input");
  return 0;
}
```

表示输入只能为e、q。输出为1-4的其中一个数，如果是e，输入4，输出3；输入3，输出2；2变1；1变回4。q则相反，比4小则+1，是4则变回1。

输入变量为V7，V7初始值为1。

通过检测后，是move函数。

```
void __cdecl move(int a1)
{
  if ( a1 == 4 )
  {
    do
      ++X;
    while ( symbolCheck() );
    --X;
  }
  else if ( a1 <= 4 )
  {
    switch ( a1 )
    {
      case 3:
        do
          --Y;
        while ( symbolCheck() );
        ++Y;
        break;
      case 1:
        do
          ++Y;
        while ( symbolCheck() );
        --Y;
        break;
      case 2:
        do
          --X;
        while ( symbolCheck() );
        ++X;
        break;
    }
  }
```

根据之前输入的qe的返回值 1 2 3 4，move函数会对X Y进行不同的操作。

1：Y一直增加，直到symbolcheck返回0 2:X一直减，直到symbolcheck返回0 3和4 同理。

再看symbolcheck函数

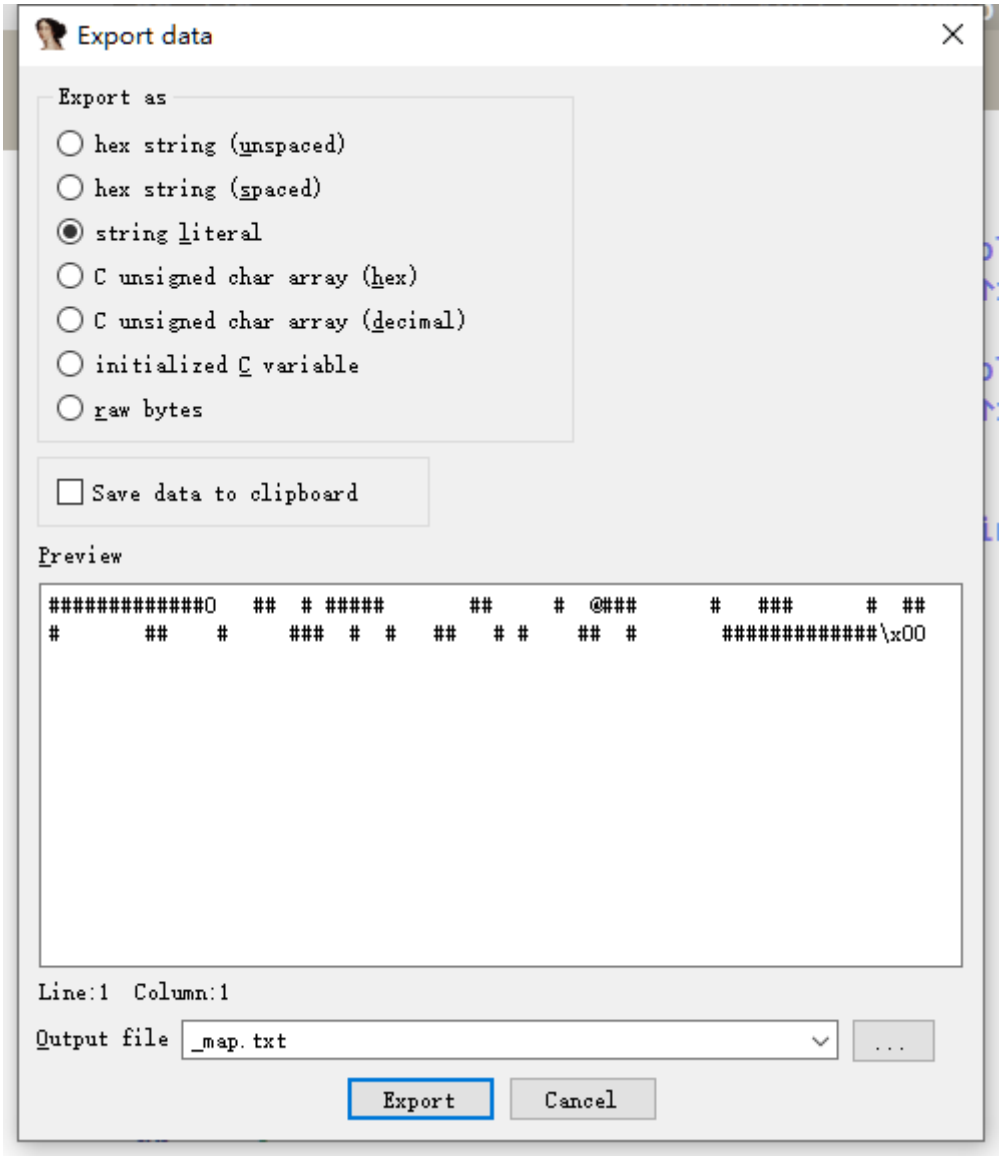
```
int symbolCheck()
{
  int v0; // eax
  int result; // eax

  v0 = map[12 * Y + X];
  if ( v0 == 'O' )
  {
    puts("WHY YOU CAME BACK?!");
    result = 0;
  }
  else
  {
    if ( v0 <= 'O' )
    {
      if ( v0 == '@' )
      {
        success = 1;
        return 0;
      }
      if ( v0 <= '@' )
      {
        if ( v0 == ' ' )
        {
          success = 0;
          return 1;
        }
        if ( v0 == '#' )
        {
          puts("Dong!you hear the ball hitting the wall");
          return 0;
        }
      }
```

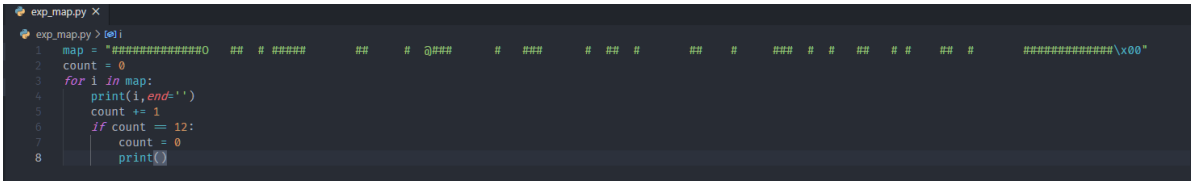
看到一个map，可以猜测是地图。并且Y*12，说明很有可能地图的宽度是12。

```
ta:00406020 ; char map[]
ta:00406020 _map            db 23h                  ; DATA XREF: _symbolCheck+1E↑r
ta:00406021 a0              db '############O   ##  # #####       ##     #  @###      #   ###    '
ta:00406021                 db '  #  ##  #       ##    #     ###  #  #   ##   # #    ##  #       '
ta:00406021                 db '############',0
ta:004060B1                 align 4
```

选中 shift + E



将地图12个为一行处理：



可以得到地图

然后经过找规律可以得到，每次q相当于将整个地图逆时针旋转，e相当于对整个地图顺时针旋转，然后每次旋转完球就会往重力方向一直移动，直到碰到'#'符号。

到此，即使上面代码不用看也能很快猜到，@和O其中一个可能是起点，另一个可能是终点。

再看X Y变量的初始值：

```
.data:004060B4                  public _X
.data:004060B4 _X               dd 1                    ; DATA XREF: _symbolCheck+17↑r
.data:004060B4                                          ; _move:loc_401570↑r ...
.data:004060B8                  public _Y
.data:004060B8 _Y               dd 1                    ; DATA XREF: _symbolCheck+6↑r
.data:004060B8                                          ; _move:loc_40154A↑r ...
.data:004060BC                  align 10h
```

初始值X Y 都为1，那么很显然初始位置就是O的位置，而我们要去的是@位置。

接下来就是玩游戏时间辣！（不要怪我地图太阴间，本来做的很简单，是阿翔让我改的，别打我！）

当symbolcheck检测到当前位置是@的时候，就会把success置1，执行printMD5flag函数。

```
1  int __cdecl printMD5Flag(char *a1)
2  {
3    int v1; // eax
4    int v3[22]; // [esp+10h] [ebp-78h] BYREF
5    char v4[16]; // [esp+68h] [ebp-20h] BYREF
6    char *Str; // [esp+78h] [ebp-10h]
7    int i; // [esp+7Ch] [ebp-Ch]
8
9    printf("here is your flag:");
10   Str = a1;
11   MD5Init(v3);
12   v1 = strlen(Str);
13   MD5Update((int)v3, Str, v1);
14   MD5Final(v3, v4);
15   printf("flag{");
16   for ( i = 0; i <= 15; ++i )
17     printf("%02x", (unsigned __int8)v4[i]);
18   puts("}");
19   return getchar();
20 }
```

这个函数不用看，实际上就是对输入的路径进行md5加密，加密结果就是flag。

只有正确的最短路径才是正确flag。

答案：路径eqqqqeeeeqqeeeeqeee

```
here is your flag:flag{a0b25d81bc6d09e799ae8967efdac129}
```

> flag{a0b25d81bc6d09e799ae8967efdac129}