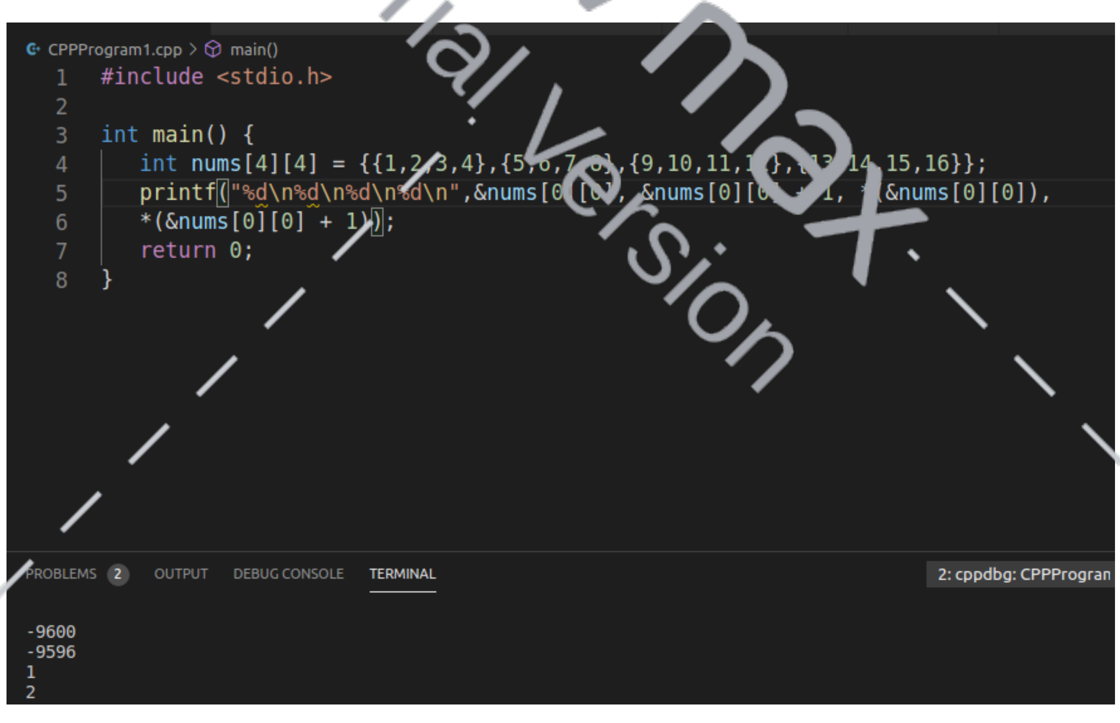
二维数组a[4][4]

&a[0][0]是取数组a第一个元素的地址,同时等价于*a

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]
a[3][0]	a[3][1]	a[3][2]	a[3][3]



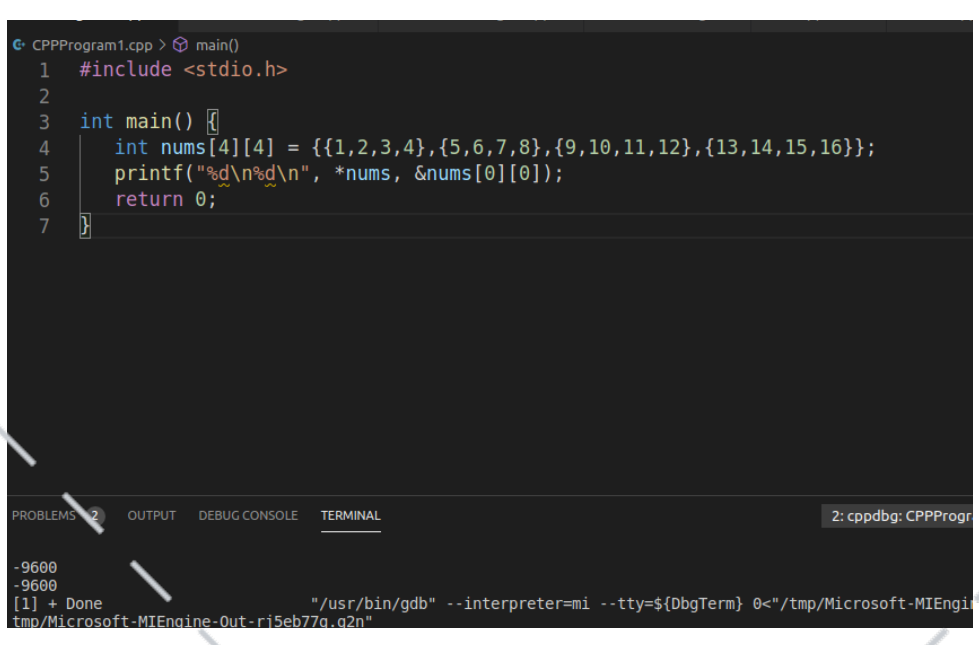
```
CPPProgram1.cpp > main()
#include <stdio.h>

int main() {
    int nums[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
    printf("%d\n%d\n%d\n%d\n",&nums[0][0], &nums[0][0] + 1, *(&nums[0][0]),
    *(&nums[0][0] + 1));
    return 0;
}
```

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL          2: cppdbg: CPPProgram

-9600
-9596
1
2
```

从上面这个例子可以看出&a[0][0] + 1后，指针到达数组中第二个数字的位置，即a[0][1],同时，我们还可以注意到地址作+1运算后，实际的内存地址+4，原因是int类型的数组.



```
CPPProgram1.cpp > main()
#include <stdio.h>

int main() {
    int nums[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
    printf("%d\n%d\n", *nums, &nums[0][0]);
    return 0;
}
```

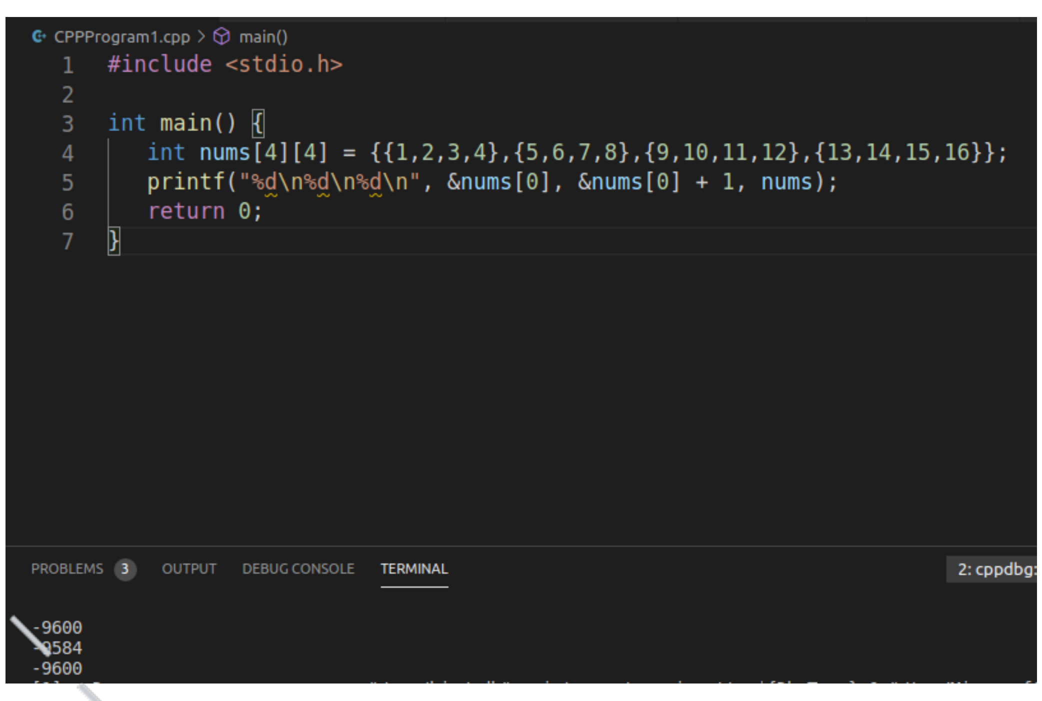
```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL          2: cppdbg: CPPProgr

-9600
-9600
[1] + Done                    "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngi
tmp/Microsoft-MIEngine-Out-rj5eb77q.g2n"
```

&a[0][0]等价于*a

a[0]相当于a数组的第0行数组,他相当于一个object.&a[0]相当于取a[0]这个object的整个地址,同时他们也等价于a.

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]
a[3][0]	a[3][1]	a[3][2]	a[3][3]



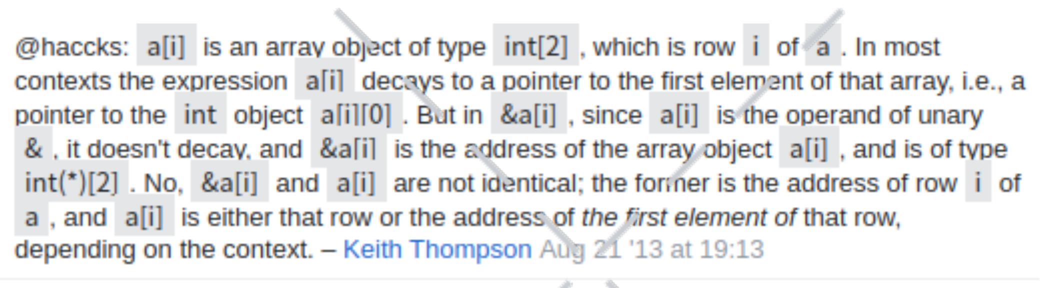
```
CPPProgram1.cpp > main()
#include <stdio.h>

int main() {
    int nums[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
    printf("%d\n%d\n%d\n", &nums[0], &nums[0] + 1, nums);
    return 0;
}
```

```
PROBLEMS 3   OUTPUT   DEBUG CONSOLE   TERMINAL          2: cppdbg:

-9600
-9584
-9600
```

从上面的例子可以看出&nums[0] + 1 是越过了nums[0]这一行，即&nums[0]代表的是数组对象nums[0]的整个地址,下面给出a[0] 与 &a[0]的区别:

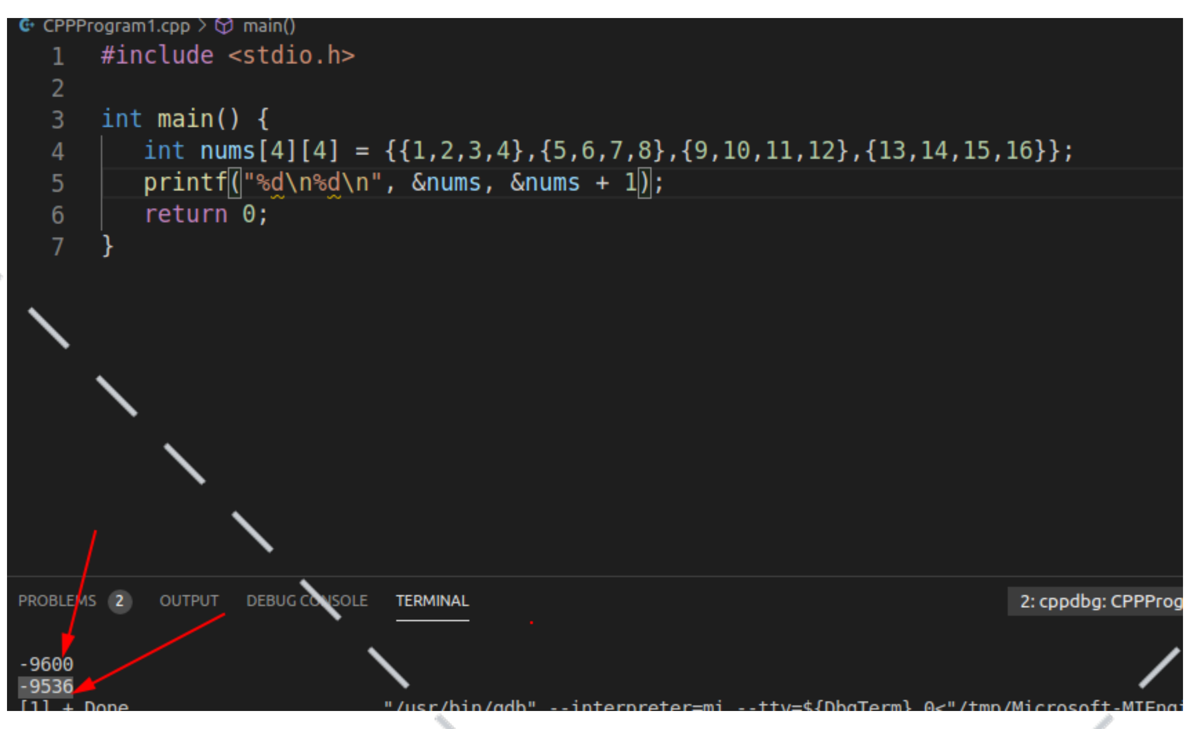


@hacck s: `a[i]` is an array object of type `int[2]`, which is row `i` of `a`. In most contexts the expression `a[i]` decays to a pointer to the first element of that array, i.e., a pointer to the `int` object `a[i][0]`. But in `&a[i]`, since `a[i]` is the operand of unary `&`, it doesn't decay, and `&a[i]` is the address of the array object `a[i]`, and is of type `int(*)[2]`. No, `&a[i]` and `a[i]` are not identical; the former is the address of row `i` of `a`, and `a[i]` is either that row or the address of *the first element of* that row, depending on the context. – Keith Thompson Aug 21 '13 at 19:13

&a

a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2][0]	a[2][1]	a[2][2]	a[2][3]
a[3][0]	a[3][1]	a[3][2]	a[3][3]

&a代表的是数组对象a的整个地址.



```
CPPProgram1.cpp > main()
#include <stdio.h>

int main() {
    int nums[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
    printf("%d\n%d\n", &nums, &nums + 1);
    return 0;
}
```

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL          2: cppdbg: CPPProg

-9600
-9536
[1] + Done                    "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngi
```