使用實際例子C++ 確認位元運算子，是屬於何種Shift Operation

　　由圖一及圖二可知，若將3及-3向左位移LSB 會持續補零，因此可以推論C++所使用的Shift Operation 不會是Circular Shift ，而目前是使用Logical Shift 或是Arithmetic Shift 還無法推論，因為在left shift 的方式皆為相同的，所以必須要再使用right shift 來推論。

　　圖三為3進行right shift 的執行畫面，因其值為正數，在進行right shift 時無法得知運算時MSB的補值是因為3 為不代符號的關係而使用Arithmetic Shift，或是Logical Shift。

Arithmetic Shift 及Logical Shift 最大的差異在於right shift 時是否維持MSB的數值。由圖四中，可以得知，若先定義一個負數，將其作right shift ，MSB會自動將sign補齊。為了方便觀察數字變化，因此我將數字定義為

-32767 ，如圖五。

由此實驗可知C++使用的shift operation 為Arithmetic Shift。

1.

#include <iostream>

using namespace std;

int main ()

{

short a=3; //3=0000000000000011

for(int i=1;i<=16;i++)

{

a=a<<1;

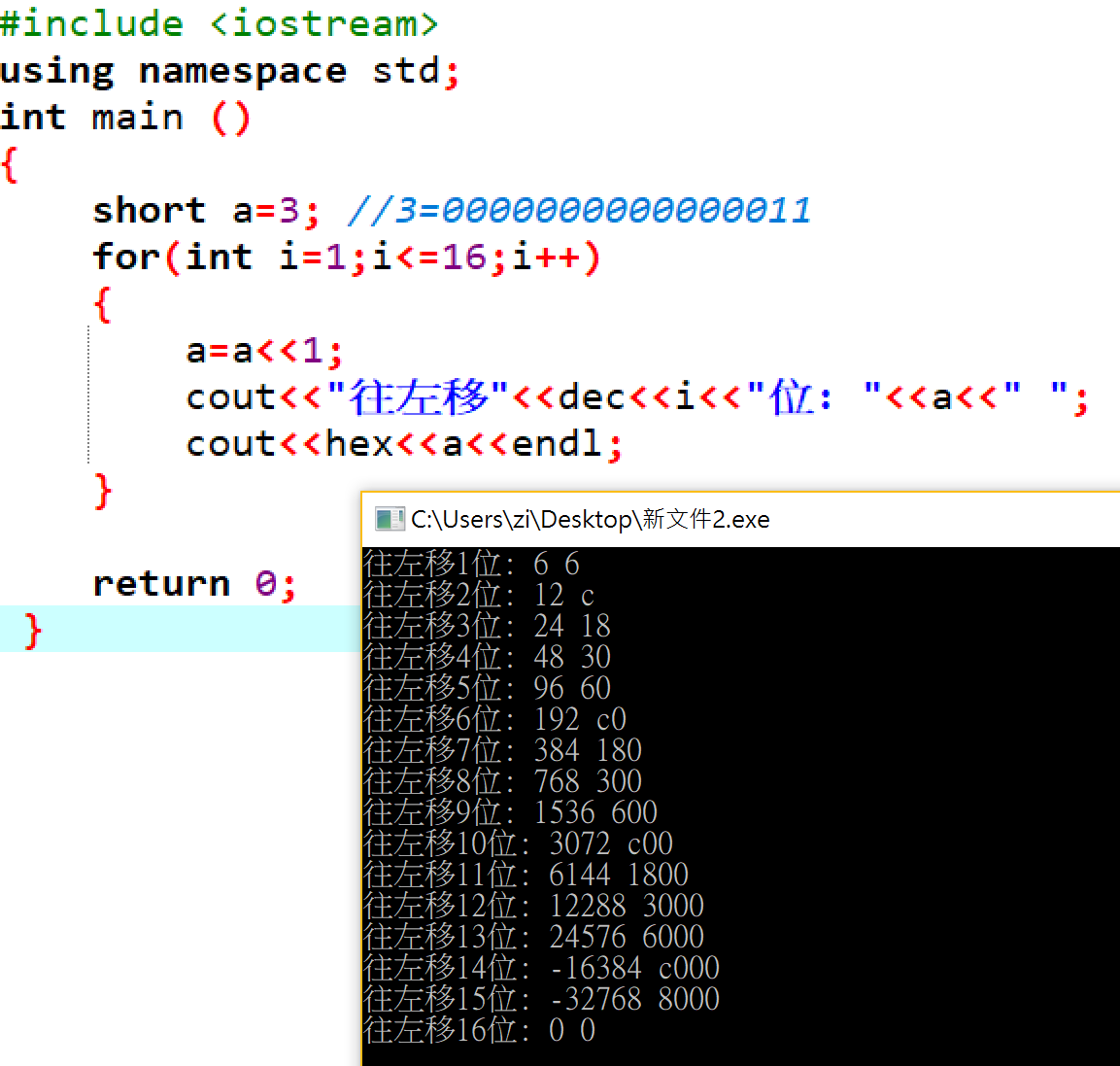
cout<<"往左移"<<dec<<i<<"位: "<<a<<" ";

cout<<hex<<a<<endl;

}

return 0;

}



圖一、3 left shift

2.

#include <iostream>

using namespace std;

int main ()

{

short a=-3; //-3=1111 1111 1111 1101

for(int i=1;i<=16;i++)

{

a=a<<1;

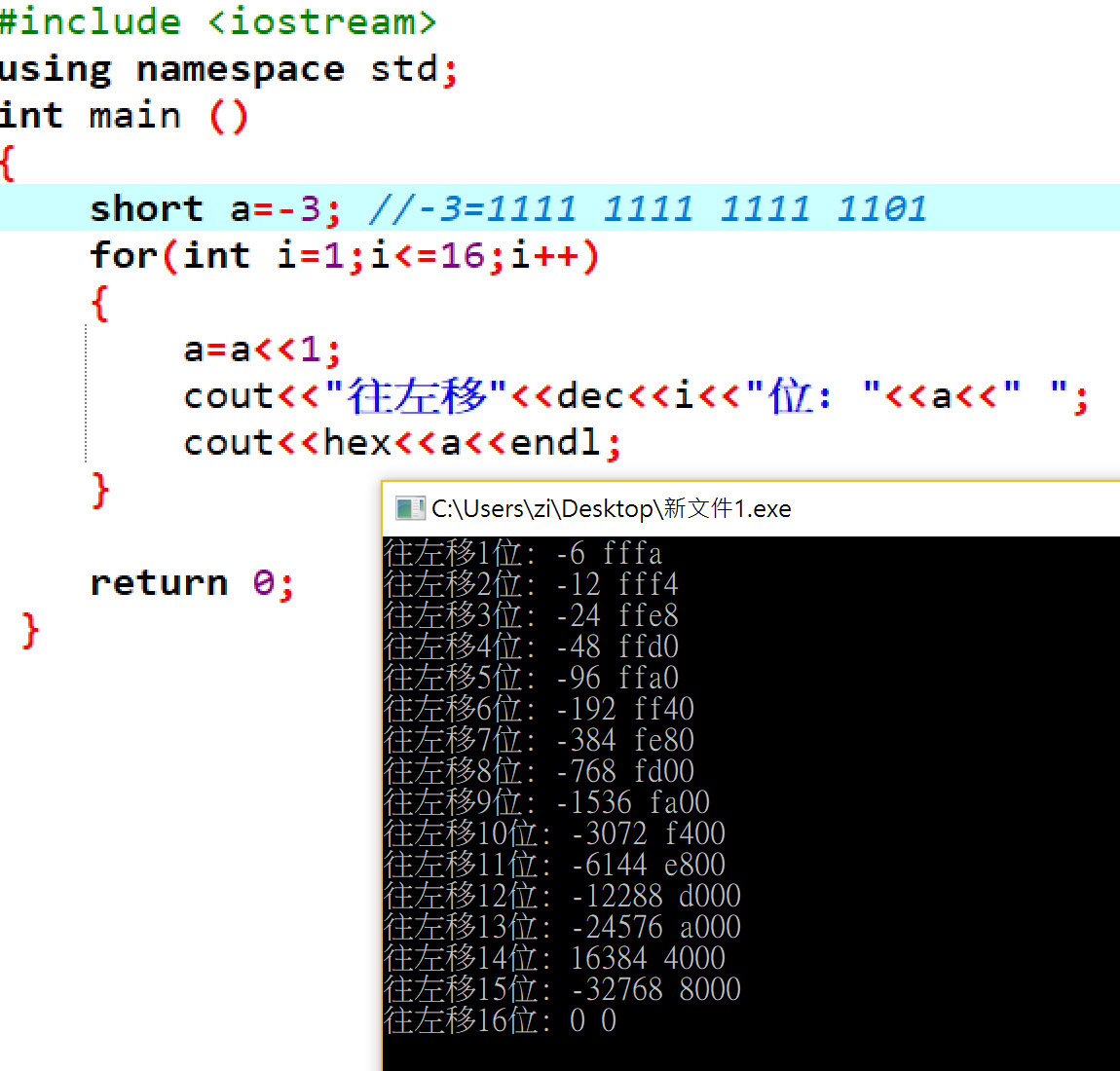
cout<<"往左移"<<dec<<i<<"位: "<<a<<" ";

cout<<hex<<a<<endl;

}

return 0;

}



圖二、-3 left shift

3.

#include <iostream>

using namespace std;

int main ()

{

short a=3; //3=0000000000000011

for(int i=1;i<=16;i++)

{

a=a>>1;

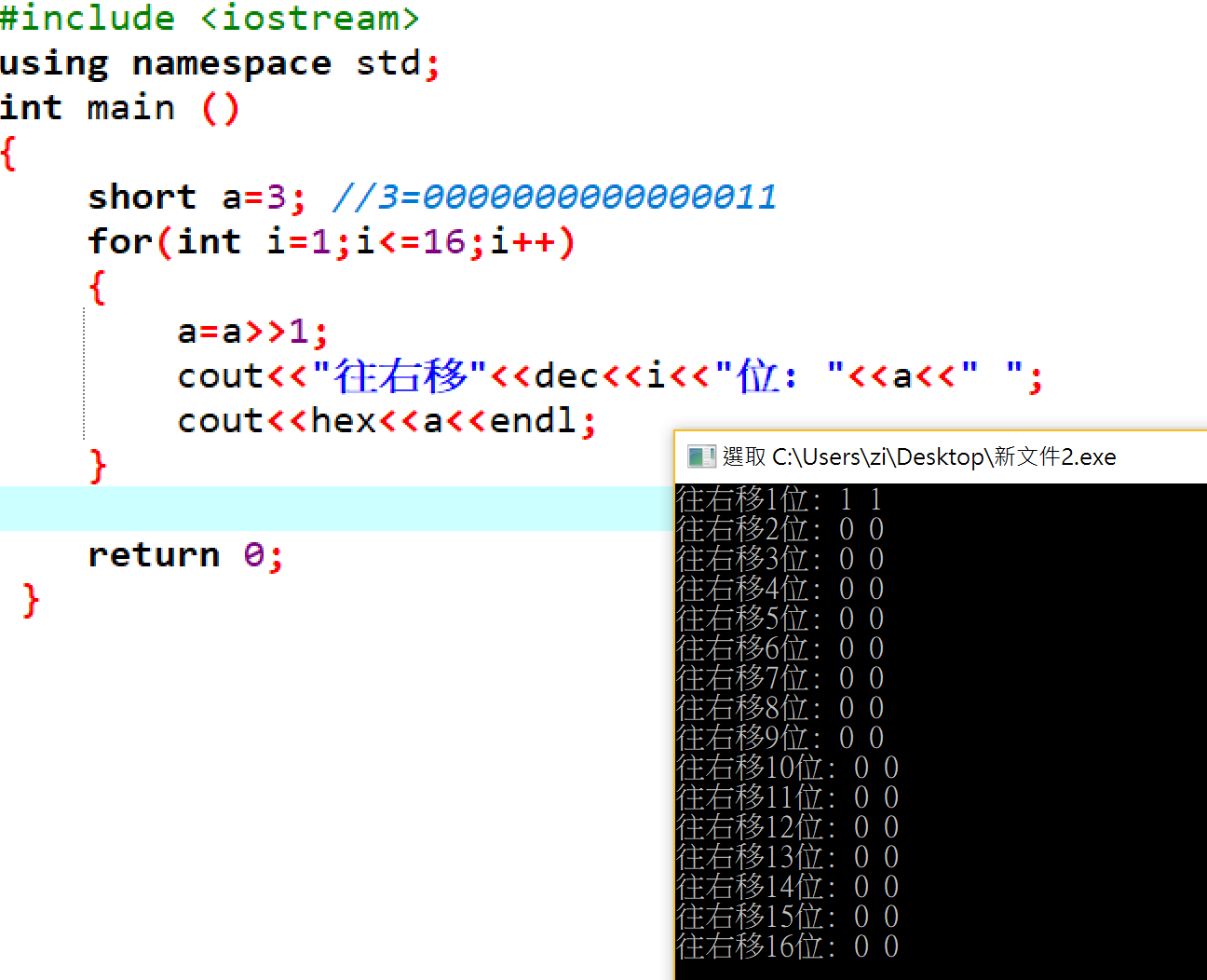
cout<<"往右移"<<dec<<i<<"位: "<<a<<" ";

cout<<hex<<a<<endl;

}

return 0;

}



圖三、3 right shift

4.

#include <iostream>

using namespace std;

int main ()

{

short a=-3; //-3=1111 1111 1111 1101

for(int i=1;i<=16;i++)

{

a=a>>1;

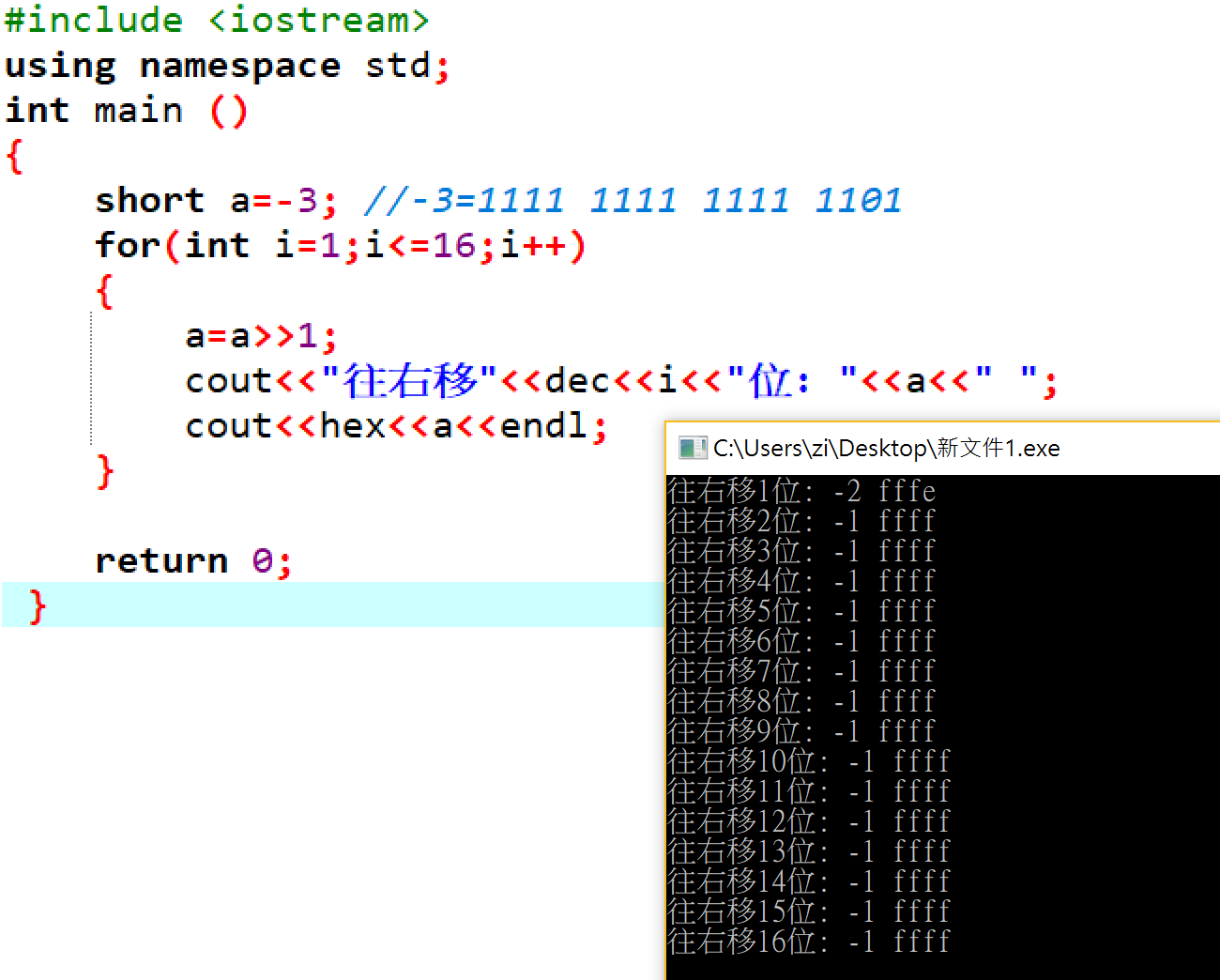
cout<<"往右移"<<dec<<i<<"位: "<<a<<" ";

cout<<hex<<a<<endl;

}

return 0;

}



圖四、-3 right shift

5.

#include <iostream>

using namespace std;

int main ()

{

short a=-32767; //32767=0111 1111 1111 1111 -32767=1000 0000 0000 0001

for(int i=1;i<=16;i++)

{

a=a>>1;

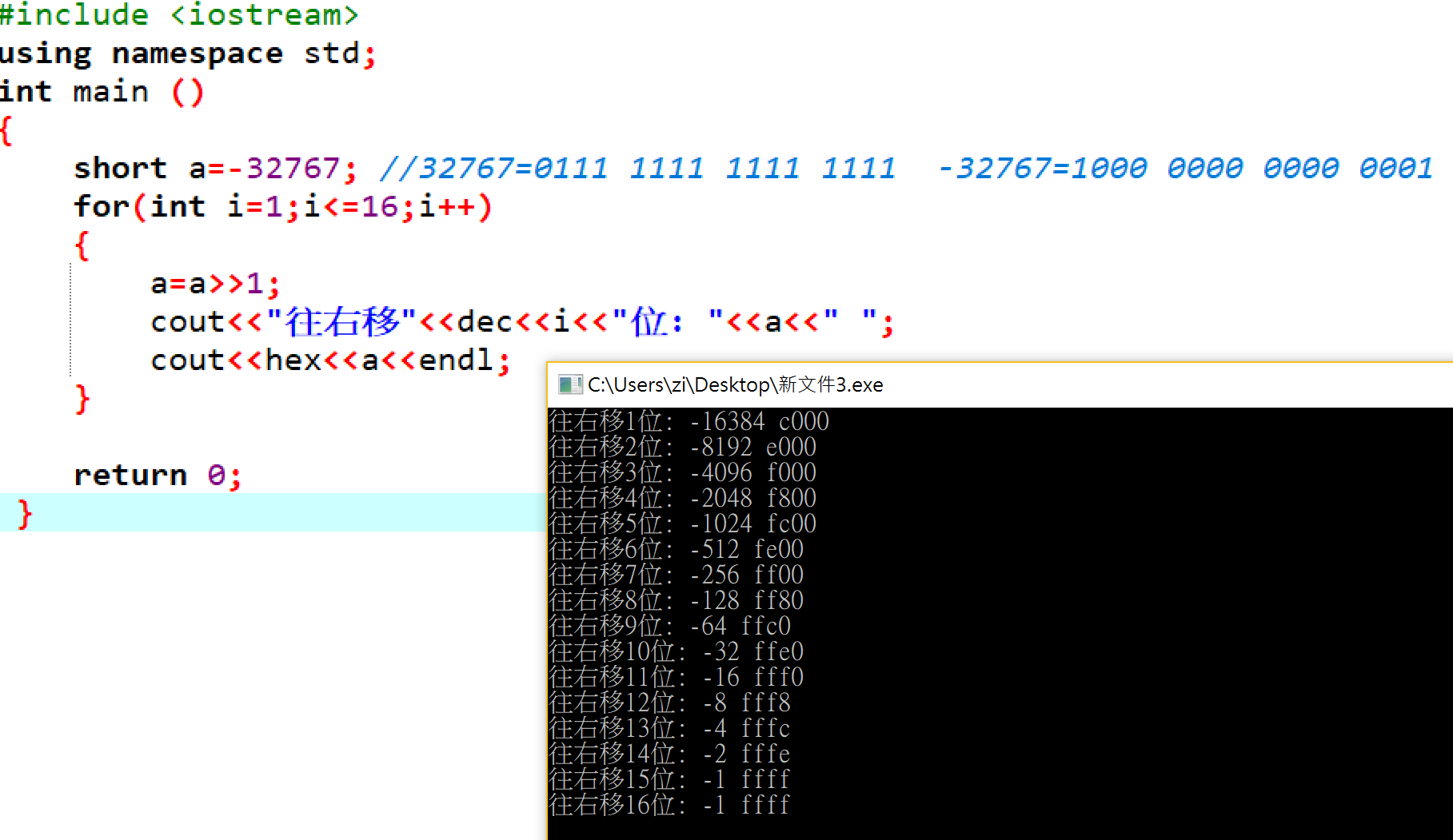
cout<<"往右移"<<dec<<i<<"位: "<<a<<" ";

cout<<hex<<a<<endl;

}

return 0;

}



圖五、-32767 right shift