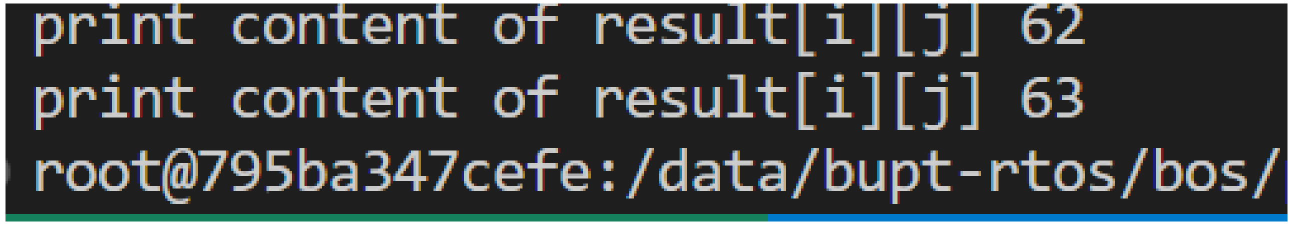
Easylab3—magic print

Magic Print —with print

```cpp
#include <iostream>
using namespace std;

#define array_number 64

int matrix[array_number][array_number];

int **double_array(size_t n) {
    int **result = new int*[8];

        for (int i = 0; i < n; ++i) {
            result[i] = matrix[i];
            for (int j = 0; j < n; ++j){
                result[i][j] = j;
            }
    }

    return result;
}
```

```cpp
int main() {
    cout<<"A magic print! If you comment this,
the program will break."<<endl;
    int **result = double_array(array_number);

    for (int i = 0; i < array_number; ++i) {
        cout<<"print address of result[i]
"<<&result[i][0]<<endl;
        for (int j = 0; j < array_number; j++) {
            result[i][j] = j;
            cout<<"print content of result[i][j]
"<<result[i][j]<<endl;
        }
    }
    free(result);
}
```

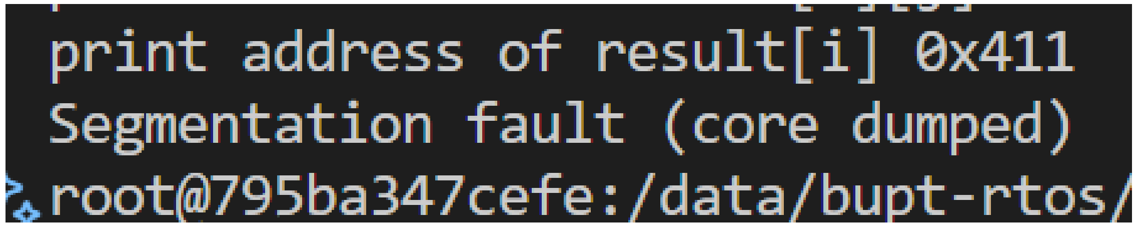


```
print content of result[i][j] 62
print content of result[i][j] 63
root@795ba347cefe:/data/bupt-rtos/bos/
```

Magic Print—without print

```cpp
#include <iostream>
using namespace std;

#define array_number 64

int matrix[array_number][array_number];

int **double_array(size_t n) {
    int **result = new int*[8];

        for (int i = 0; i < n; ++i) {
            result[i] = matrix[i];
            for (int j = 0; j < n; ++j){
                result[i][j] = j;
            }
    }

    return result;
}
```

```cpp
int main() {
    // cout<<"A magic print! If you comment this,
the program will break."<<endl;
    int **result = double_array(array_number);

    for (int i = 0; i < array_number; ++i) {
        cout<<"print address of result[i]
"<<&result[i][0]<<endl;
        for (int j = 0; j < array_number; j++) {
            result[i][j] = j;
            cout<<"print content of result[i][j]
"<<result[i][j]<<endl;
        }
    }
    free(result);
}
```

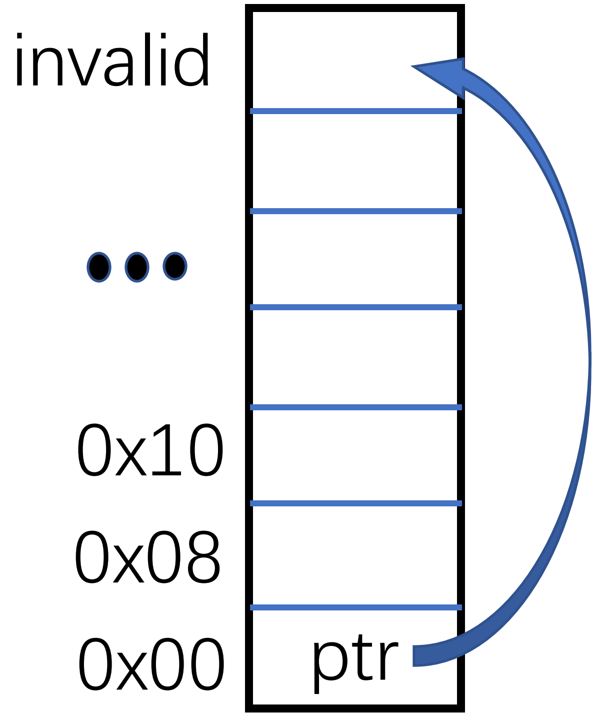


```
print address of result[i] 0x411
Segmentation fault (core dumped)
root@795ba347cefe:/data/bupt-rtos/
```

How does `Print` affiliated with memory model

```
cout<<"A magic print! If you comment this, the program will break."<<endl;
```

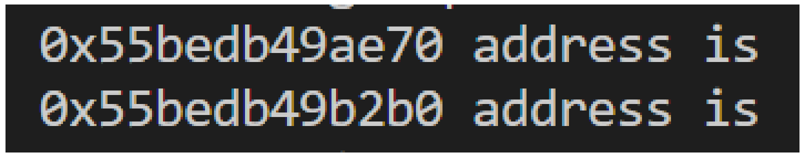
Print may access mem



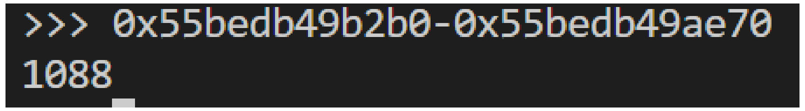
```
cout<<"Ptr value
is "<<*ptr<<endl;
```

Print mallocs buffer

```
void *a = malloc(32);
cout<<"malloc"<<endl;
void *b = malloc(64);
cout<< a<<" address is "<<endl;
cout<< b<<" address is "<<endl;
```

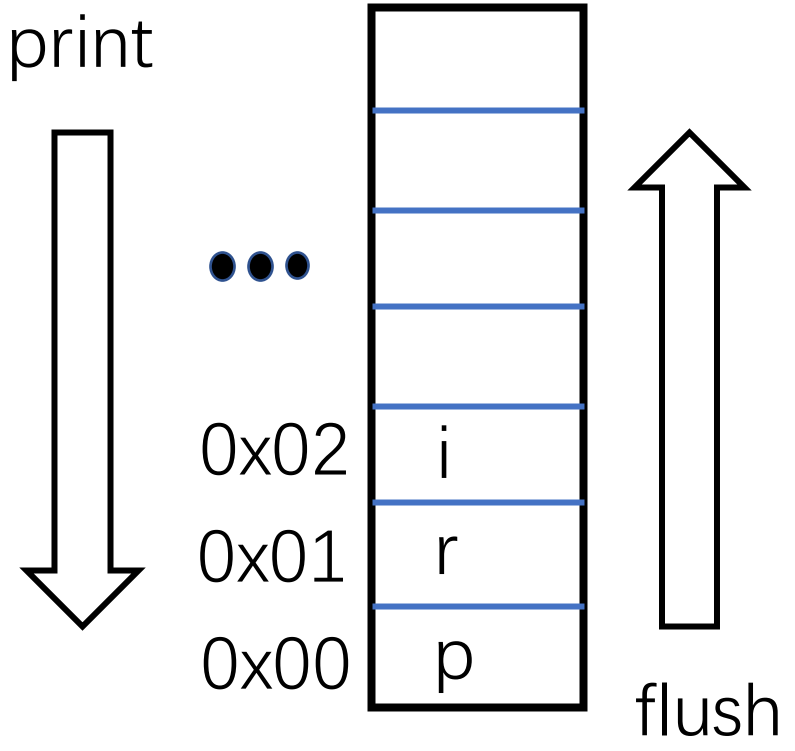


```
0x55bedb49ae70 address is
0x55bedb49b2b0 address is
```



```
>>> 0x55bedb49b2b0-0x55bedb49ae70
1088
```

Print flushes the buffer

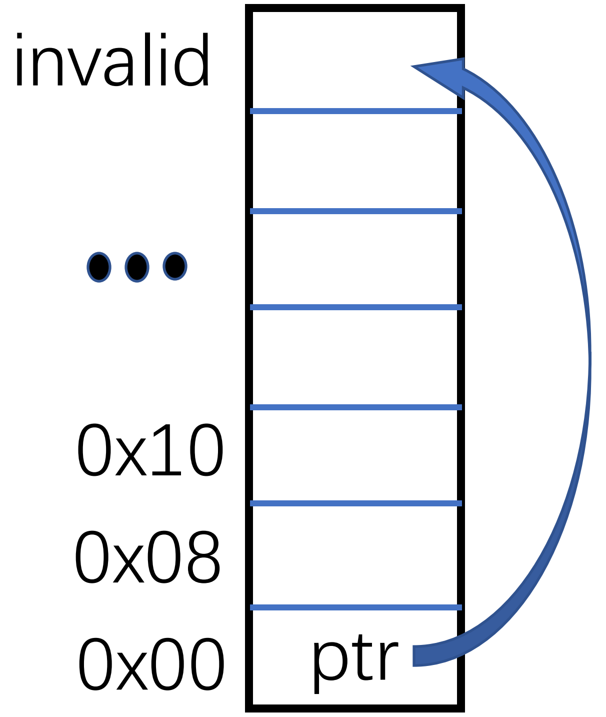


```
cout<<"print buffer" <<endl;
```

How does `Print` interact with memory model

```
cout<<"A magic print! If you comment this, the program will break."<<endl;
```

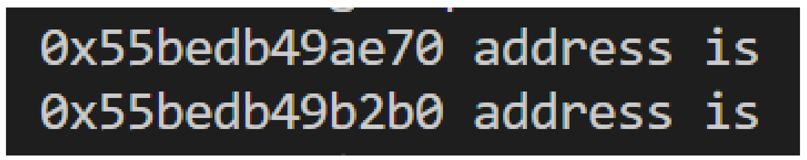
Print may access mem



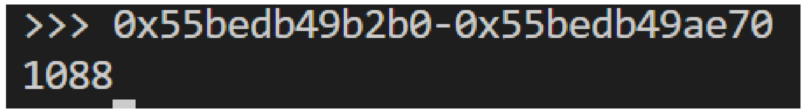
```
cout<<"Ptr value
is "<<*ptr<<endl;
```

Print mallocs buffer

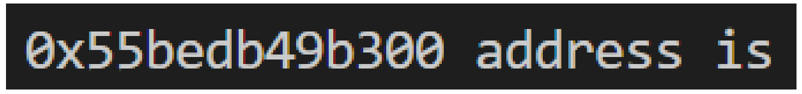
```
void *a = malloc(32);
cout<<"malloc"<<endl;
void *b = malloc(64);
cout<<"Another malloc"<<endl;
void *f = malloc(96);
cout<< a<<" address is "<<endl;
cout<< b<<" address is "<<endl;
cout<< f<<" address is "<<endl;
```



```
0x55bedb49ae70 address is
0x55bedb49b2b0 address is
```

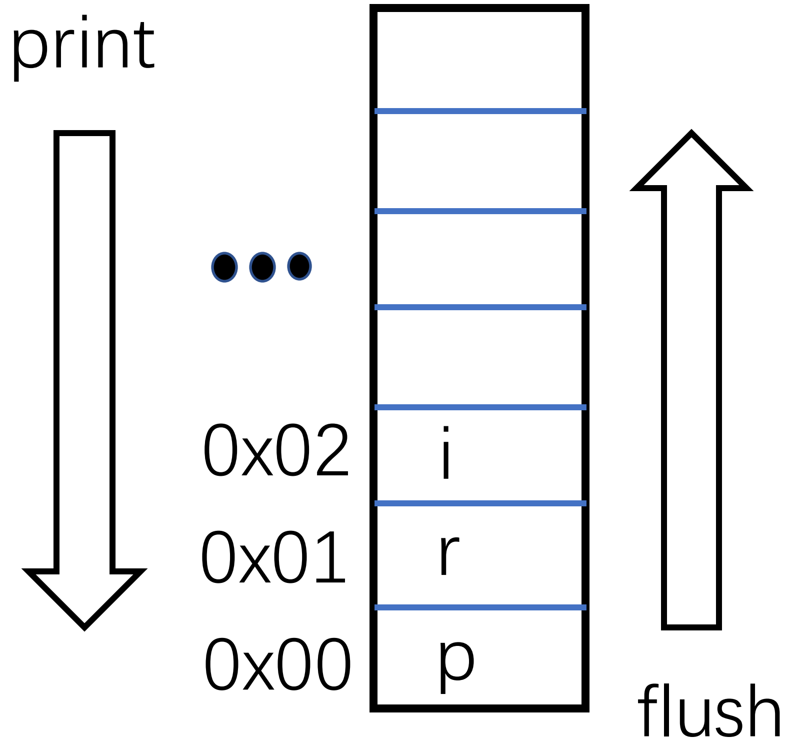


```
>>> 0x55bedb49b2b0-0x55bedb49ae70
1088
```



```
0x55bedb49b300 address is
```

Print flushes the buffer



```
cout<<"print buffer" <<endl;
```

Program analysis

```
#include <iostream>
using namespace std;

#define array_number 64

int matrix[array_number][array_number];

int **double_array(size_t n) {
    int **result = new int*[8];  2

        for (int i = 0; i < n; ++i) {
            result[i] = matrix[i];
            for (int j = 0; j < n; ++j){
                result[i][j] = j;
            }
    }

    return result;
}
```

```
                                   1
int main() {
    cout<<"A magic print! If you comment this,
the program will break."<<endl;
    int **result = double_array(array_number);

    for (int i = 0; i < array_number; ++i) {
        cout<<"print address of result[i]  ?
"<<&result[i][0]<<endl;
        for (int j = 0; j < array_number; j++) {
            result[i][j] = j;
            cout<<"print content of result[i][j]
"<<result[i][j]<<endl;
        }                                   ?
    }
    free(result);
}
```

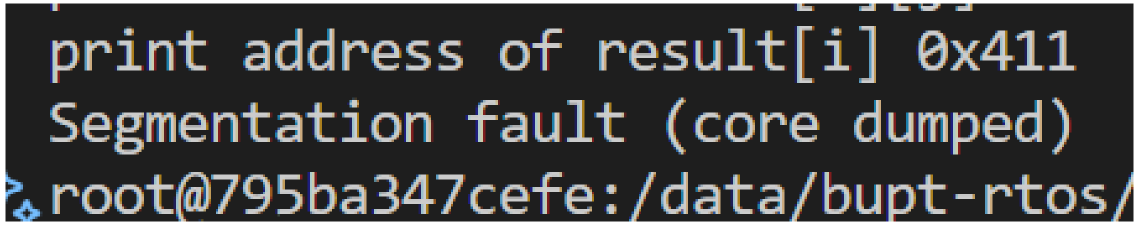
Program analysis

```
#include <iostream>
using namespace std;

#define array_number 64

int matrix[array_number][array_number];

int **double_array(size_t n) {
    int **result = new int*[8];   1

        for (int i = 0; i < n; ++i) {
            result[i] = matrix[i];
            for (int j = 0; j < n; ++j){
                result[i][j] = j;
            }
    }

    return result;
}
```

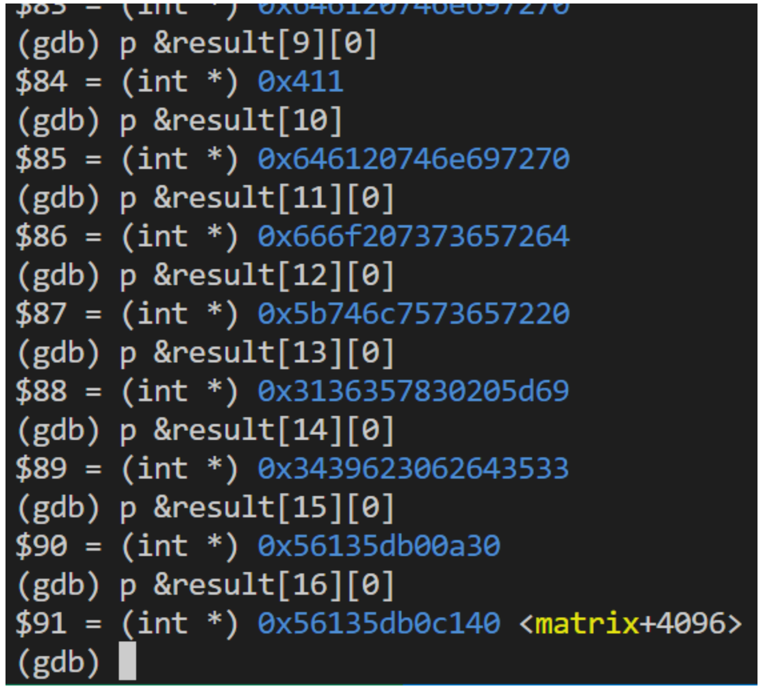
no

```
int main() {
    // cout<<"A magic print! If you comment this,
the program will break."<<endl;
    int **result = double_array(array_number);

    for (int i = 0; i < array_number; ++i) {
        cout<<"print address of result[i]   2
"<<&result[i][0]<<endl;
        for (int j = 0; j < array_number; j++) {
            result[i][j] = j;
            cout<<"print content of result[i][j]
"<<result[i][j]<<endl;
        }                                         ?
    }
    free(result);
}
```

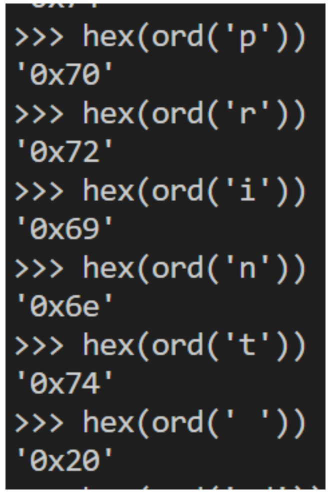
What's the problem?



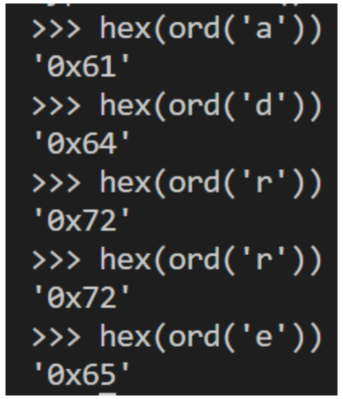
```
print address of result[i] 0x411
Segmentation fault (core dumped)
root@795ba347cefe:/data/bupt-rtos/
```



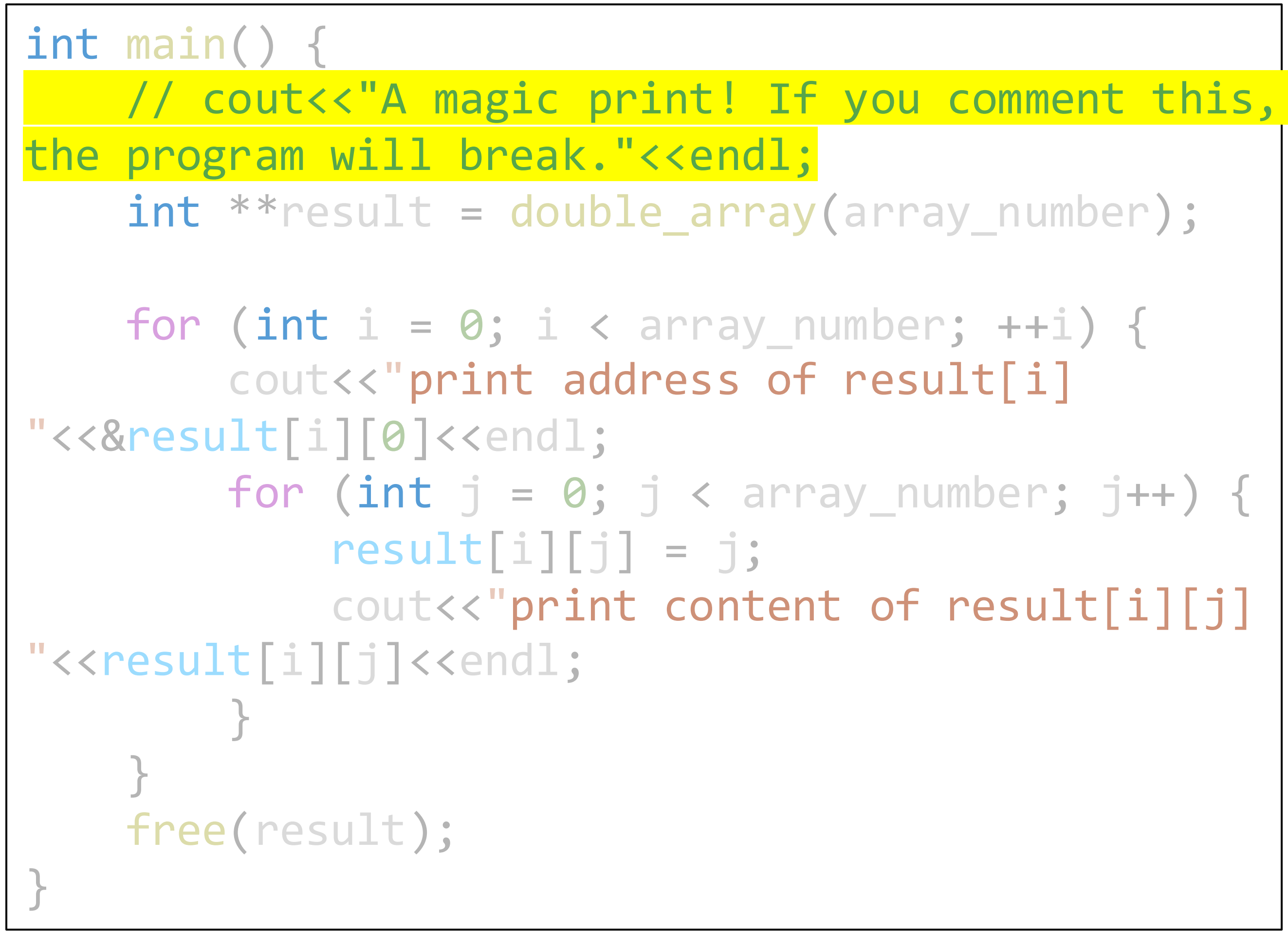
```
(gdb) p &result[9][0]
$84 = (int *) 0x411
(gdb) p &result[10]
$85 = (int *) 0x646120746e697270
(gdb) p &result[11][0]
$86 = (int *) 0x666f207373657264
(gdb) p &result[12][0]
$87 = (int *) 0x5b746c7573657220
(gdb) p &result[13][0]
$88 = (int *) 0x3136357830205d69
(gdb) p &result[14][0]
$89 = (int *) 0x3439623062643533
(gdb) p &result[15][0]
$90 = (int *) 0x56135db00a30
(gdb) p &result[16][0]
$91 = (int *) 0x56135db0c140 <matrix+4096>
(gdb)
```



```
>>> hex(ord('p'))
'0x70'
>>> hex(ord('r'))
'0x72'
>>> hex(ord('i'))
'0x69'
>>> hex(ord('n'))
'0x6e'
>>> hex(ord('t'))
'0x74'
>>> hex(ord(' '))
'0x20'
```



```
>>> hex(ord('a'))
'0x61'
>>> hex(ord('d'))
'0x64'
>>> hex(ord('r'))
'0x72'
>>> hex(ord('r'))
'0x72'
>>> hex(ord('e'))
'0x65'
```



```
int main() {
    // cout<<"A magic print! If you comment this,
the program will break."<<endl;
    int **result = double_array(array_number);

    for (int i = 0; i < array_number; ++i) {
        cout<<"print address of result[i]
"<<&result[i][0]<<endl;
        for (int j = 0; j < array_number; j++) {
            result[i][j] = j;
            cout<<"print content of result[i][j]
"<<result[i][j]<<endl;
        }
    }
    free(result);
}
```

Think time

Can we replace this cout with malloc?

```
int main() {
    // cout<<"A magic print! If you comment this, the program will break."<<endl;
    int **result = double_array(array_number);

    for (int i = 0; i < array_number; ++i) {
        cout<<"print address of result[i] "<<&result[i][0]<<endl;
        for (int j = 0; j < array_number; j++) {
            result[i][j] = j;
            cout<<"print content of result[i][j] "<<result[i][j]<<endl;
        }
    }
    free(result);
}
```

Think time

Can we replace this cout with malloc?

No!
The next cout will malloc a new buffer rewriting the prior buffer.

```
int main() {
    // cout<<"A magic print! If you comment this,
the program will break."<<endl;
    int **result = double_array(array_number);

    for (int i = 0; i < array_number; ++i) {
        cout<<"print address of result[i]
"<<&result[i][0]<<endl;
        for (int j = 0; j < array_number; j++) {
            result[i][j] = j;
            cout<<"print content of result[i][j]
"<<result[i][j]<<endl;
        }
    }
    free(result);
}
```