Linked List

Implement linked list class, their basic function and time complexity

(addAtHead, addAtTail, removeAtHead, removeAtTail)

- Single linked list
  https://replit.com/@melvinsyriac/single#main.cpp
- Double linked list
  https://replit.com/@melvinsyriac/double#main.cpp
- Circular
  https://replit.com/@melvinsyriac/circular#main.cpp

Sorting

- Bubble Sort

  Best case: O(n)

  Worst case: O(n^2)

- Selection Sort

  Best case: O(n^2)

  Worst case: O(n^2)

- Insertion Sort

  Best case: O(n)

  Worst case: O(n^2)

- Implementation: https://replit.com/@melvinsyriac/sorting#main.cpp

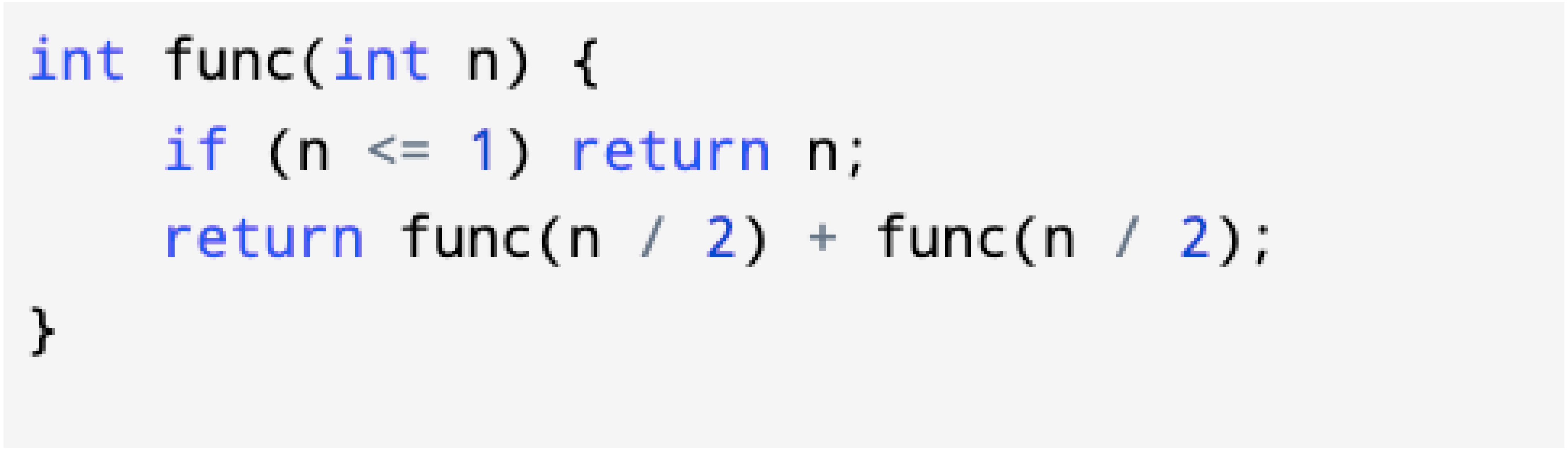
Sorting

- Bubble Sort execution time is longer than Selection Sort because Bubble Sort uses more swap than Selection Sort.
- Insertion Sort have faster execution time than Bubble Sort and Selection Sort because Insertion Sort doesn’t swap elements around (Insertion Sort shifting index).
- In general, Insertion Sort is more efficient than Selection Sort, and Selection Sort is more efficient than Bubble Sort.
- Best case scenario for these 3 sorting method happen when the list is already sorted.
- Worst case scenario for these 3 sorting method happen when the list is sorted but in reverse order.

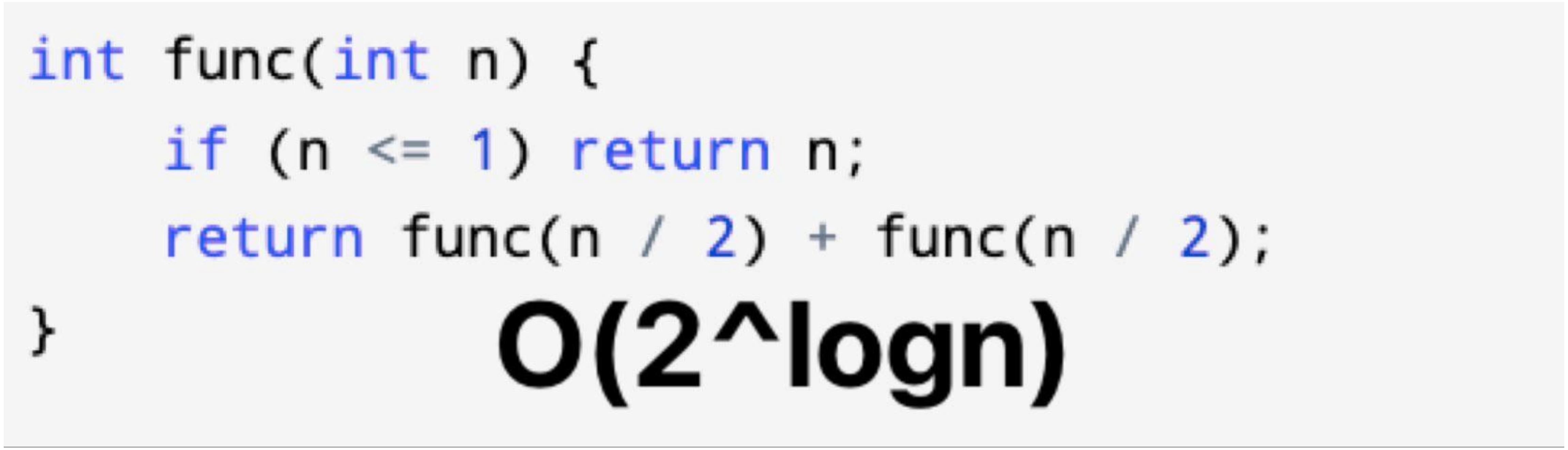
Recursion

https://replit.com/@melvinsyriac/recursion#main.cpp

Big O

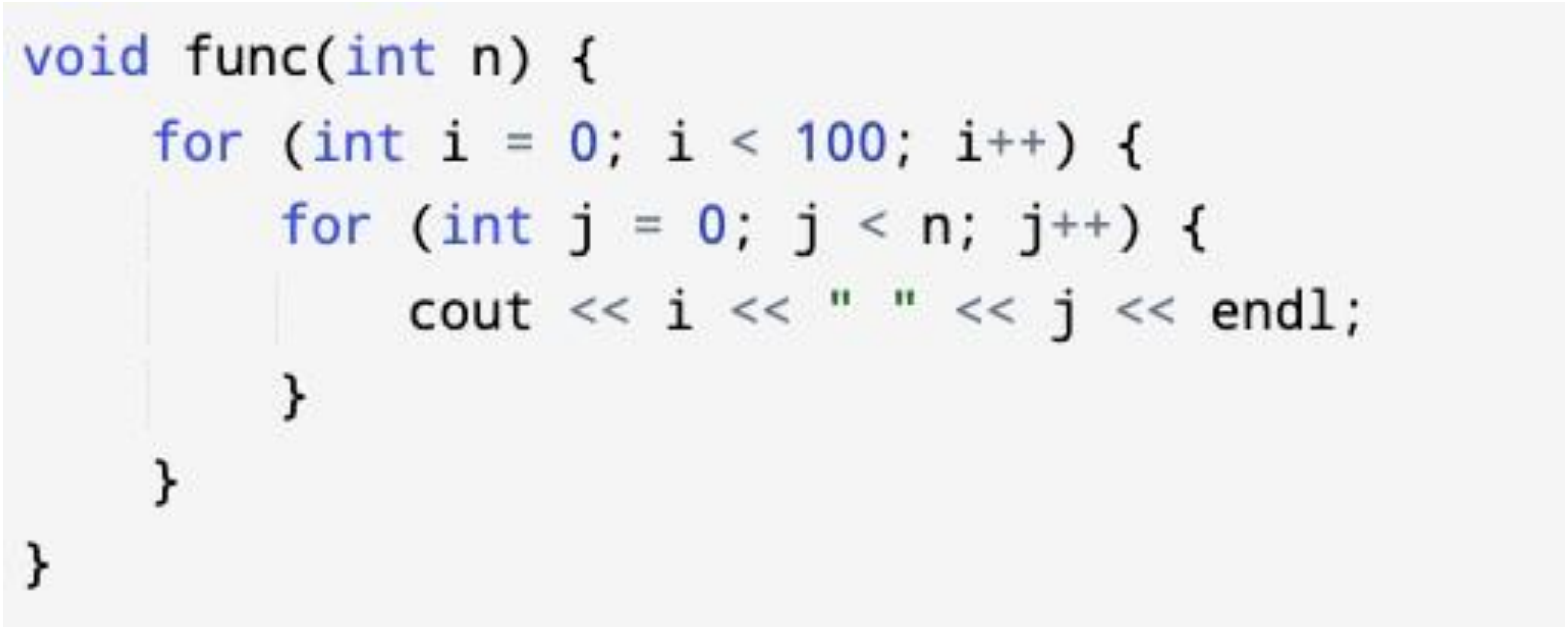


```
int func(int n) {
    if (n <= 1) return n;
    return func(n / 2) + func(n / 2);
}
```

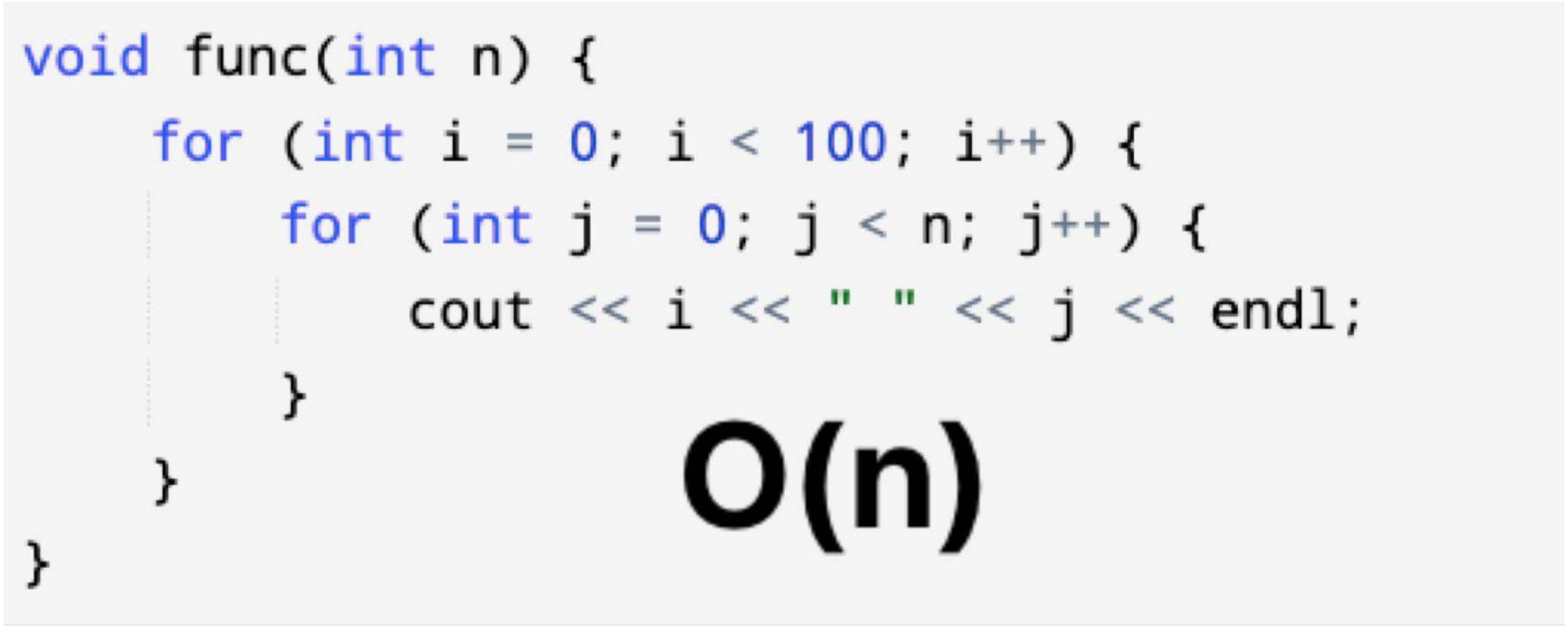


```
int func(int n) {
    if (n <= 1) return n;
    return func(n / 2) + func(n / 2);
}
```

O(2^logn)

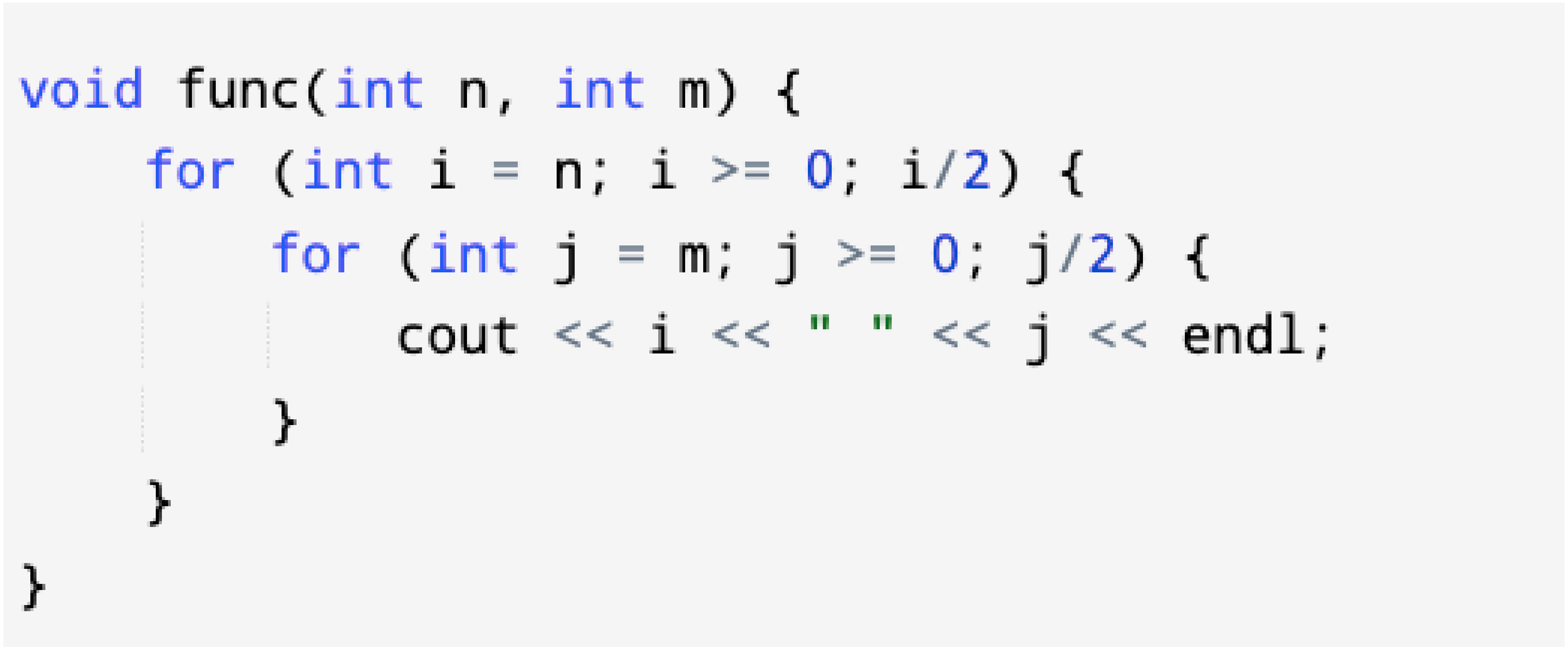


```cpp
void func(int n) {
    for (int i = 0; i < 100; i++) {
        for (int j = 0; j < n; j++) {
            cout << i << " " << j << endl;
        }
    }
}
```

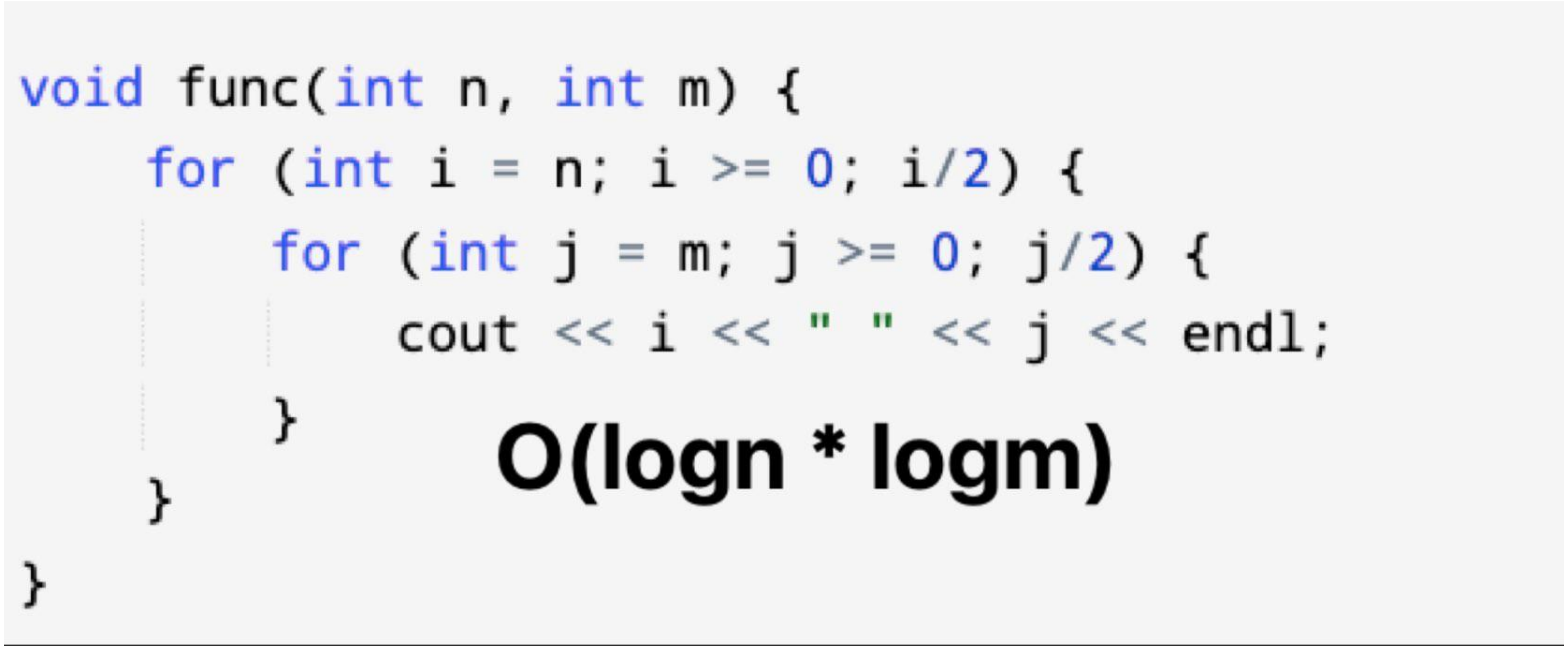


```
void func(int n) {
    for (int i = 0; i < 100; i++) {
        for (int j = 0; j < n; j++) {
            cout << i << " " << j << endl;
        }
    }
}
```

O(n)

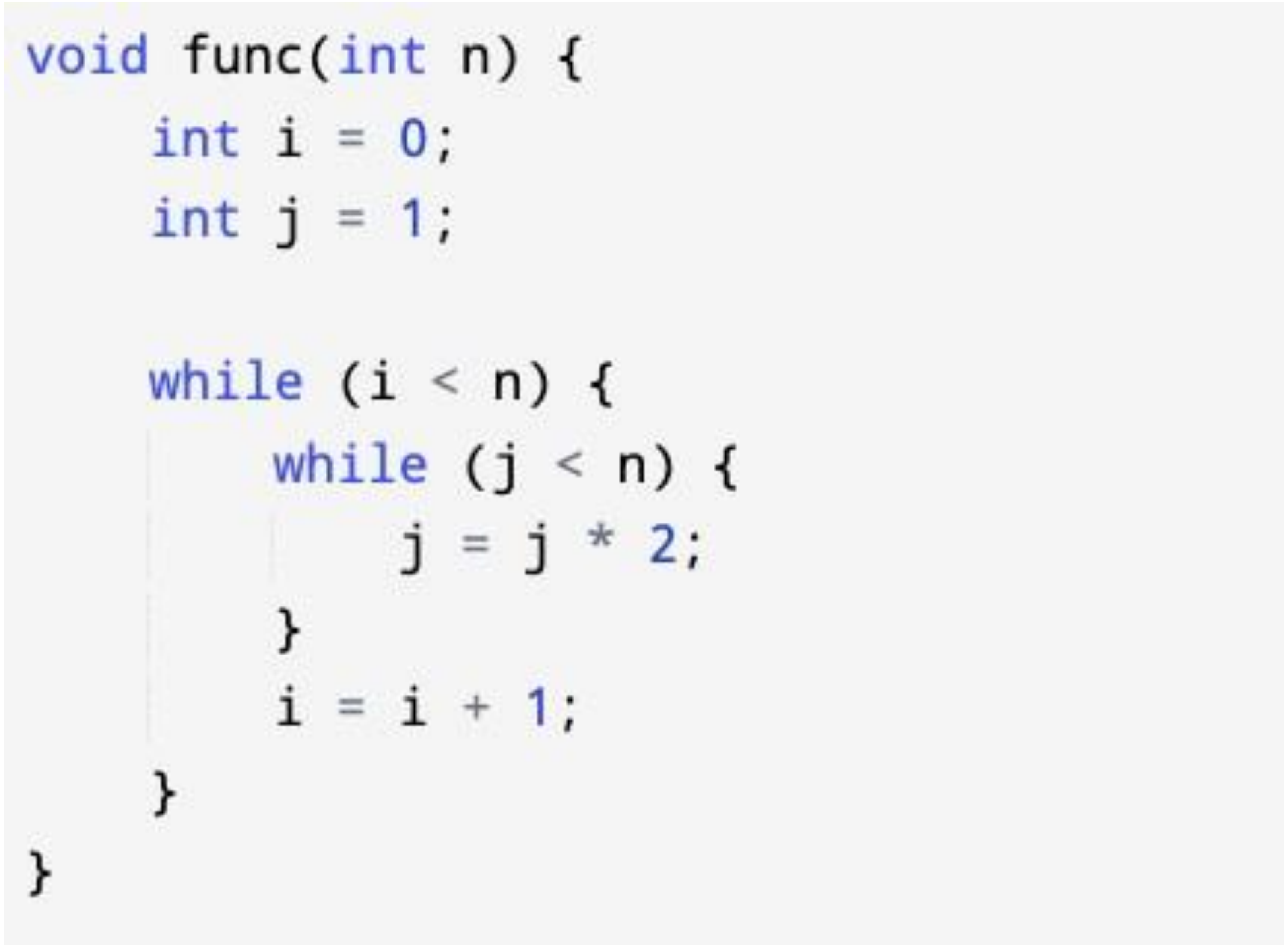


```cpp
void func(int n, int m) {
    for (int i = n; i >= 0; i/2) {
        for (int j = m; j >= 0; j/2) {
            cout << i << " " << j << endl;
        }
    }
}
```



```
void func(int n, int m) {
    for (int i = n; i >= 0; i/2) {
        for (int j = m; j >= 0; j/2) {
            cout << i << " " << j << endl;
        }
    }
}
```

O(logn * logm)



```
void func(int n) {
    int i = 0;
    int j = 1;

    while (i < n) {
        while (j < n) {
            j = j * 2;
        }
        i = i + 1;
    }
}
```

```
void func(int n) {
    int i = 0;
    int j = 1;

    while (i < n) {
        while (j < n) {
            j = j * 2;
        }
        i = i + 1;
    }
}
```

O(nlogn)