10. Найти произведение чисел на побочной диагонали и на основной диагонали.

Код программы на C++:

```cpp
#include "pch.h"
#include <iostream>
#include <ctime>

using namespace std;

int main()
{
    srand(time(NULL));
    int matrix[5][5], mult1 = 1, mult2 = 1, count = 0;

    cout << "Matrix: " << endl;

    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            matrix[i][j] = rand()%9 + 1;
            cout << matrix[i][j] << " ";
        }

        cout << endl;
    }

    for (int i = 0; i < 5; i++) {
        count = i;
        mult1 *= matrix[count][count];
        mult2 *= matrix[count][4 - count];
    }

    cout << "\nMultiple of main diagonal: " << mult1 << endl;
    cout << "Multiple of secondary diagonal: " << mult2 << endl;
}
```

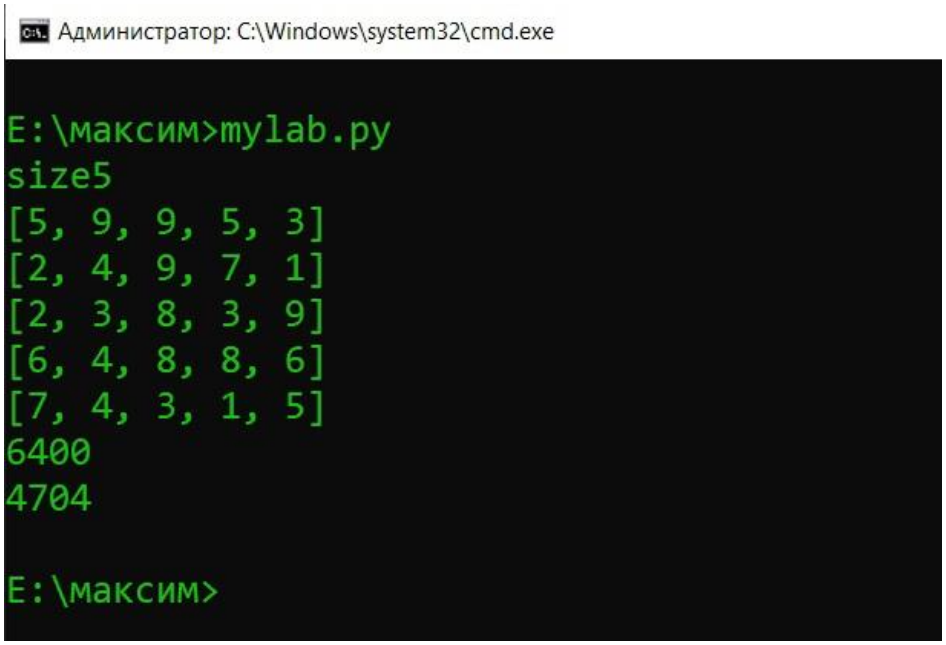
Результат работы программы на C++:

```
Matrix:
8 6 7 5 7
1 1 4 9 5
4 8 1 5 7
8 4 3 4 7
5 9 8 5 3

Multiple of main diagonal: 96
Multiple of secondary diagonal: 1260
```

Код программы на Python:

```python
import random

matrix = []
mult1 = 1
mult2 = 1

size = int(input("size"))

for i in range(size):
    li = []
    for j in range(size):
        li.append(random.randint(1,9))
    matrix.append(li)

for i in matrix:
    print(i)

for j in range(size):
        mult1 *= matrix[j][j]
        mult2 *= matrix[j][size - 1 -j]

print(mult1)
print(mult2)
```

Результат работы на Python:



Вывод: выделение памяти под матрицу на C++ можно осуществлять двумя способами: динамически и статически. На примере лабораторной работы я выделял память статически, что предотвращает возможную утечку памяти, но уменьшает гибкость программы и возможность увеличения размерности двумерного массива. На примере языка Python я выделял память динамически. Это позволило задавать размер матрицы во время выполнения кода, а не на этапе компиляции. К тому же, python сам очищает память в отличие от C++ (команда delete).