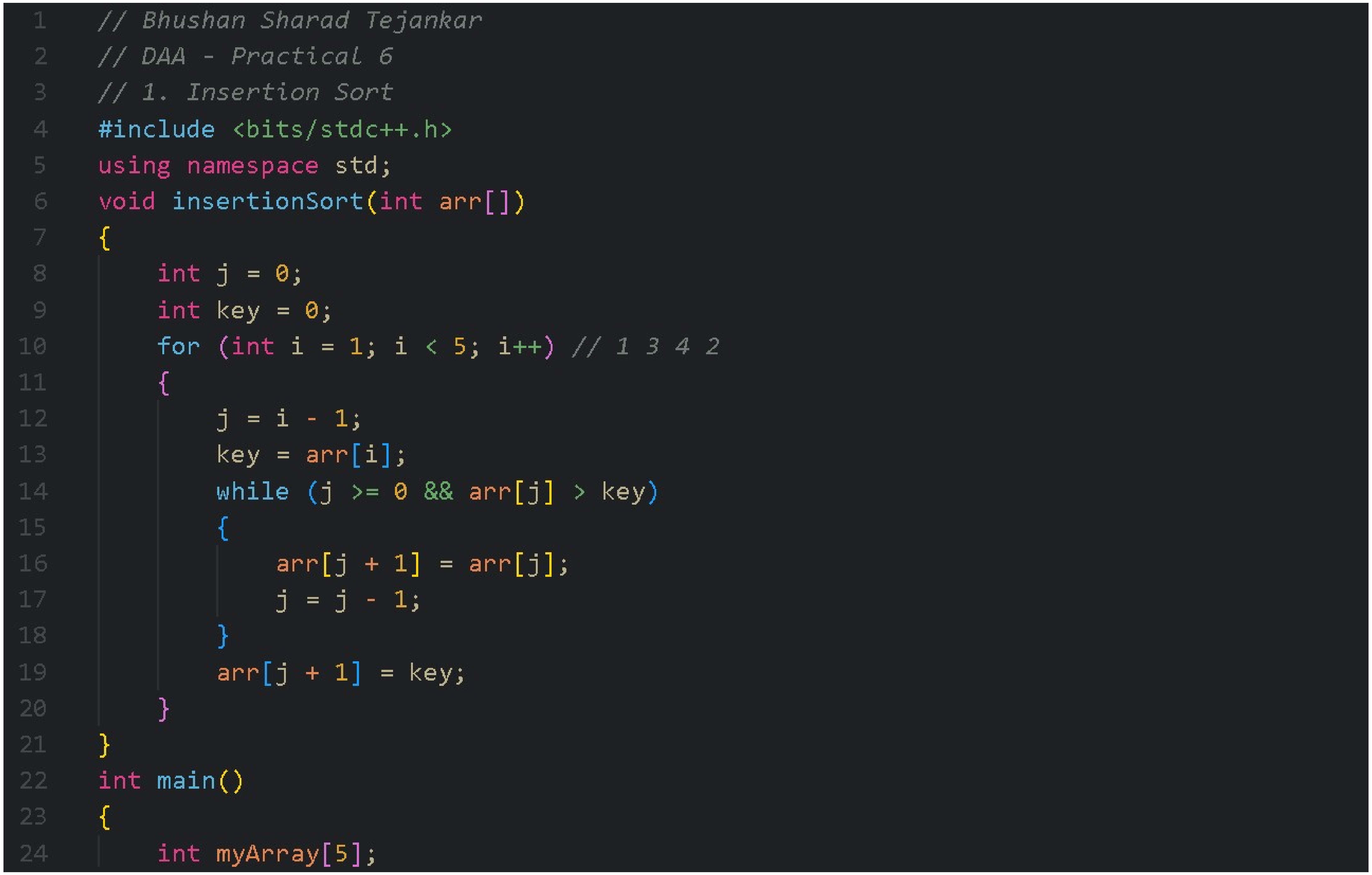
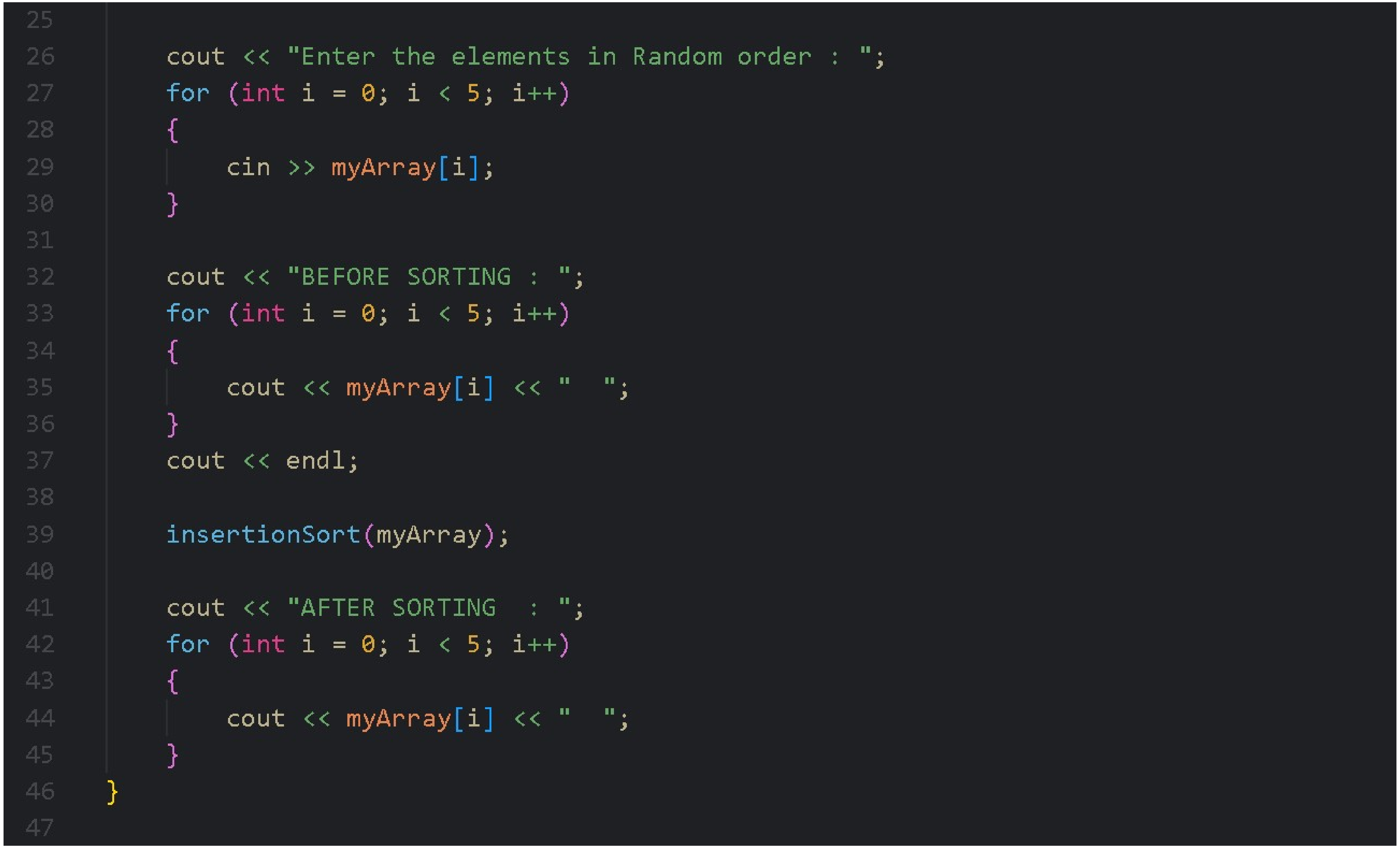
Name : Bhushan Sharad Tejankar

Roll no. : I30

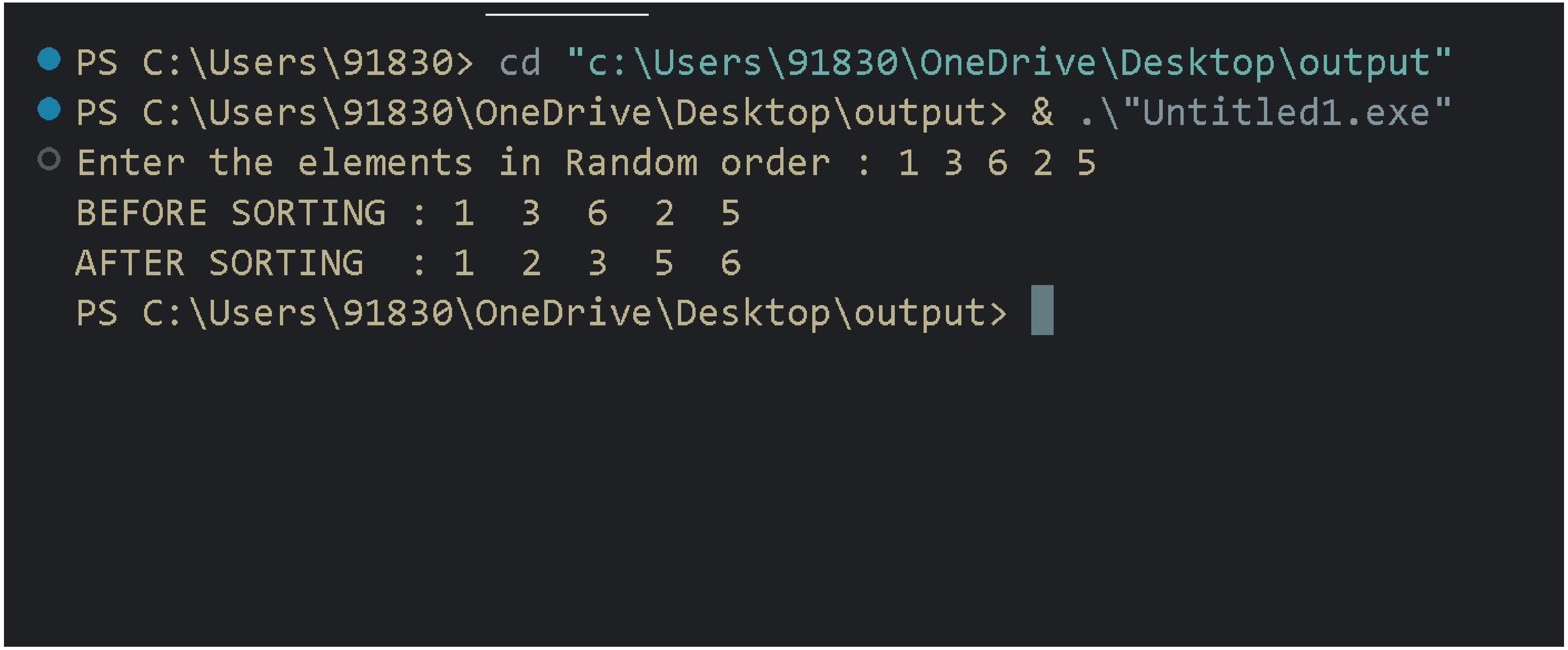
reg_no. : 2020BIT030



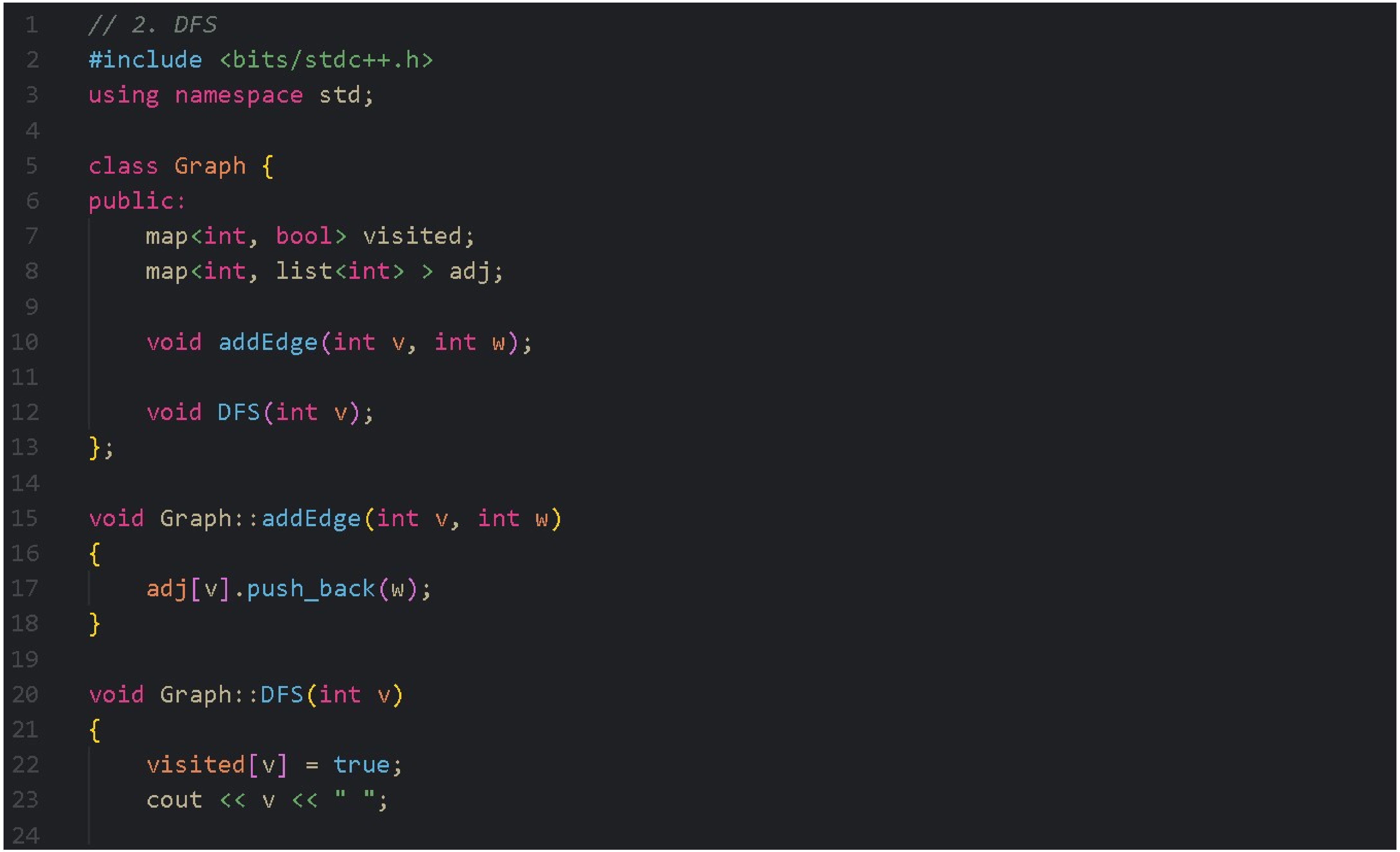
```cpp
// Bhushan Sharad Tejankar
// DAA - Practical 6
// 1. Insertion Sort
#include <bits/stdc++.h>
using namespace std;
void insertionSort(int arr[])
{
    int j = 0;
    int key = 0;
    for (int i = 1; i < 5; i++) // 1 3 4 2
    {
        j = i - 1;
        key = arr[i];
        while (j >= 0 && arr[j] > key)
        {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
int main()
{
    int myArray[5];
```



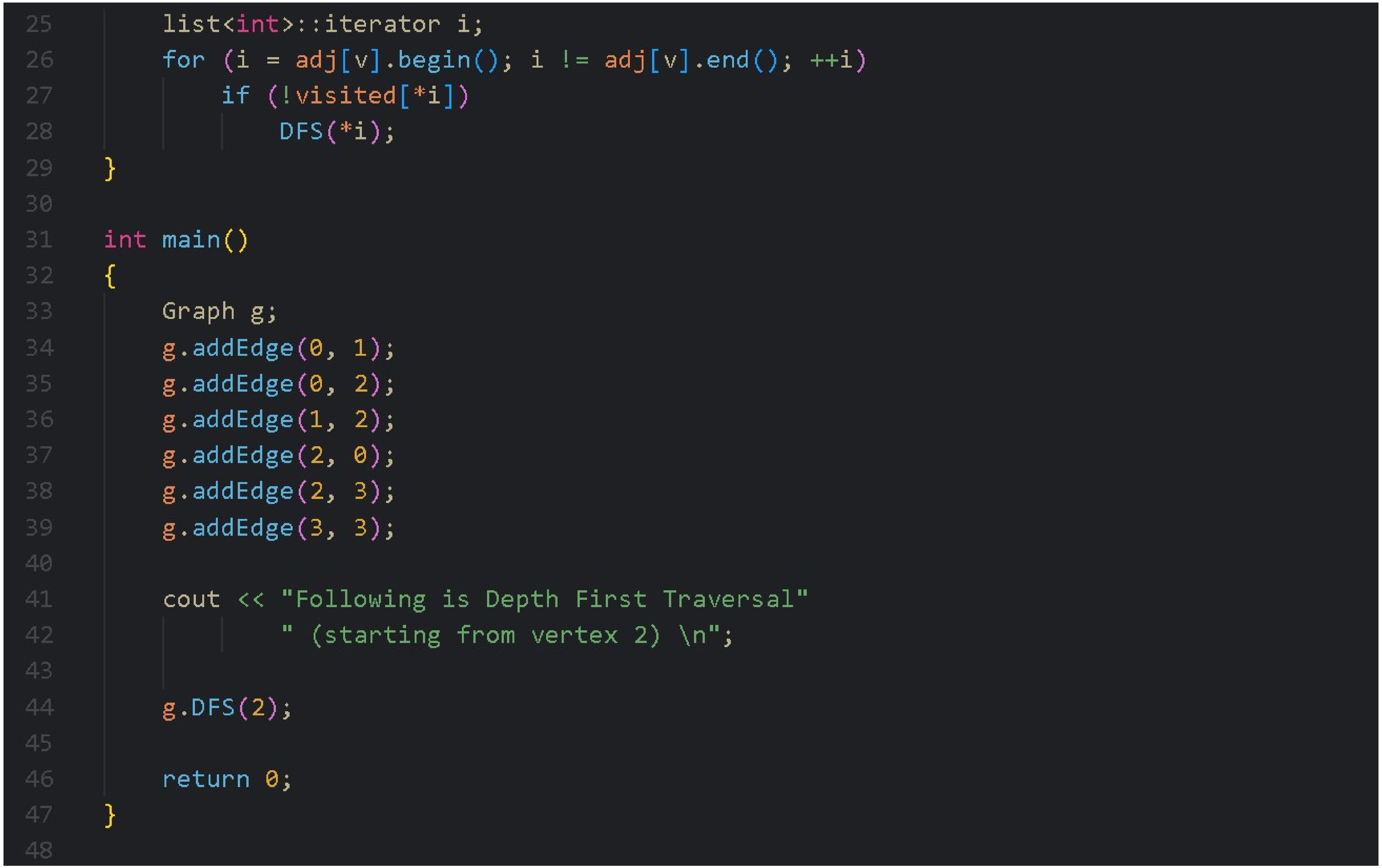
```cpp

    cout << "Enter the elements in Random order : ";
    for (int i = 0; i < 5; i++)
    {
        cin >> myArray[i];
    }

    cout << "BEFORE SORTING : ";
    for (int i = 0; i < 5; i++)
    {
        cout << myArray[i] << "  ";
    }
    cout << endl;

    insertionSort(myArray);

    cout << "AFTER SORTING  : ";
    for (int i = 0; i < 5; i++)
    {
        cout << myArray[i] << "  ";
    }
}

```

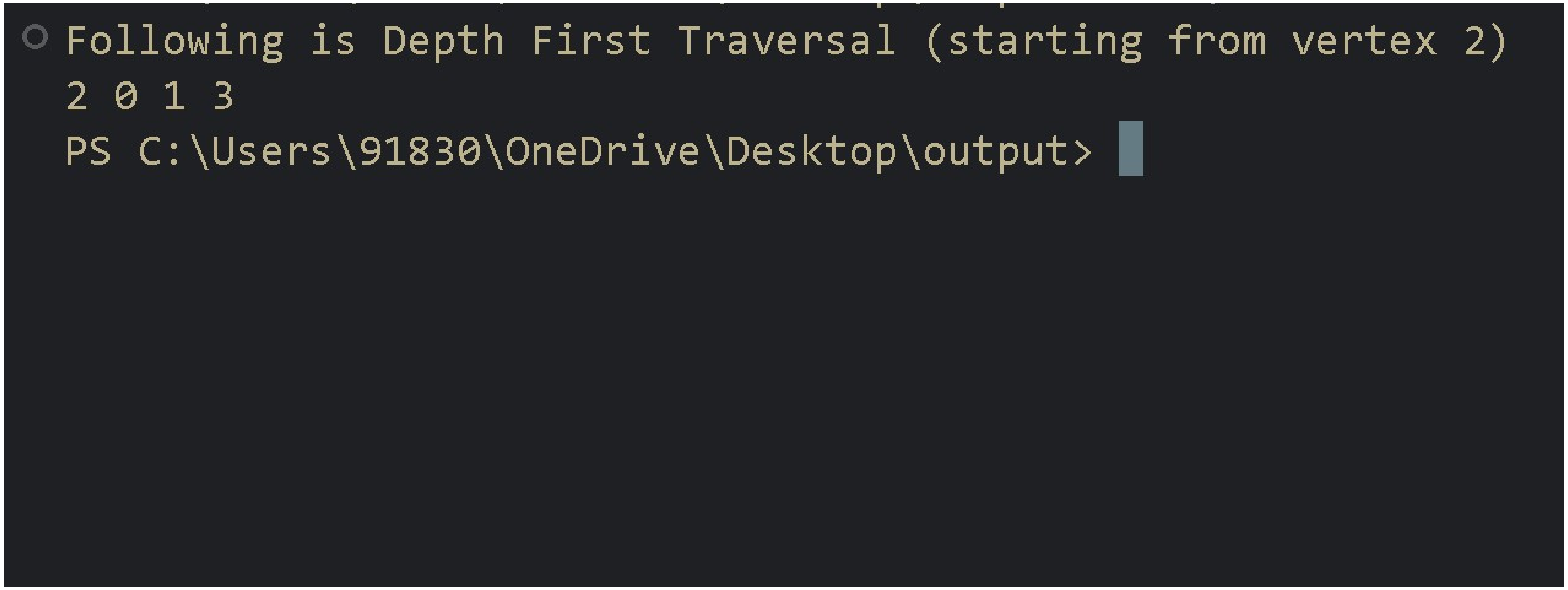


```
● PS C:\Users\91830> cd "c:\Users\91830\OneDrive\Desktop\output"
● PS C:\Users\91830\OneDrive\Desktop\output> & .\"Untitled1.exe"
○ Enter the elements in Random order : 1 3 6 2 5
  BEFORE SORTING : 1   3   6   2   5
  AFTER SORTING  : 1   2   3   5   6
  PS C:\Users\91830\OneDrive\Desktop\output>
```

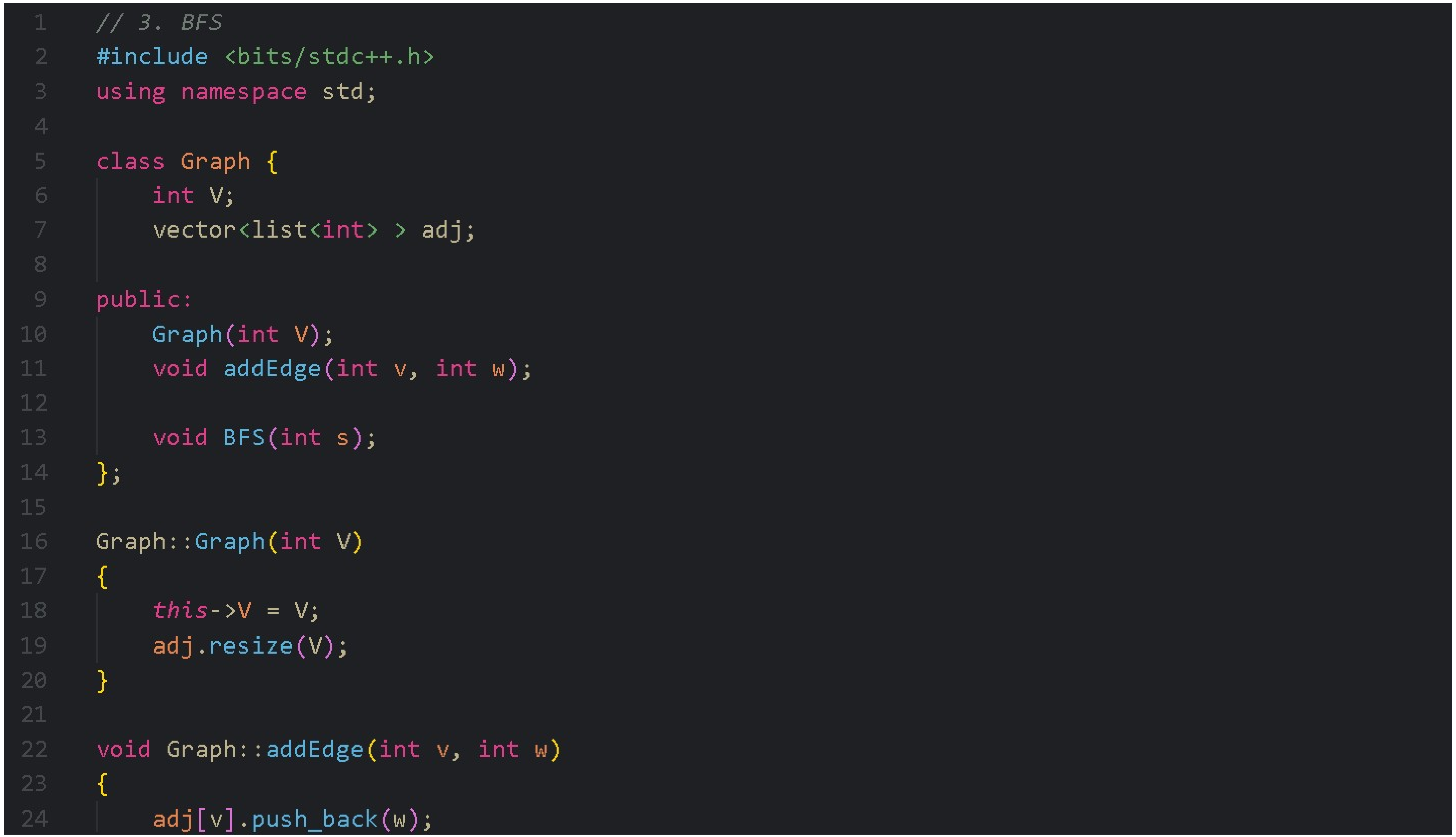


```cpp
// 2. DFS
#include <bits/stdc++.h>
using namespace std;

class Graph {
public:
    map<int, bool> visited;
    map<int, list<int> > adj;

    void addEdge(int v, int w);

    void DFS(int v);
};

void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w);
}

void Graph::DFS(int v)
{
    visited[v] = true;
    cout << v << " ";

```

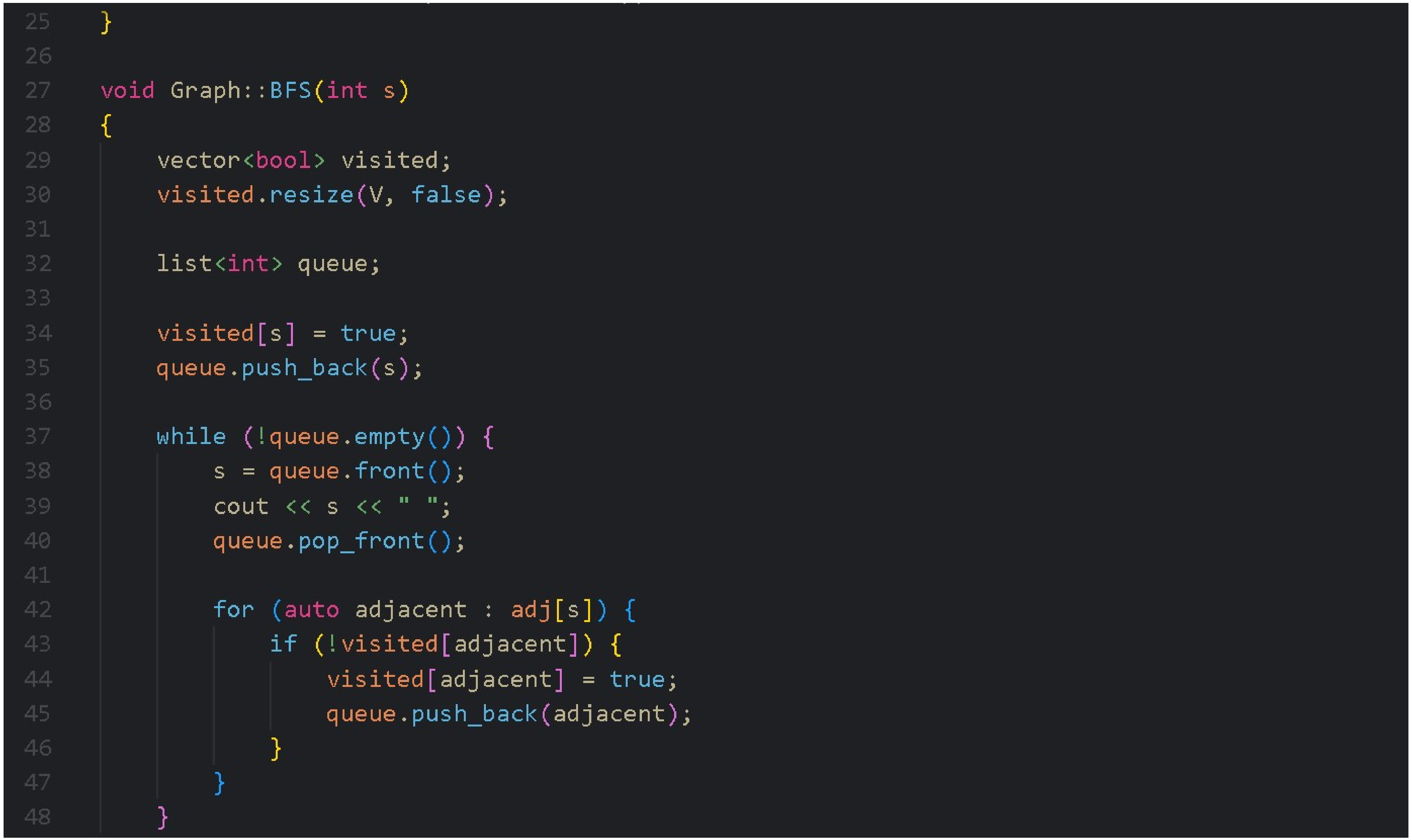

```
    list<int>::iterator i;
    for (i = adj[v].begin(); i != adj[v].end(); ++i)
        if (!visited[*i])
            DFS(*i);
}

int main()
{
    Graph g;
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);

    cout << "Following is Depth First Traversal"
            " (starting from vertex 2) \n";

    g.DFS(2);

    return 0;
}
```



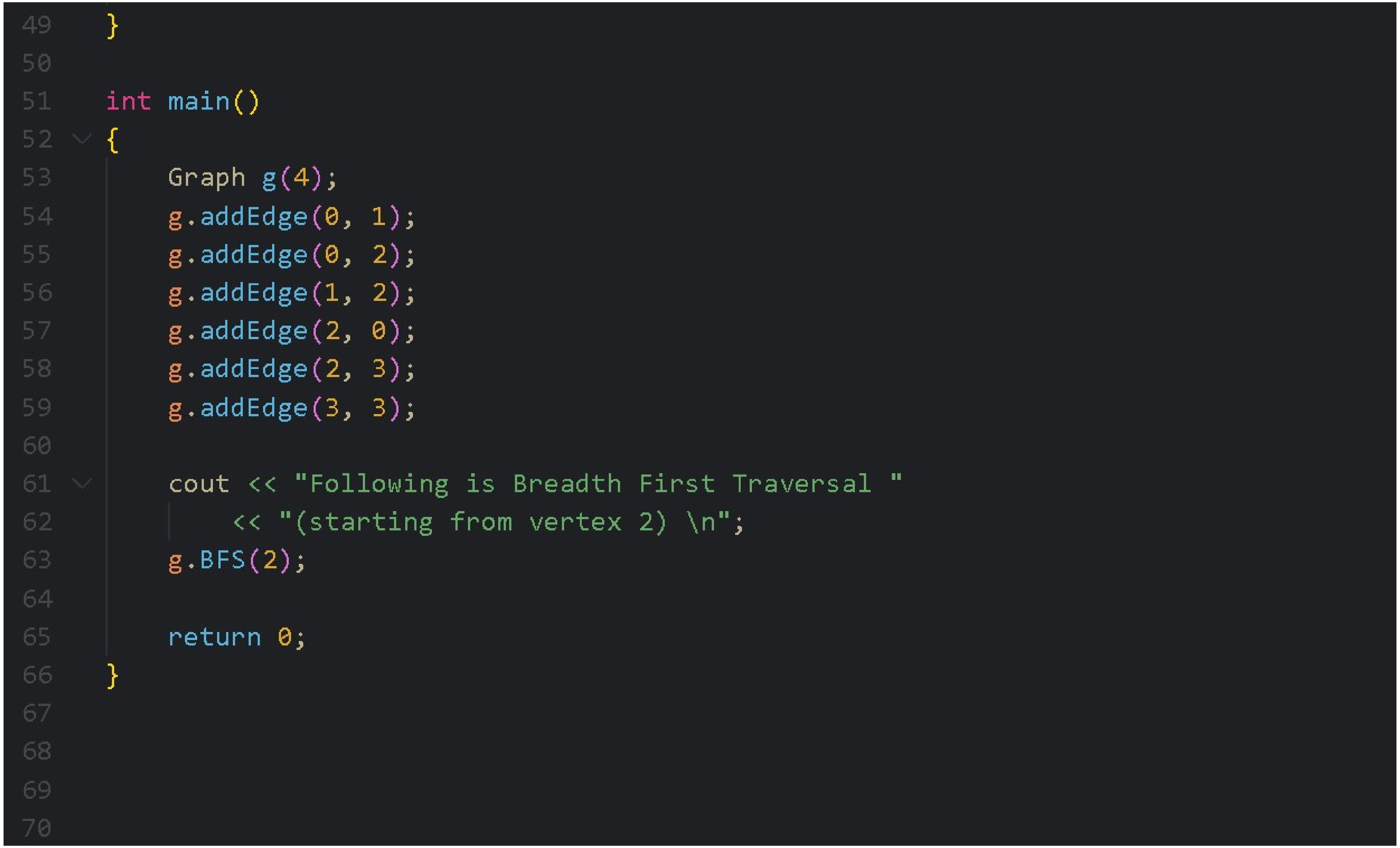
```
Following is Depth First Traversal (starting from vertex 2)
2 0 1 3
PS C:\Users\91830\OneDrive\Desktop\output>
```



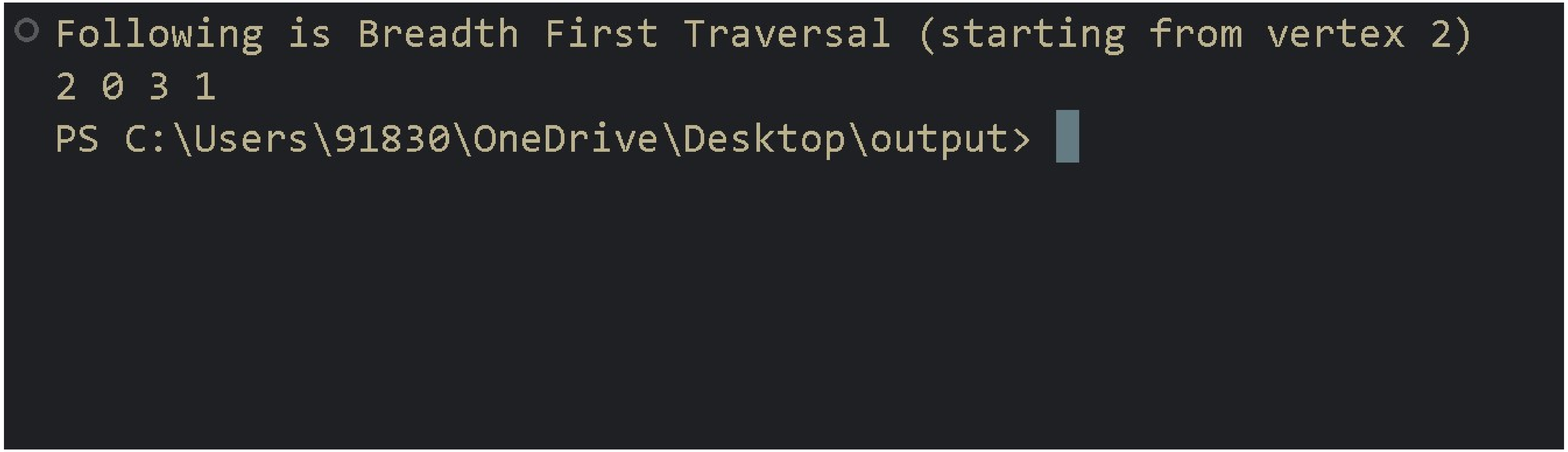
```cpp
// 3. BFS
#include <bits/stdc++.h>
using namespace std;

class Graph {
    int V;
    vector<list<int> > adj;

public:
    Graph(int V);
    void addEdge(int v, int w);

    void BFS(int s);
};

Graph::Graph(int V)
{
    this->V = V;
    adj.resize(V);
}

void Graph::addEdge(int v, int w)
{
    adj[v].push_back(w);
```



```cpp
}

void Graph::BFS(int s)
{
    vector<bool> visited;
    visited.resize(V, false);

    list<int> queue;

    visited[s] = true;
    queue.push_back(s);

    while (!queue.empty()) {
        s = queue.front();
        cout << s << " ";
        queue.pop_front();

        for (auto adjacent : adj[s]) {
            if (!visited[adjacent]) {
                visited[adjacent] = true;
                queue.push_back(adjacent);
            }
        }
    }
```



```cpp
}

int main()
{
    Graph g(4);
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);

    cout << "Following is Breadth First Traversal "
        << "(starting from vertex 2) \n";
    g.BFS(2);

    return 0;
}
```



```
Following is Breadth First Traversal (starting from vertex 2)
2 0 3 1
PS C:\Users\91830\OneDrive\Desktop\output>
```

THANK
YOU!