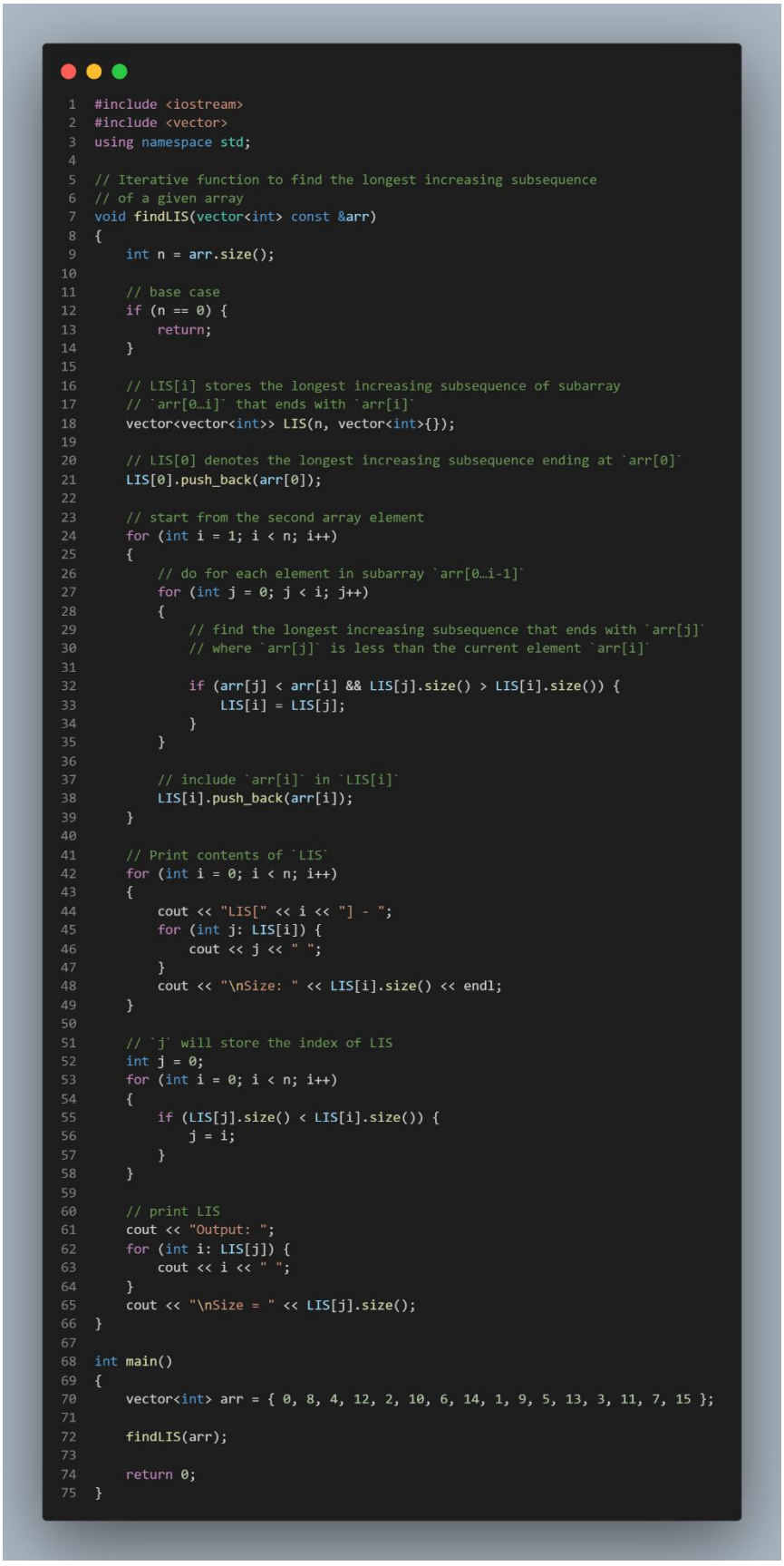
C version of the algorithm (found on internet):



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

// Iterative function to find the longest increasing subsequence
// of a given array
void findLIS(vector<int> const &arr)
{
    int n = arr.size();

    // base case
    if (n == 0) {
        return;
    }

    // LIS[i] stores the longest increasing subsequence of subarray
    // `arr[0…i]` that ends with `arr[i]`
    vector<vector<int>> LIS(n, vector<int>{});

    // LIS[0] denotes the longest increasing subsequence ending at `arr[0]`
    LIS[0].push_back(arr[0]);

    // start from the second array element
    for (int i = 1; i < n; i++)
    {
        // do for each element in subarray `arr[0…i-1]`
        for (int j = 0; j < i; j++)
        {
            // find the longest increasing subsequence that ends with `arr[j]`
            // where `arr[j]` is less than the current element `arr[i]`

            if (arr[j] < arr[i] && LIS[j].size() > LIS[i].size()) {
                LIS[i] = LIS[j];
            }
        }

        // include `arr[i]` in `LIS[i]`
        LIS[i].push_back(arr[i]);
    }

    // Print contents of `LIS`
    for (int i = 0; i < n; i++)
    {
        cout << "LIS[" << i << "] - ";
        for (int j: LIS[i]) {
            cout << j << " ";
        }
        cout << "\nSize: " << LIS[i].size() << endl;
    }

    // `j` will store the index of LIS
    int j = 0;
    for (int i = 0; i < n; i++)
    {
        if (LIS[j].size() < LIS[i].size()) {
            j = i;
        }
    }

    // print LIS
    cout << "Output: ";
    for (int i: LIS[j]) {
        cout << i << " ";
    }
    cout << "\nSize = " << LIS[j].size();
}

int main()
{
    vector<int> arr = { 0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15 };

    findLIS(arr);

    return 0;
}
```

Time complexity = O(n^2), **Space complexity =** O(n^2)

- All explanations are in the code comments.
- Result is wrong and produces infinite loop.
- I can not test the program in this case.
- Program uses procedures (jal&jr)