easy_lab3

错误分析

注释掉 cout 后，报错：

```
malloc(): corrupted top size
Aborted
```

可知，内存访问越界，在 `double_array` 函数中，为 `result` 变量分配了空间，用于存放 8 个 `int*` 指针：

```
int **result = new int*[8];
```

然而，在接下来的循环中，尝试访问并修改 `result` 数组的前 `n` 个元素，其中 `n` 定义为 `array_number`（64）。这意味着尝试访问 `result[8]` 到 `result[63]`，这些位置是未被正确分配的：

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

修复方法

确保 `result` 数组的大小与你实际访问的大小一致。代码修改如下：

```
int **result = new int*[array_number];
```

Cout 为什么会隐藏这个问题

下面这段代码展示了 cout 对于内存布局的影响

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

int main() {
  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;
  return 0;
}
```

结果：

```
malloc
Another malloc
0x55dd93186eb0address is
0x55dd931872f0address is
0x55dd93187340address is
```

```
>>> 0x55dd931872f0-0x55dd93186eb0
1088

>>> 0x55dd93187340-0x55dd931872f0
80
```

可以看出 cout 在首次运行的时候会去申请一块内存空间用于缓冲，这部分的缓冲区是在堆上的，也就是和我们动态内存分配（new/malloc）是在同一个内存区域中。对于原程序，如果没注释掉第一行的 `cout` ，则程序执行时会首先在堆上 `malloc` 一个缓冲区，此时，尝试访问 `result[8]` 到 `result[63]` 时会将缓冲区中的地址代替未分配的地址。