Quadratic Equation and Quadratic Formula

$$ax^2 + bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Order Matters	Repetition Allowed	Formula
Yes (Permutation)	Yes	$P(n, r) = n^r$
Yes (Permutation)	No	$P(n, r) = \frac{n!}{(n - r)!}$
No (Combination)	No	$C(n, r) = \frac{n!}{r!(n - r)!}$
No (Combination)	Yes	$C(n + r - 1, r) = \frac{(n + r - 1)!}{r!(n - 1)!}$

সমান্তর ধারার সূত্রাবলি

সমান্তর ধারার প্রথম পদ a, শেষপদ l এবং সাধারণ অন্তর d হলে,

- সাধারণ অন্তর, d = (পরপদ - পূর্বপদ)
- n তম পদ = a + (n-1) d
- প্রথম n পদের সমষ্টি, S = $\frac{n(a+l)}{2} = \frac{n}{2}\{2a + (n-1)d\}$

গুণোত্তর ধারার সূত্রাবলি

কোনো গুণোত্তর ধারার প্রথম পদ a এবং সাধারণ অনুপাত r হলে

- n তম পদ = $ar^n - 1$
- প্রথম n পদের সমষ্টি = $\frac{a(r^n - 1)}{r - 1}$ যখন r > 1
- এবং প্রথম n পদের সমষ্টি = $\frac{a(1 - r^n)}{1 - r}$ যখন r < 1

Some useful Method

```cpp
int n= -5;
abs(n); // "5"
```

```cpp
string s;
getline(cin, s);
```

```cpp
string s = "abcd"; // will be "abc"
s.erase(3, 1); // Index 3, remove 1 character
```

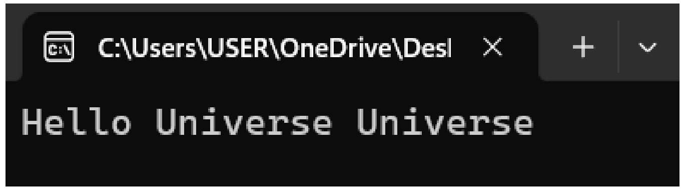
```cpp
while(cin>>n>>a){
    .......
}
```

```cpp
string a="AbcD";
transform(a.begin(), a.end(), a.begin(), ::tolower);
transform(a.begin(), a.end(), a.begin(), ::toupper);
```

```cpp
sort(_array, _array + size);
sort(_array, _array + size, greater<int>());
```

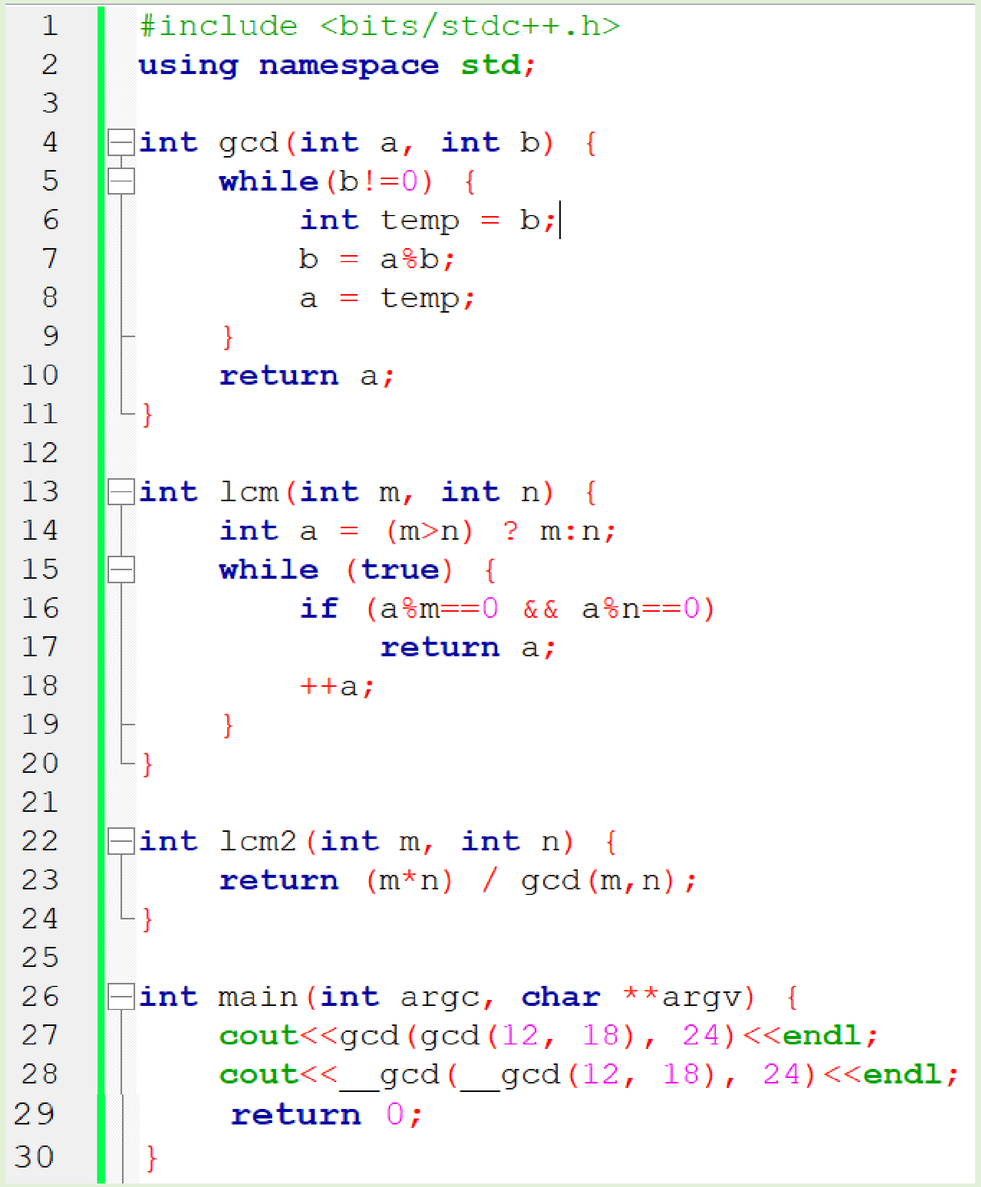
```cpp
cin>>date; // 25/07/2024 //string
stringstream ss(date);
string token;

getline(ss, token, '/');
int day = stoi(token);
getline(ss, token, '/');
int month = stoi(token);
getline(ss, token, '/');
int year = stoi(token);
```

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

int main() {
    std::string S = "Hello World World";
    std::string W = "World", R = "Universe";

    size_t pos = 0;
    while ((pos = S.find(W, pos)) != std::string::npos) {
        S.replace(pos, W.size(), R);
        pos += R.size();
    }
    cout<<S<<endl;
    return 0;
}
```



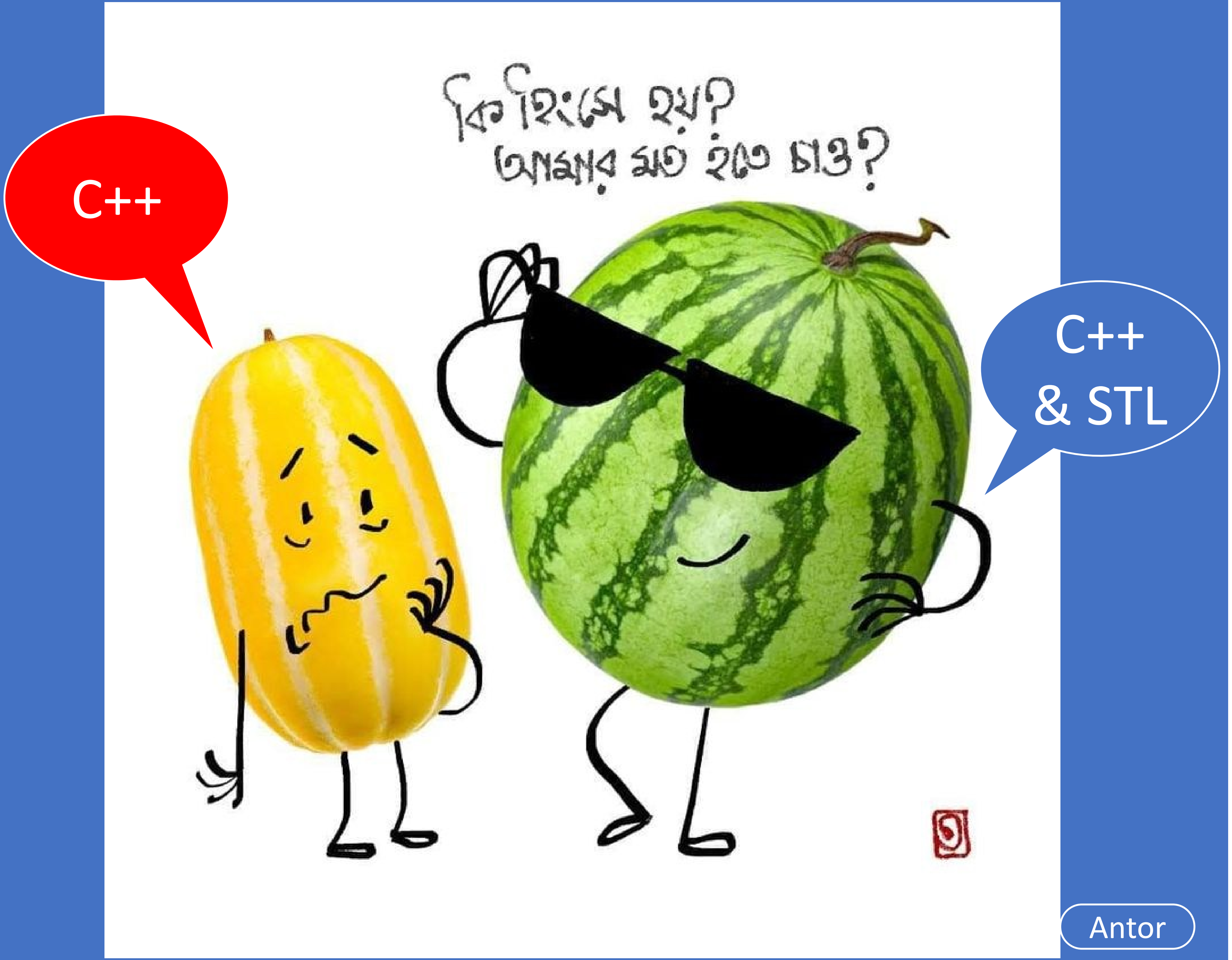
```
Hello Universe Universe
```

GCD & LCM



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

int gcd(int a, int b) {
    while(b!=0) {
        int temp = b;
        b = a%b;
        a = temp;
    }
    return a;
}

int lcm(int m, int n) {
    int a = (m>n) ? m:n;
    while (true) {
        if (a%m==0 && a%n==0)
            return a;
        ++a;
    }
}

int lcm2(int m, int n) {
    return (m*n) / gcd(m,n);
}

int main(int argc, char **argv) {
    cout<<gcd(gcd(12, 18), 24)<<endl;
    cout<<__gcd(__gcd(12, 18), 24)<<endl;
     return 0;
 }
```

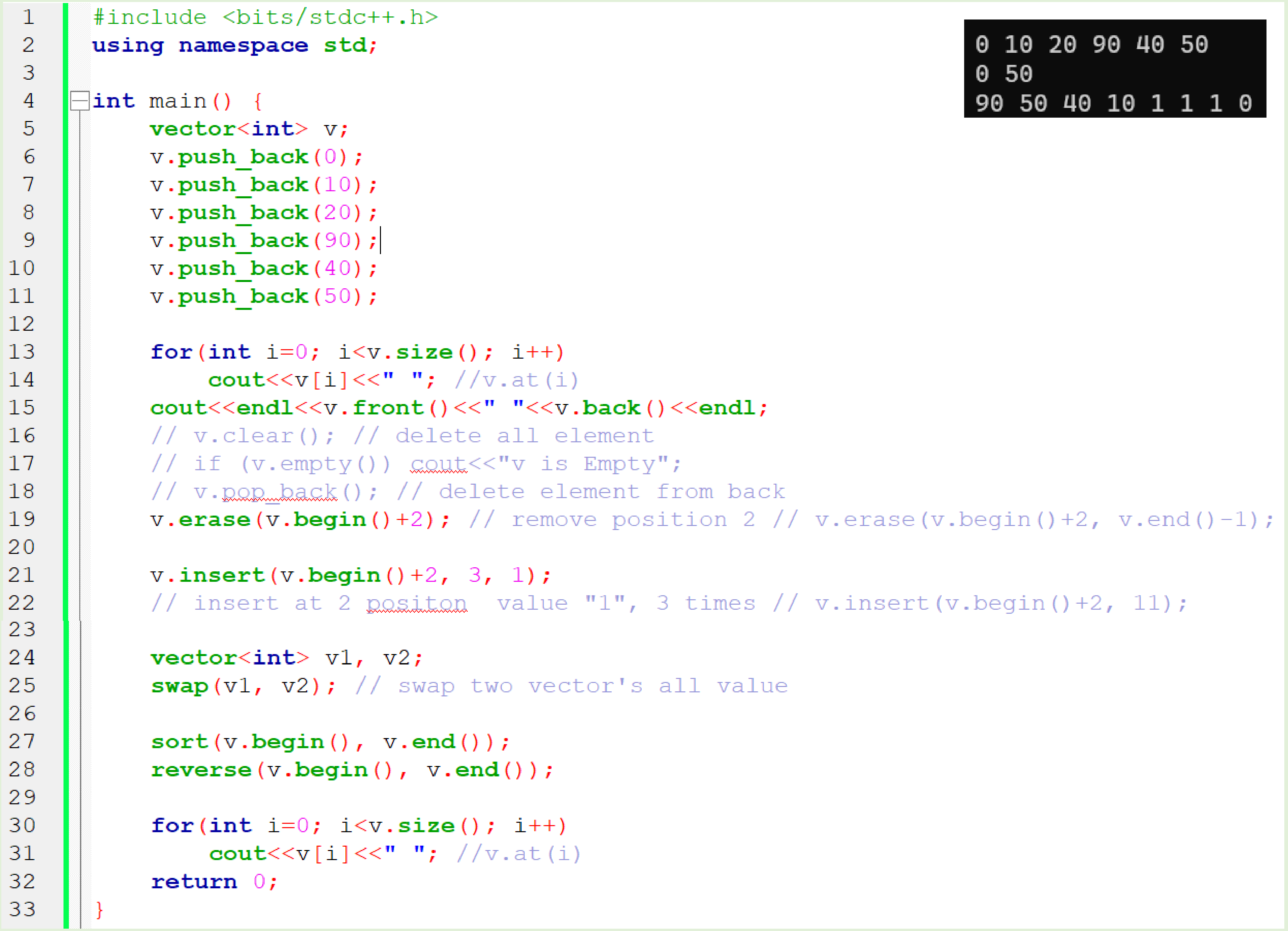
C++ (STL)



CSE Department, SMUCT

Antor

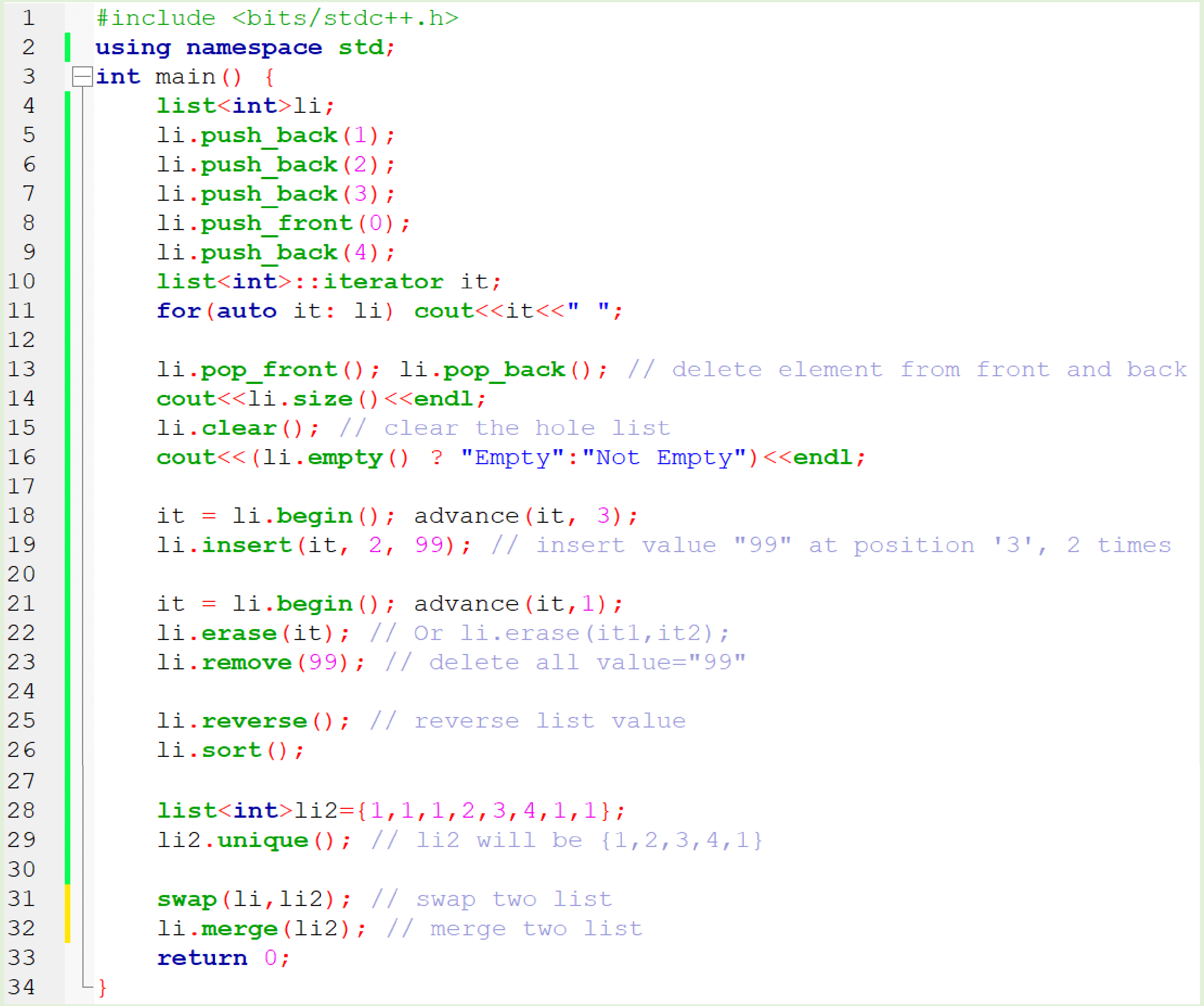
Vector



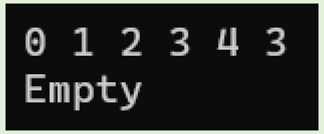
```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    vector<int> v;
    v.push_back(0);
    v.push_back(10);
    v.push_back(20);
    v.push_back(90);
    v.push_back(40);
    v.push_back(50);

    for(int i=0; i<v.size(); i++)
        cout<<v[i]<<" "; //v.at(i)
    cout<<endl<<v.front()<<" "<<v.back()<<endl;
    // v.clear(); // delete all element
    // if (v.empty()) cout<<"v is Empty";
    // v.pop_back(); // delete element from back
    v.erase(v.begin()+2); // remove position 2 // v.erase(v.begin()+2, v.end()-1);

    v.insert(v.begin()+2, 3, 1);
    // insert at 2 positon  value "1", 3 times // v.insert(v.begin()+2, 11);

    vector<int> v1, v2;
    swap(v1, v2); // swap two vector's all value

    sort(v.begin(), v.end());
    reverse(v.begin(), v.end());

    for(int i=0; i<v.size(); i++)
        cout<<v[i]<<" "; //v.at(i)
    return 0;
}
```

```
0 10 20 90 40 50
0 50
90 50 40 10 1 1 1 0
```

List

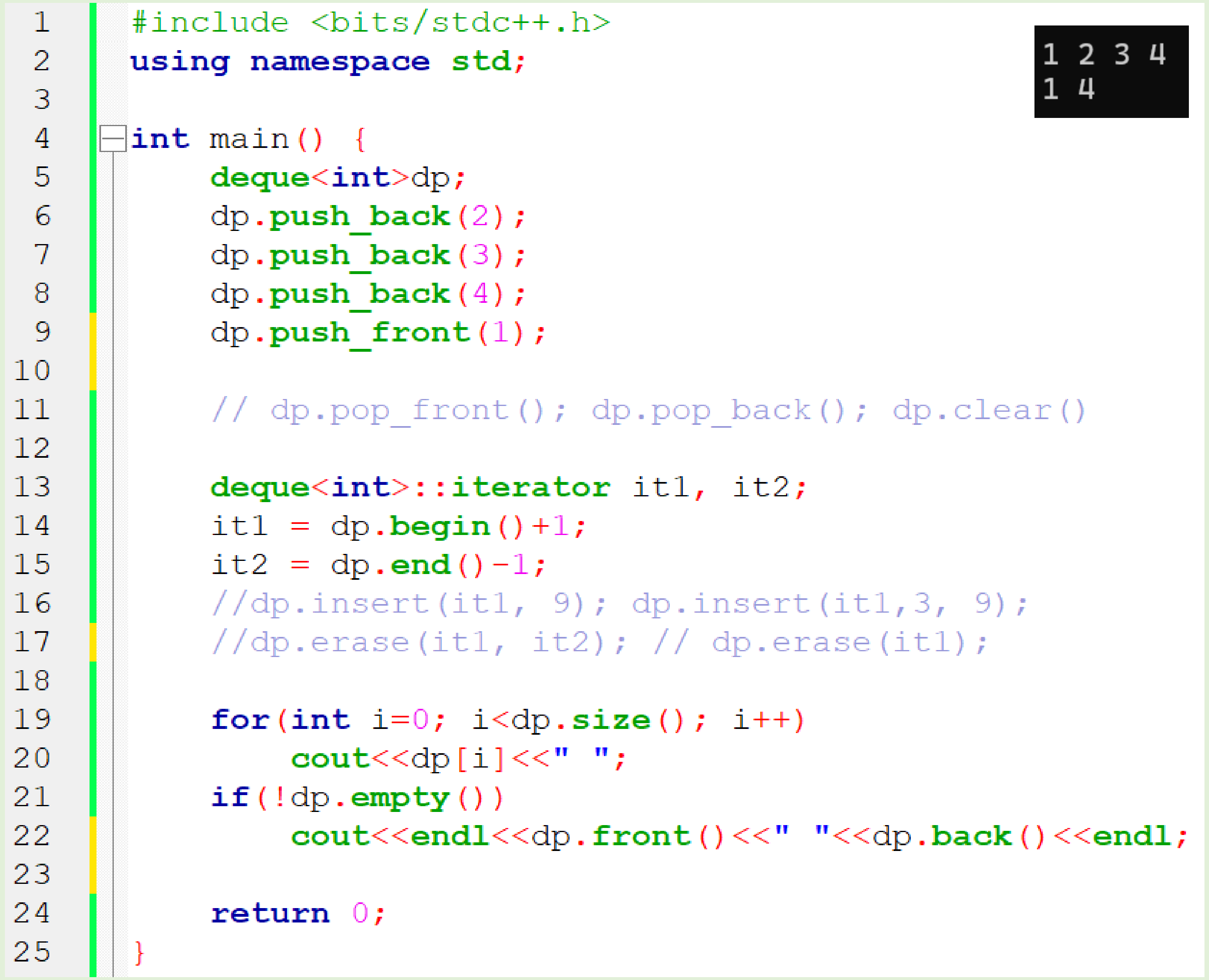


```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    list<int>li;
    li.push_back(1);
    li.push_back(2);
    li.push_back(3);
    li.push_front(0);
    li.push_back(4);
    list<int>::iterator it;
    for(auto it: li) cout<<it<<" ";

    li.pop_front(); li.pop_back(); // delete element from front and back
    cout<<li.size()<<endl;
    li.clear(); // clear the hole list
    cout<<(li.empty() ? "Empty":"Not Empty")<<endl;

    it = li.begin(); advance(it, 3);
    li.insert(it, 2, 99); // insert value "99" at position '3', 2 times

    it = li.begin(); advance(it,1);
    li.erase(it); // Or li.erase(it1,it2);
    li.remove(99); // delete all value="99"

    li.reverse(); // reverse list value
    li.sort();

    list<int>li2={1,1,1,2,3,4,1,1};
    li2.unique(); // li2 will be {1,2,3,4,1}

    swap(li,li2); // swap two list
    li.merge(li2); // merge two list
    return 0;
}
```



```
0 1 2 3 4 3
Empty
```

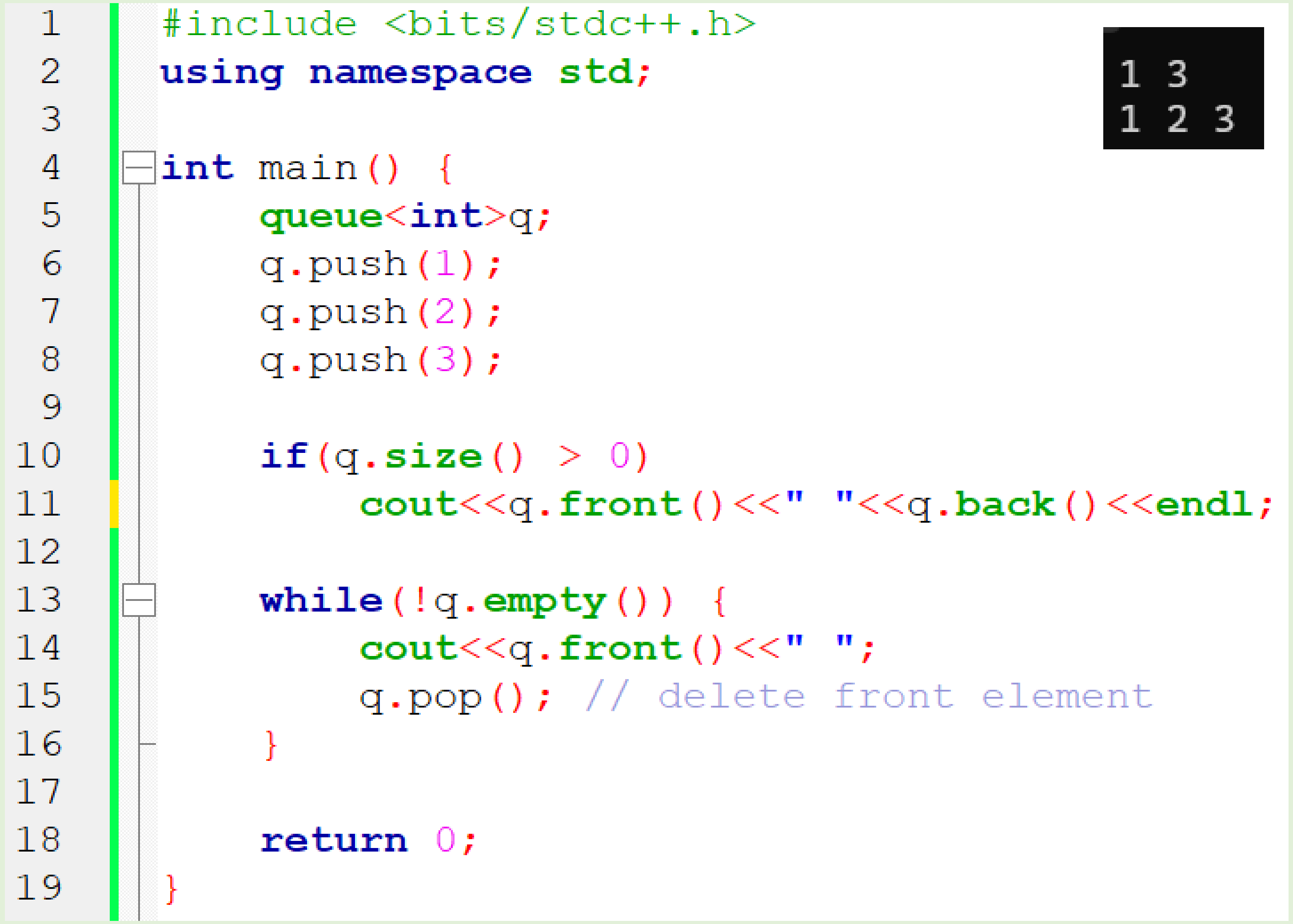
Deque



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

int main() {
    deque<int>dp;
    dp.push_back(2);
    dp.push_back(3);
    dp.push_back(4);
    dp.push_front(1);

    // dp.pop_front(); dp.pop_back(); dp.clear()

    deque<int>::iterator it1, it2;
    it1 = dp.begin()+1;
    it2 = dp.end()-1;
    //dp.insert(it1, 9); dp.insert(it1,3, 9);
    //dp.erase(it1, it2); // dp.erase(it1);

    for(int i=0; i<dp.size(); i++)
        cout<<dp[i]<<" ";
    if(!dp.empty())
        cout<<endl<<dp.front()<<" "<<dp.back()<<endl;

    return 0;
}
```

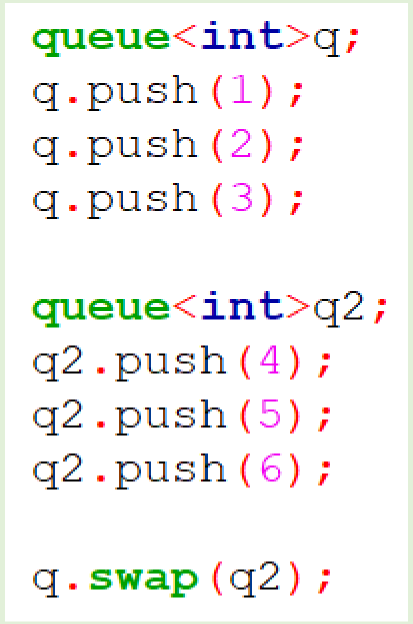
```
1 2 3 4
1 4
```

Queue



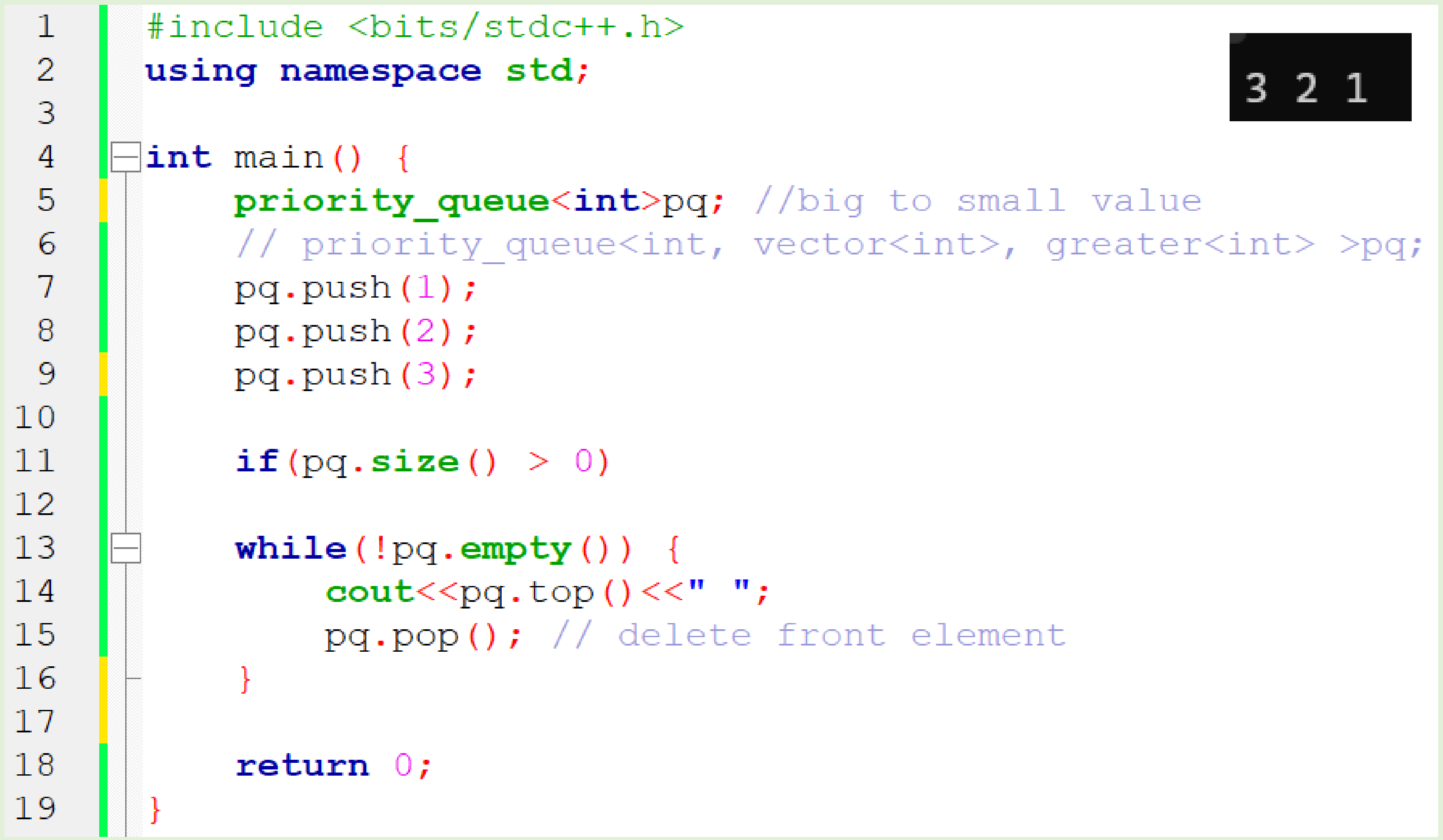
```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    queue<int>q;
    q.push(1);
    q.push(2);
    q.push(3);

    if(q.size() > 0)
        cout<<q.front()<<" "<<q.back()<<endl;

    while(!q.empty()) {
        cout<<q.front()<<" ";
        q.pop(); // delete front element
    }

    return 0;
}
```

```
1 3
1 2 3
```



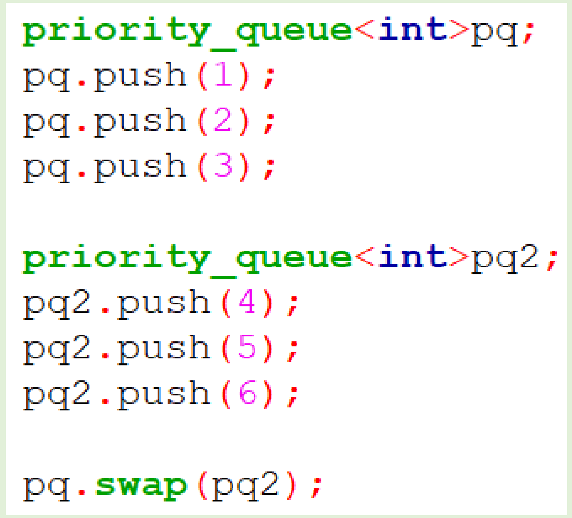
```cpp
queue<int>q;
q.push(1);
q.push(2);
q.push(3);

queue<int>q2;
q2.push(4);
q2.push(5);
q2.push(6);

q.swap(q2);
```

Priority Queue

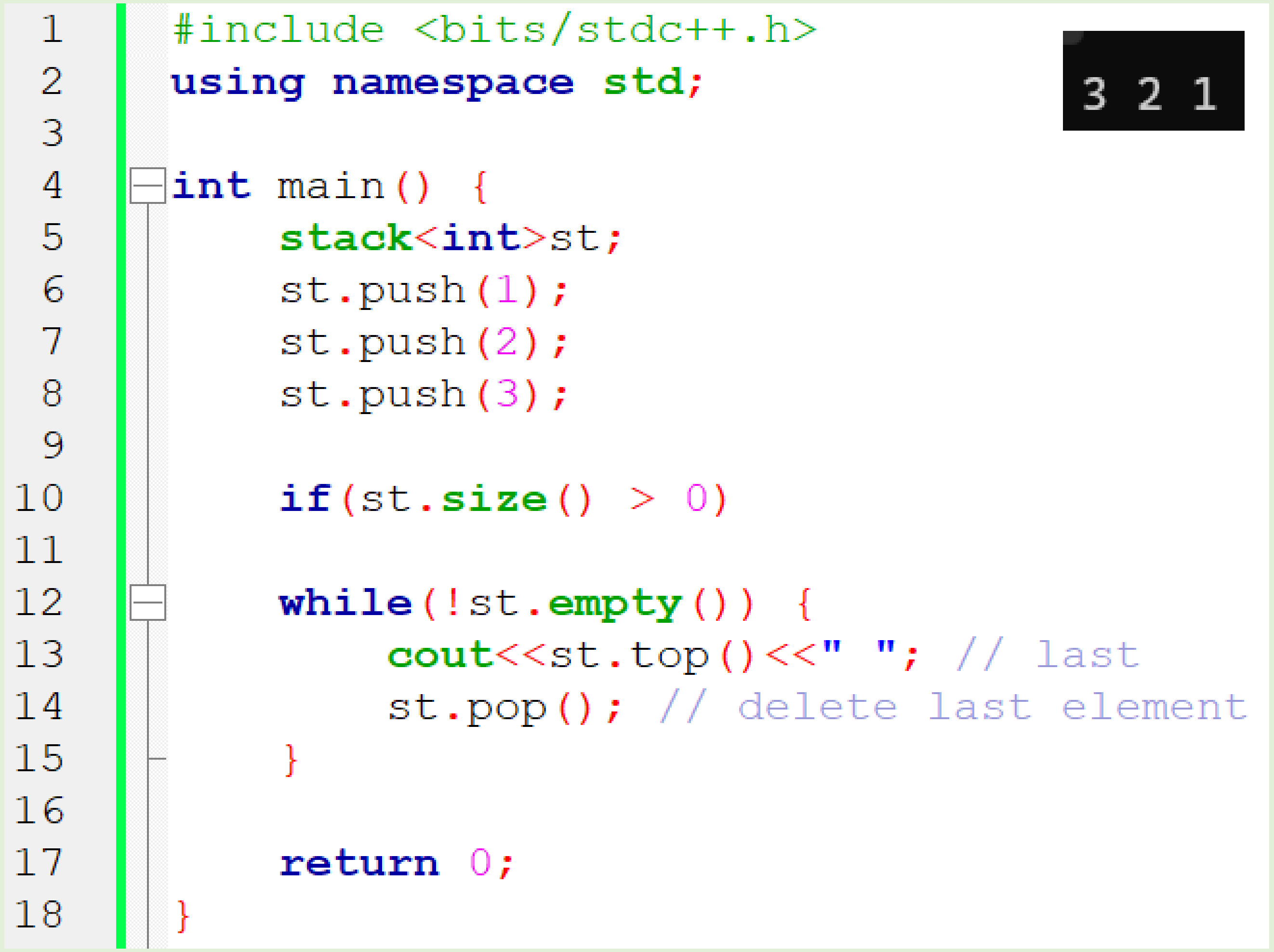


```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    priority_queue<int>pq; //big to small value
    // priority_queue<int, vector<int>, greater<int> >pq;
    pq.push(1);
    pq.push(2);
    pq.push(3);

    if(pq.size() > 0)

    while(!pq.empty()) {
        cout<<pq.top()<<" ";
        pq.pop(); // delete front element
    }

    return 0;
}
```

```
3 2 1
```



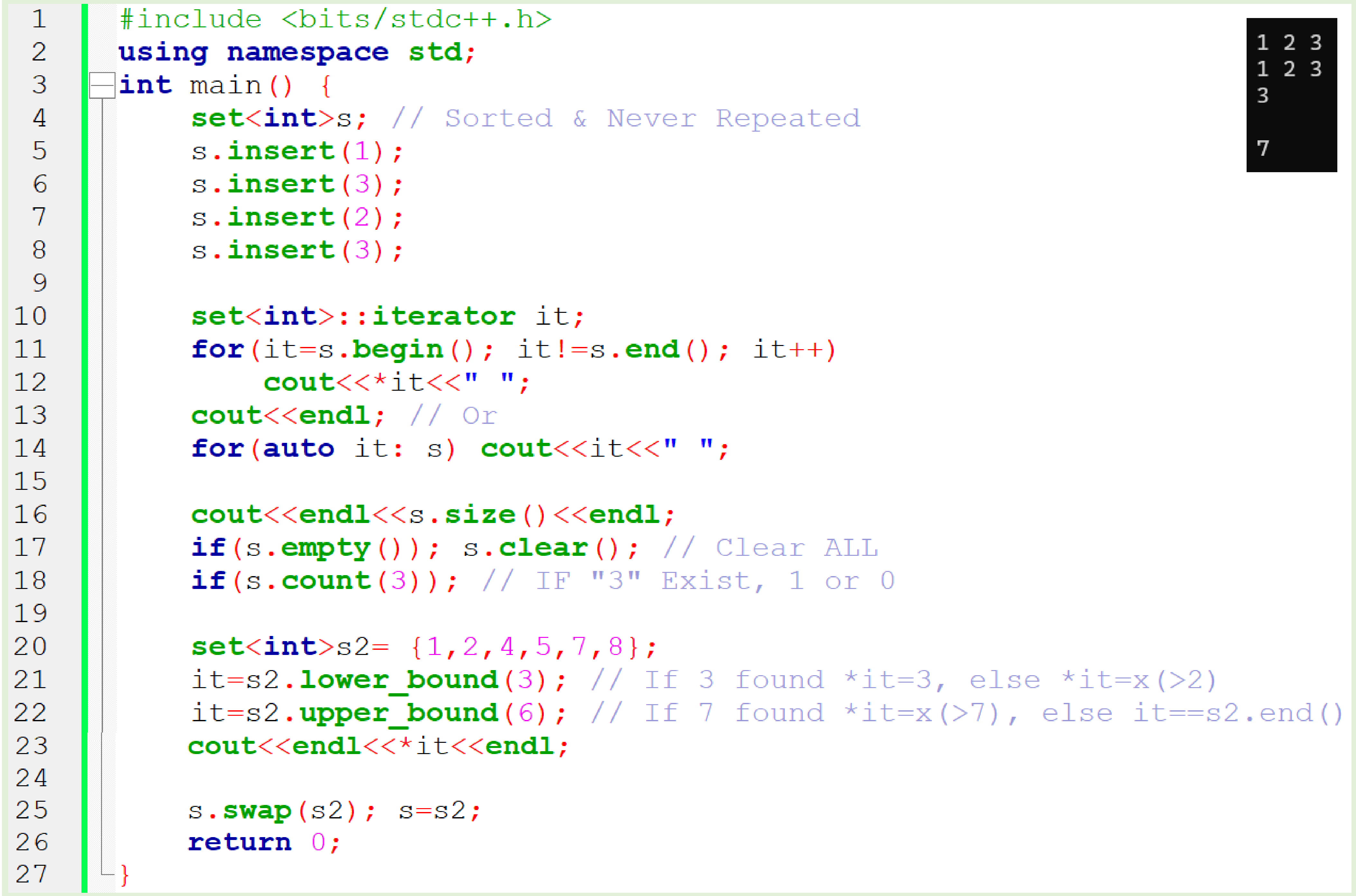
```cpp
priority_queue<int>pq;
pq.push(1);
pq.push(2);
pq.push(3);

priority_queue<int>pq2;
pq2.push(4);
pq2.push(5);
pq2.push(6);

pq.swap(pq2);
```



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

int main() {
    stack<int>st;
    st.push(1);
    st.push(2);
    st.push(3);

    if(st.size() > 0)

    while(!st.empty()) {
        cout<<st.top()<<" "; // last
        st.pop(); // delete last element
    }

    return 0;
}
```

```
3 2 1
```

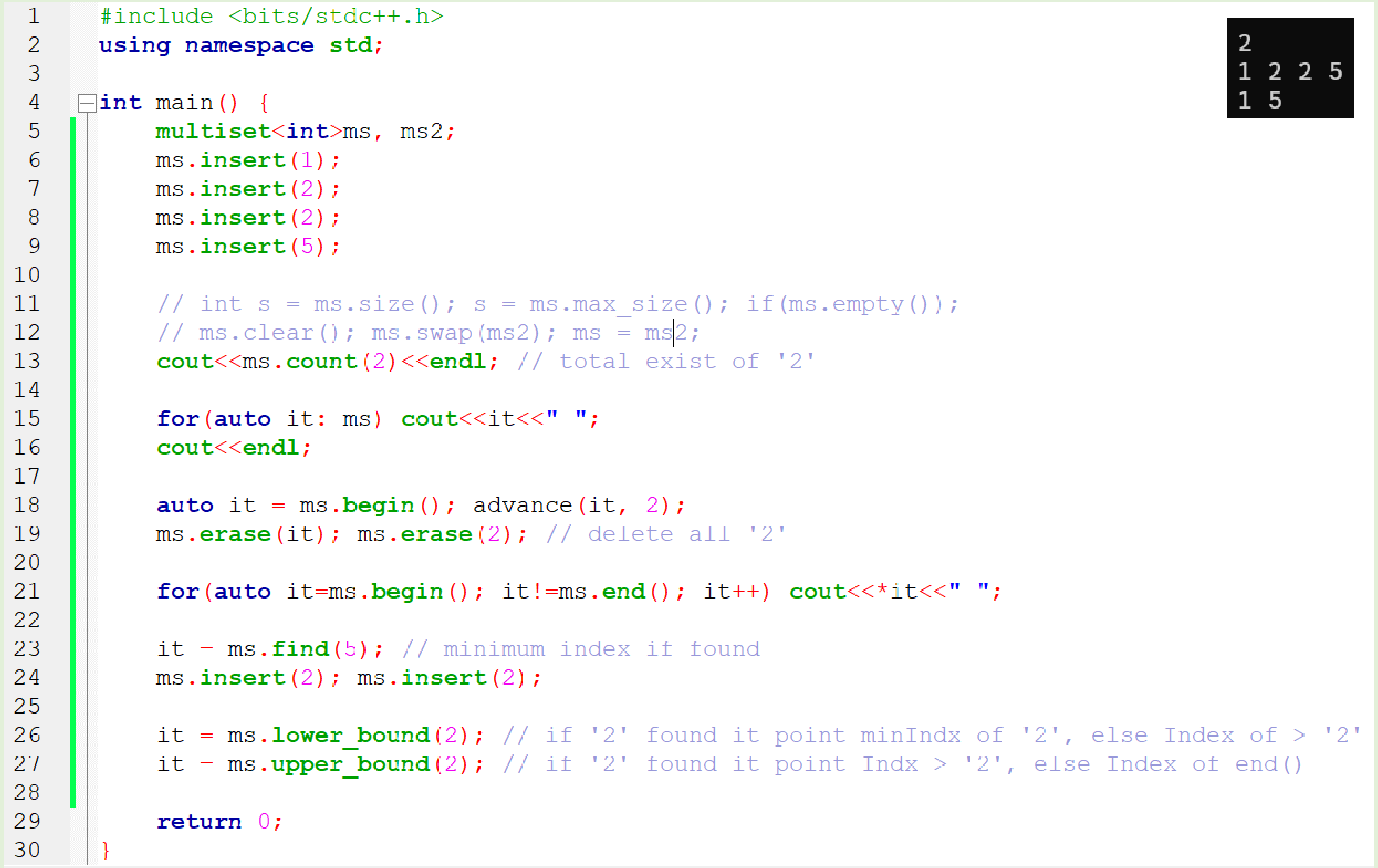
Set



```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    set<int>s; // Sorted & Never Repeated
    s.insert(1);
    s.insert(3);
    s.insert(2);
    s.insert(3);

    set<int>::iterator it;
    for(it=s.begin(); it!=s.end(); it++)
        cout<<*it<<" ";
    cout<<endl; // Or
    for(auto it: s) cout<<it<<" ";

    cout<<endl<<s.size()<<endl;
    if(s.empty()); s.clear(); // Clear ALL
    if(s.count(3)); // IF "3" Exist, 1 or 0

    set<int>s2= {1,2,4,5,7,8};
    it=s2.lower_bound(3); // If 3 found *it=3, else *it=x(>2)
    it=s2.upper_bound(6); // If 7 found *it=x(>7), else it==s2.end()
    cout<<endl<<*it<<endl;

    s.swap(s2); s=s2;
    return 0;
}
```

```
1 2 3
1 2 3
3

7
```

MultiSet



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

int main() {
    multiset<int>ms, ms2;
    ms.insert(1);
    ms.insert(2);
    ms.insert(2);
    ms.insert(5);

    // int s = ms.size(); s = ms.max_size(); if(ms.empty());
    // ms.clear(); ms.swap(ms2); ms = ms2;
    cout<<ms.count(2)<<endl; // total exist of '2'

    for(auto it: ms) cout<<it<<" ";
    cout<<endl;

    auto it = ms.begin(); advance(it, 2);
    ms.erase(it); ms.erase(2); // delete all '2'

    for(auto it=ms.begin(); it!=ms.end(); it++) cout<<*it<<" ";

    it = ms.find(5); // minimum index if found
    ms.insert(2); ms.insert(2);

    it = ms.lower_bound(2); // if '2' found it point minIndx of '2', else Index of > '2'
    it = ms.upper_bound(2); // if '2' found it point Indx > '2', else Index of end()

    return 0;
}
```

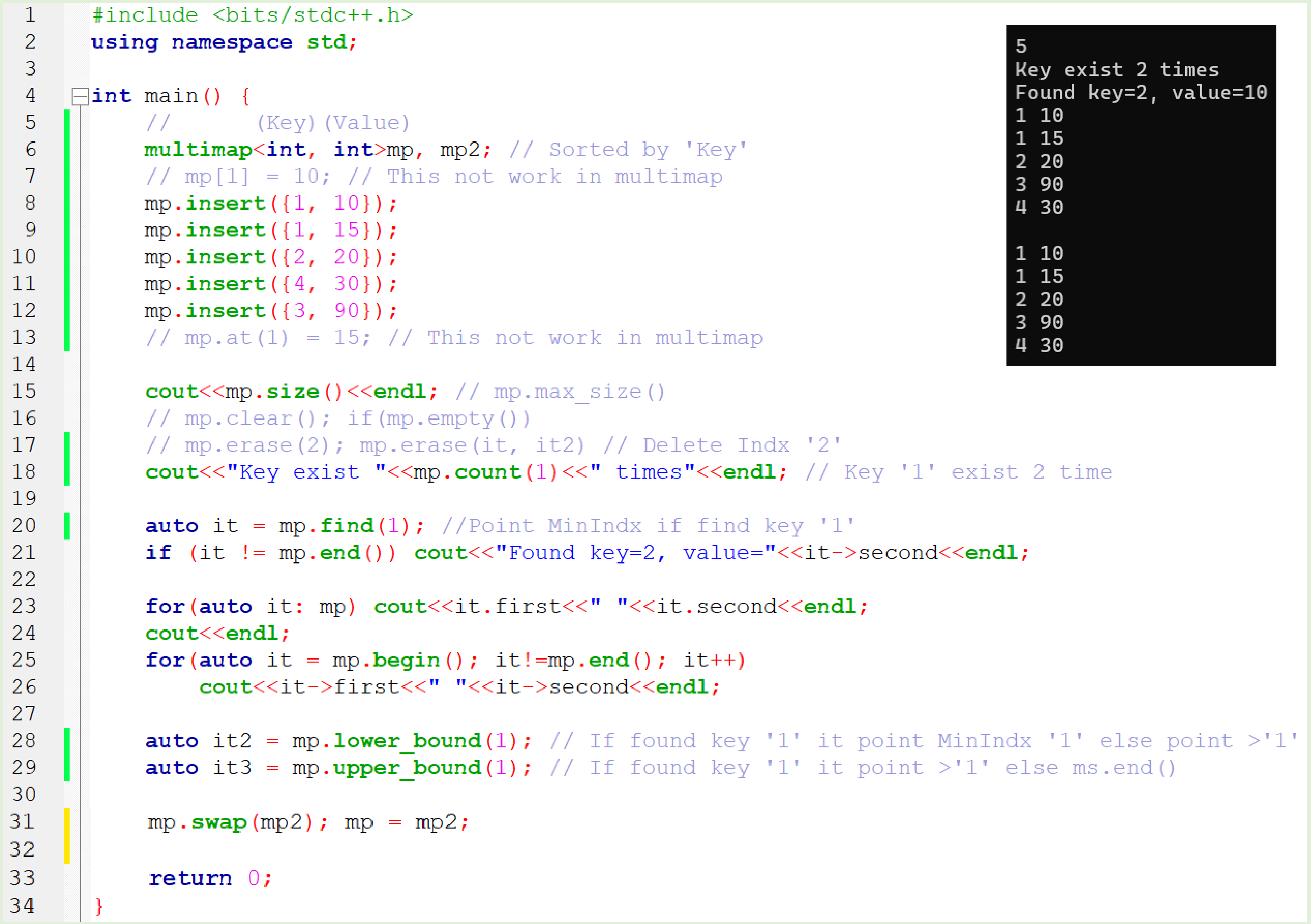
```
2
1 2 2 5
1 5
```

Map

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

int main() {
    // (Key)(Value)
    map<int, int>mp, mp2; // Sorted by 'Key'
    mp[1] = 10;
    mp.insert({2, 20});
    mp.insert({4, 30});
    mp.insert({3, 90});
    mp.insert({5, 30});
    mp.at(1) = 15; // For update value;

    cout<<mp.size()<<endl; // mp.max_size()
    // mp.clear(); if(mp.empty())
    // mp.erase(2); // Delete Indx '2'
    if(mp.count(2)) cout<<"Exist"<<endl; // Is Indx '2' is exist?

    auto it = mp.find(2); //Point Indx if find key '2'
    if (it != mp.end()) cout<<"Found key=2, value="<<it->second<<endl;

    for(auto it: mp) cout<<it.first<<" "<<it.second<<endl;
    cout<<endl;
    for(auto it = mp.begin(); it!=mp.end(); it++)
        cout<<it->first<<" "<<it->second<<endl;

    auto it2 = mp.lower_bound(1); // If found key '1' it point '1' else point >'1'
    auto it3 = mp.upper_bound(1); // If found key '1' it point >'1' else ms.end()

    mp.swap(mp2); mp = mp2;

    return 0;
}
```

```
5
Exist
Found key=2, value=20
1 15
2 20
3 90
4 30
5 30

1 15
2 20
3 90
4 30
5 30
```

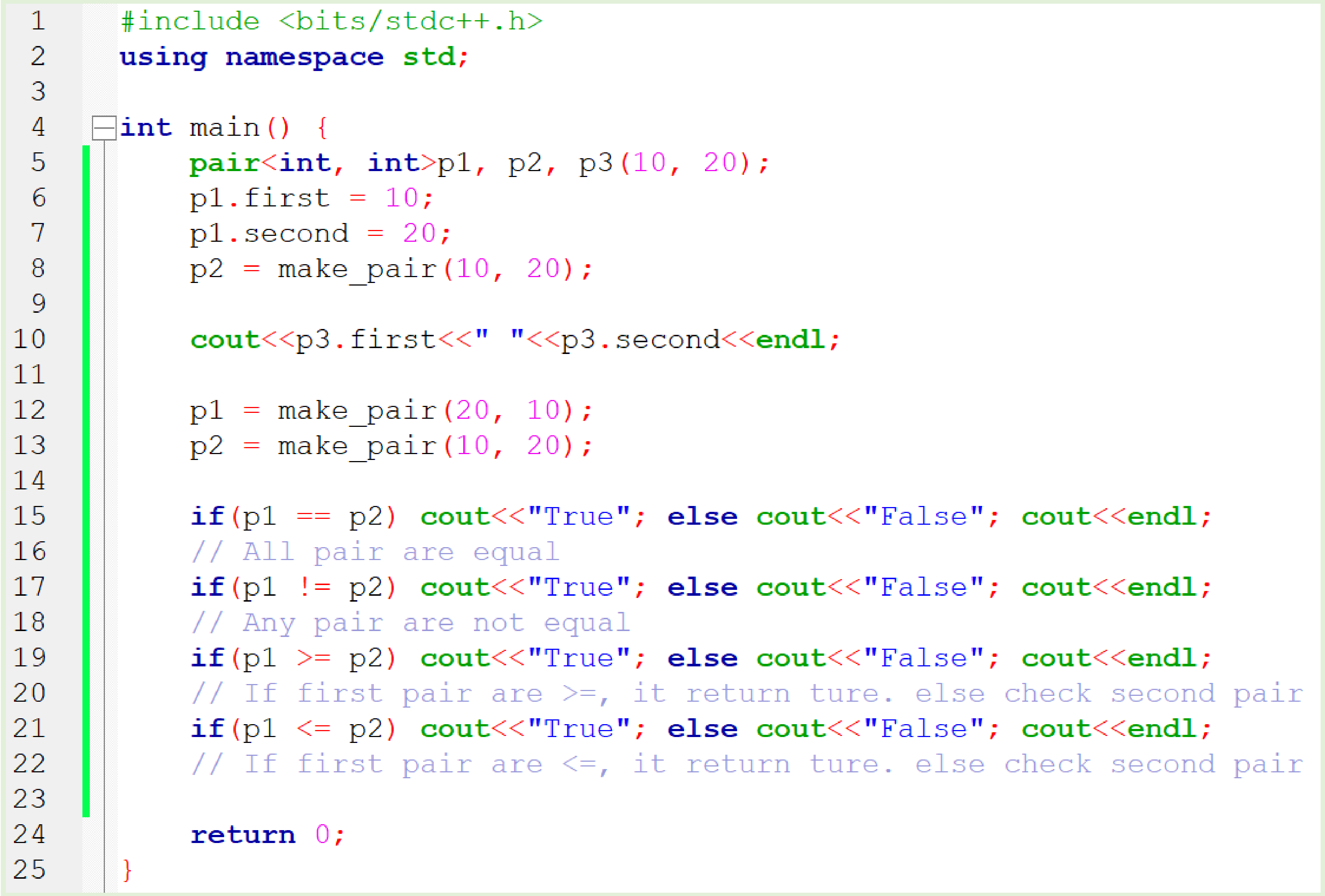
MultiMap



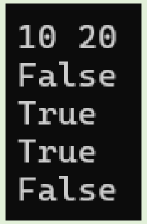
```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    //       (Key)(Value)
    multimap<int, int>mp, mp2; // Sorted by 'Key'
    // mp[1] = 10; // This not work in multimap
    mp.insert({1, 10});
    mp.insert({1, 15});
    mp.insert({2, 20});
    mp.insert({4, 30});
    mp.insert({3, 90});
    // mp.at(1) = 15; // This not work in multimap

    cout<<mp.size()<<endl; // mp.max_size()
    // mp.clear(); if(mp.empty())
    // mp.erase(2); mp.erase(it, it2) // Delete Indx '2'
    cout<<"Key exist "<<mp.count(1)<<" times"<<endl; // Key '1' exist 2 time

    auto it = mp.find(1); //Point MinIndx if find key '1'
    if (it != mp.end()) cout<<"Found key=2, value="<<it->second<<endl;

    for(auto it: mp) cout<<it.first<<" "<<it.second<<endl;
    cout<<endl;
    for(auto it = mp.begin(); it!=mp.end(); it++)
        cout<<it->first<<" "<<it->second<<endl;

    auto it2 = mp.lower_bound(1); // If found key '1' it point MinIndx '1' else point >'1'
    auto it3 = mp.upper_bound(1); // If found key '1' it point >'1' else ms.end()

    mp.swap(mp2); mp = mp2;

    return 0;
}
```

```
5
Key exist 2 times
Found key=2, value=10
1 10
1 15
2 20
3 90
4 30

1 10
1 15
2 20
3 90
4 30
```

Pair

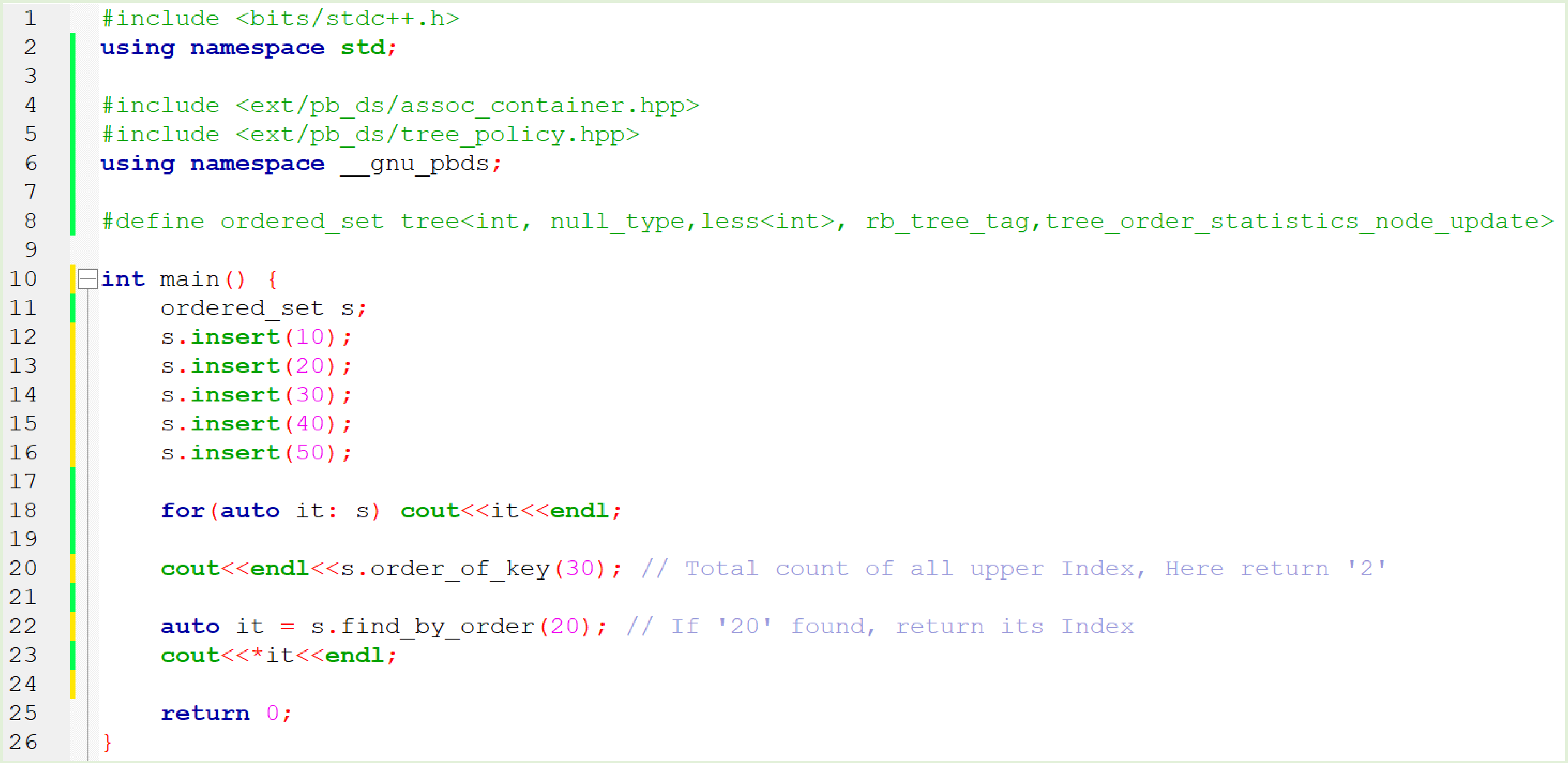


```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    pair<int, int>p1, p2, p3(10, 20);
    p1.first = 10;
    p1.second = 20;
    p2 = make_pair(10, 20);

    cout<<p3.first<<" "<<p3.second<<endl;

    p1 = make_pair(20, 10);
    p2 = make_pair(10, 20);

    if(p1 == p2) cout<<"True"; else cout<<"False"; cout<<endl;
    // All pair are equal
    if(p1 != p2) cout<<"True"; else cout<<"False"; cout<<endl;
    // Any pair are not equal
    if(p1 >= p2) cout<<"True"; else cout<<"False"; cout<<endl;
    // If first pair are >=, it return ture. else check second pair
    if(p1 <= p2) cout<<"True"; else cout<<"False"; cout<<endl;
    // If first pair are <=, it return ture. else check second pair

    return 0;
}
```



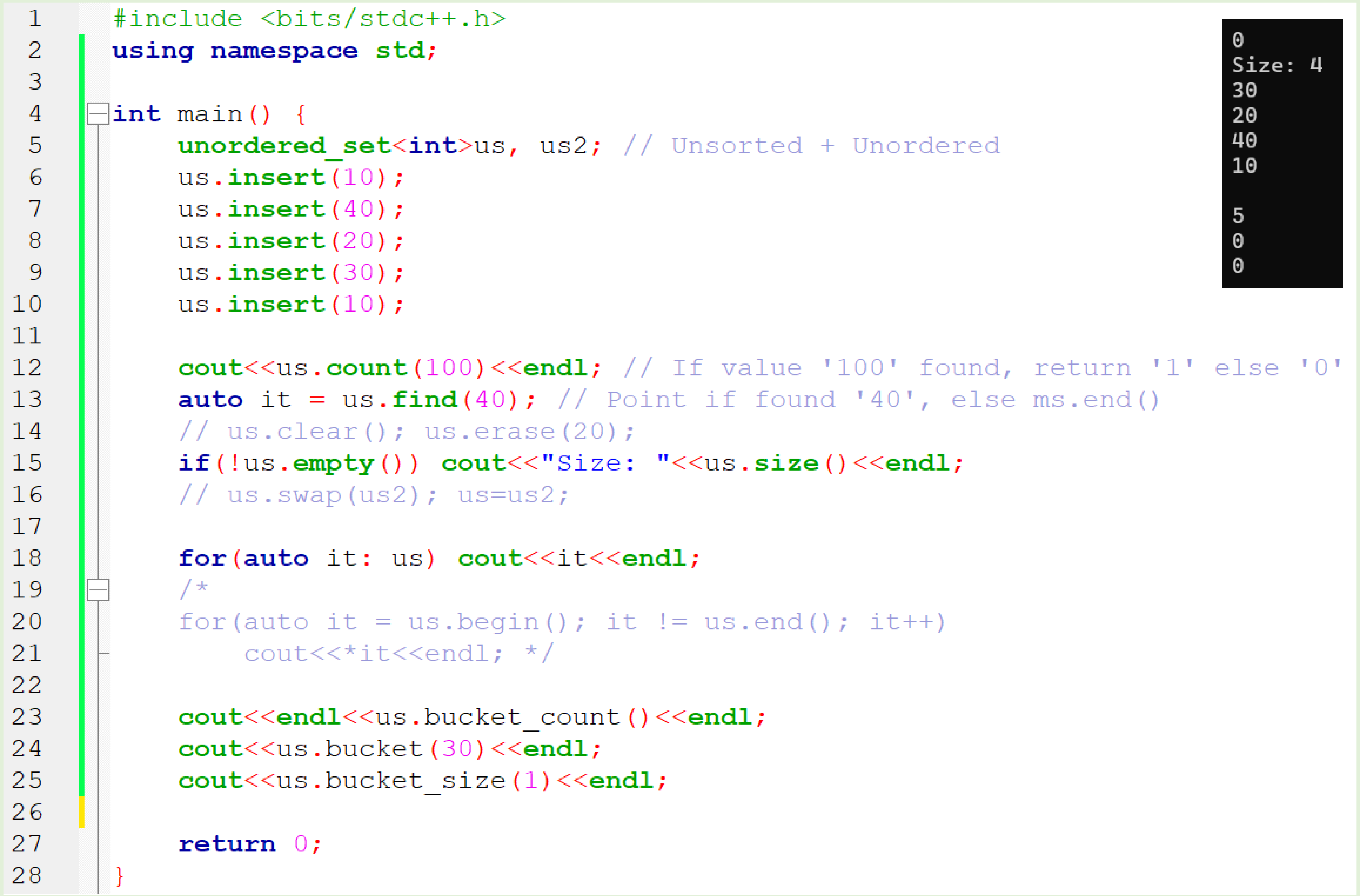
```
10 20
False
True
True
False
```

Ordered Set (Include All Set fn)



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

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;

#define ordered_set tree<int, null_type,less<int>, rb_tree_tag,tree_order_statistics_node_update>

int main() {
    ordered_set s;
    s.insert(10);
    s.insert(20);
    s.insert(30);
    s.insert(40);
    s.insert(50);

    for(auto it: s) cout<<it<<endl;

    cout<<endl<<s.order_of_key(30); // Total count of all upper Index, Here return '2'

    auto it = s.find_by_order(20); // If '20' found, return its Index
    cout<<*it<<endl;

    return 0;
}
```

Unordered Set

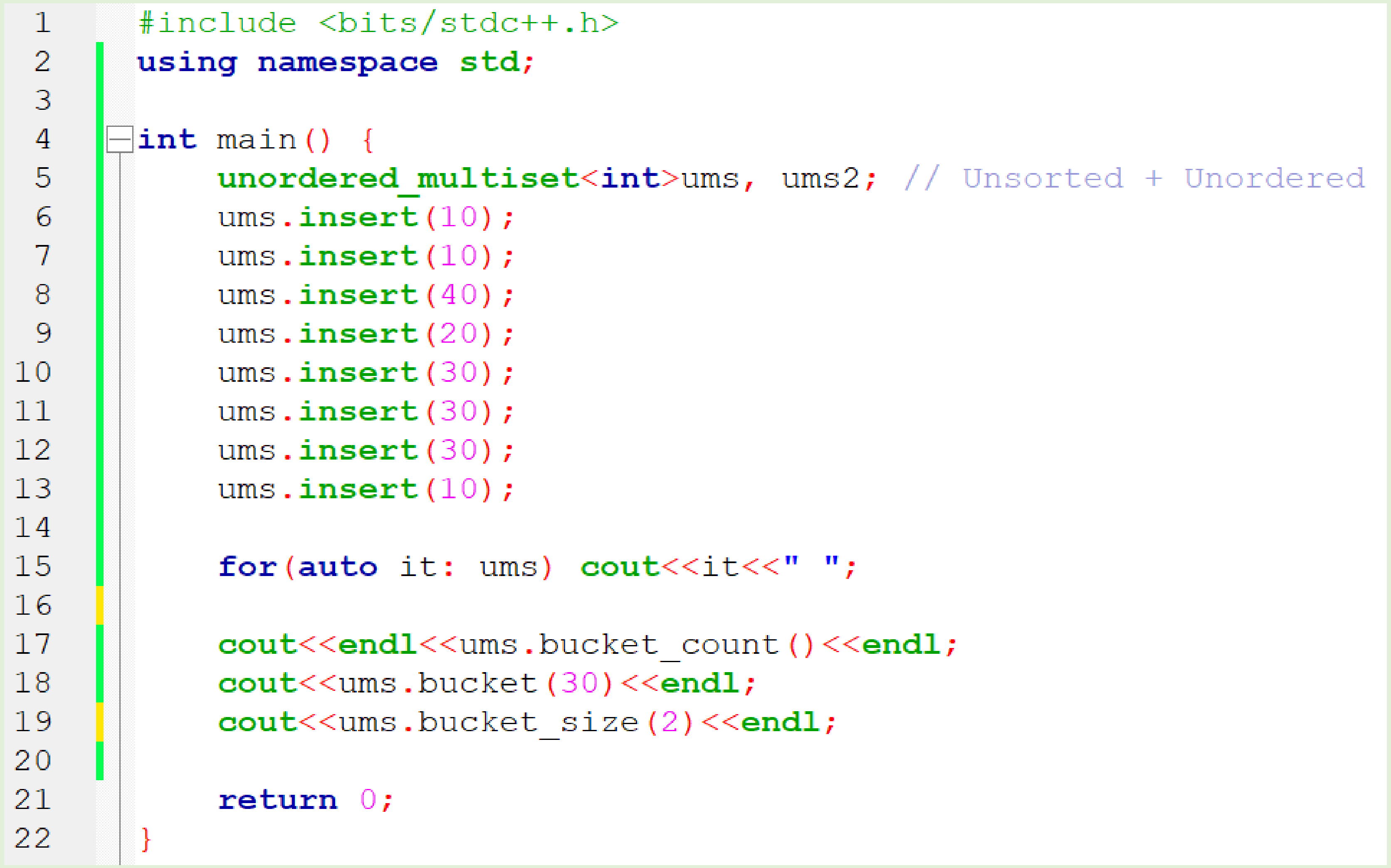


```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    unordered_set<int>us, us2; // Unsorted + Unordered
    us.insert(10);
    us.insert(40);
    us.insert(20);
    us.insert(30);
    us.insert(10);

    cout<<us.count(100)<<endl; // If value '100' found, return '1' else '0'
    auto it = us.find(40); // Point if found '40', else ms.end()
    // us.clear(); us.erase(20);
    if(!us.empty()) cout<<"Size: "<<us.size()<<endl;
    // us.swap(us2); us=us2;

    for(auto it: us) cout<<it<<endl;
    /*
    for(auto it = us.begin(); it != us.end(); it++)
        cout<<*it<<endl; */

    cout<<endl<<us.bucket_count()<<endl;
    cout<<us.bucket(30)<<endl;
    cout<<us.bucket_size(1)<<endl;

    return 0;
}
```

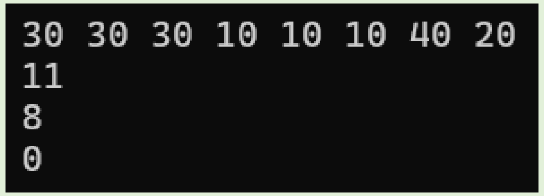
```
0
Size: 4
30
20
40
10

5
0
0
```

Unordered Multiset

(Include All Multiset fn)



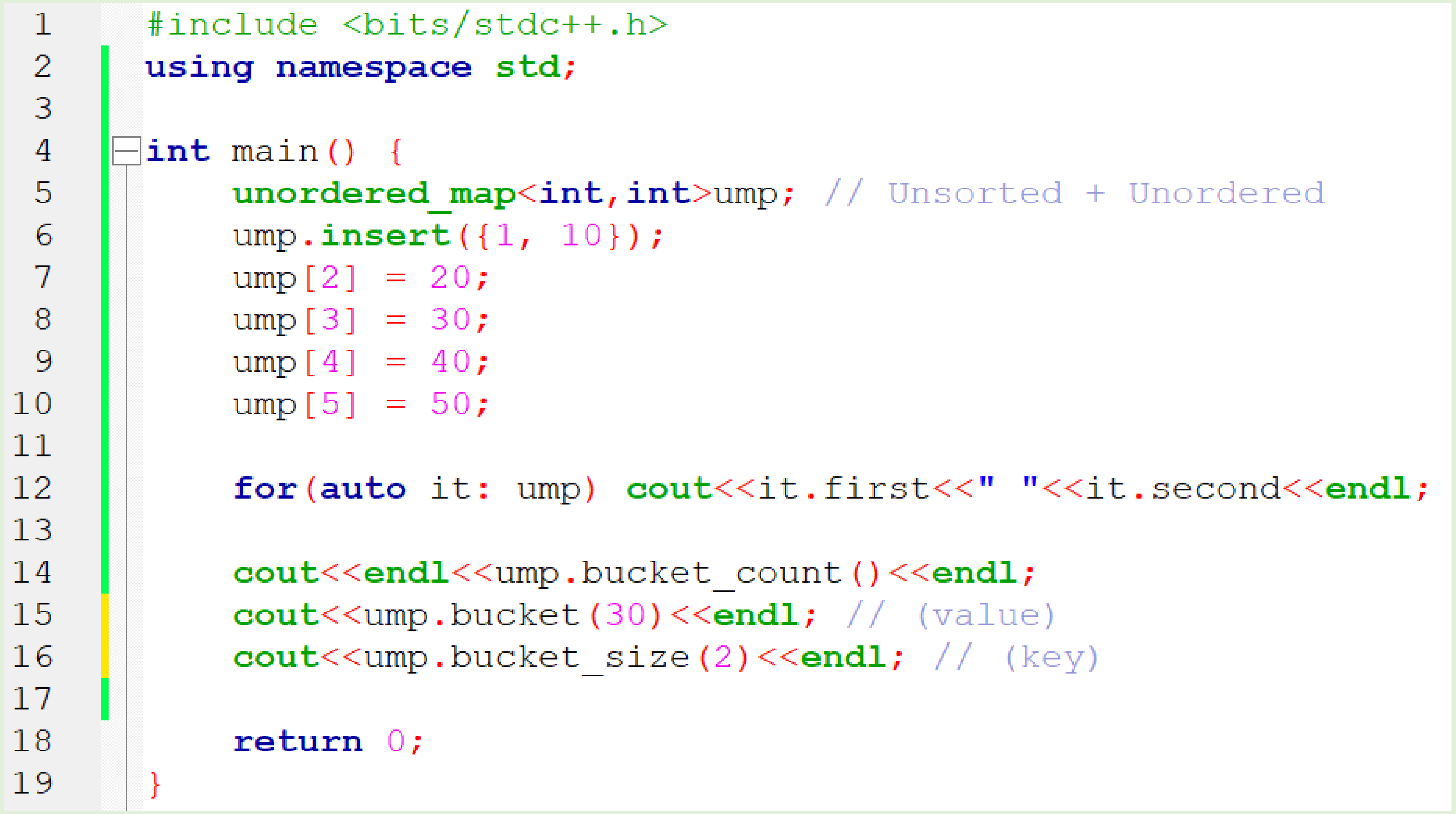
```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    unordered_multiset<int>ums, ums2; // Unsorted + Unordered
    ums.insert(10);
    ums.insert(10);
    ums.insert(40);
    ums.insert(20);
    ums.insert(30);
    ums.insert(30);
    ums.insert(30);
    ums.insert(10);

    for(auto it: ums) cout<<it<<" ";

    cout<<endl<<ums.bucket_count()<<endl;
    cout<<ums.bucket(30)<<endl;
    cout<<ums.bucket_size(2)<<endl;

    return 0;
}
```



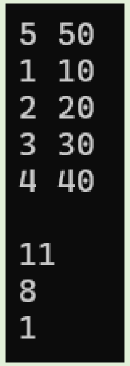
```
30 30 30 10 10 10 40 20
11
8
0
```

Unordered Map

(Include All Map fn)

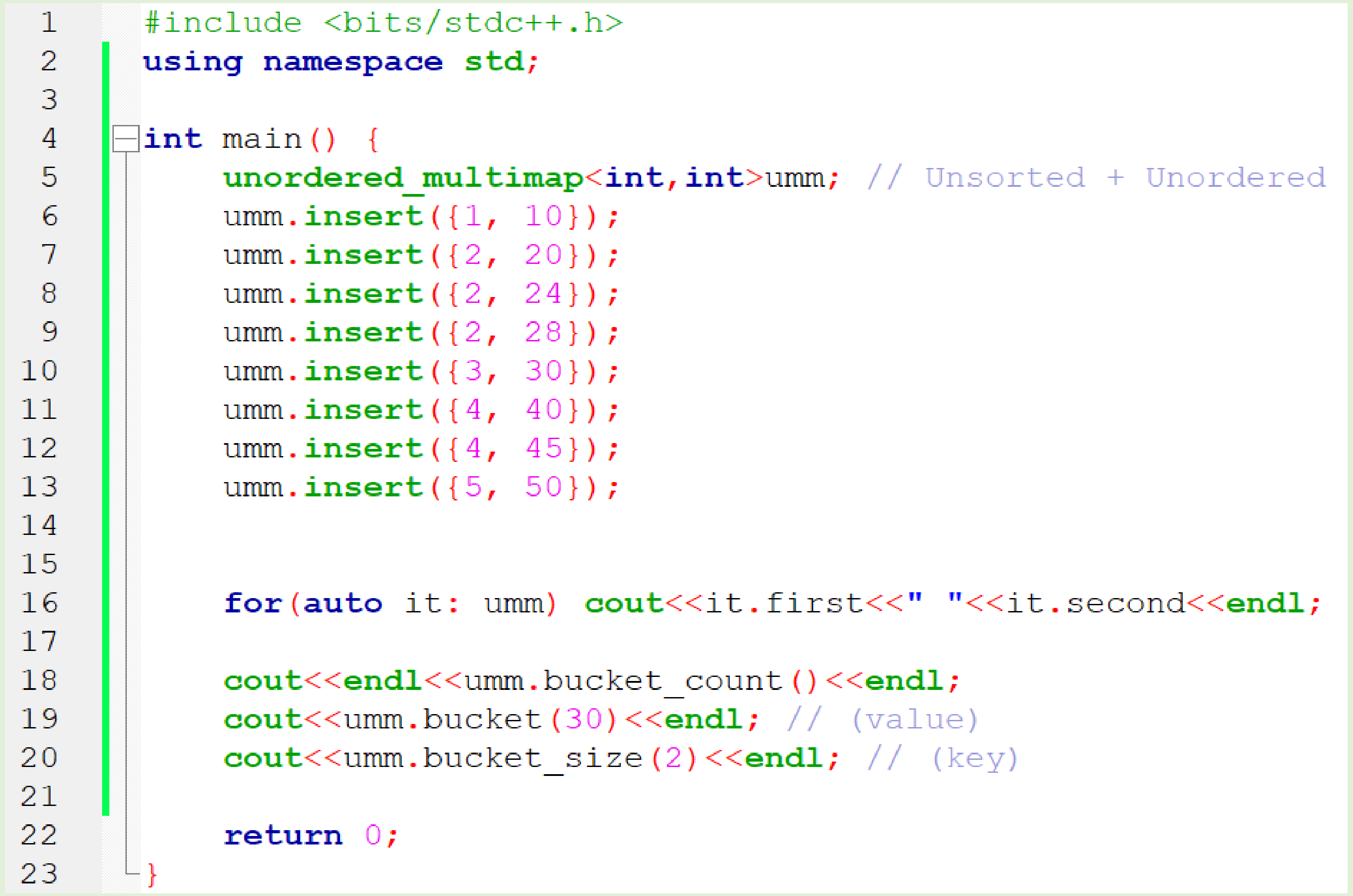


```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    unordered_map<int,int>ump; // Unsorted + Unordered
    ump.insert({1, 10});
    ump[2] = 20;
    ump[3] = 30;
    ump[4] = 40;
    ump[5] = 50;

    for(auto it: ump) cout<<it.first<<" "<<it.second<<endl;

    cout<<endl<<ump.bucket_count()<<endl;
    cout<<ump.bucket(30)<<endl; // (value)
    cout<<ump.bucket_size(2)<<endl; // (key)

    return 0;
}
```

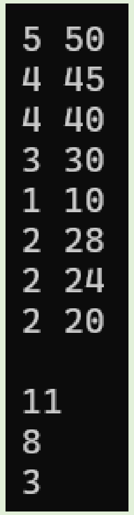


```
5 50
1 10
2 20
3 30
4 40

11
8
1
```

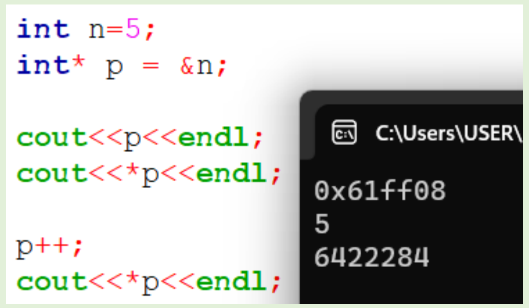
Unordered Multimap
(Include All Multimap fn)



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

int main() {
    unordered_multimap<int,int>umm; // Unsorted + Unordered
    umm.insert({1, 10});
    umm.insert({2, 20});
    umm.insert({2, 24});
    umm.insert({2, 28});
    umm.insert({3, 30});
    umm.insert({4, 40});
    umm.insert({4, 45});
    umm.insert({5, 50});


    for(auto it: umm) cout<<it.first<<" "<<it.second<<endl;

    cout<<endl<<umm.bucket_count()<<endl;
    cout<<umm.bucket(30)<<endl; // (value)
    cout<<umm.bucket_size(2)<<endl; // (key)

    return 0;
}
```

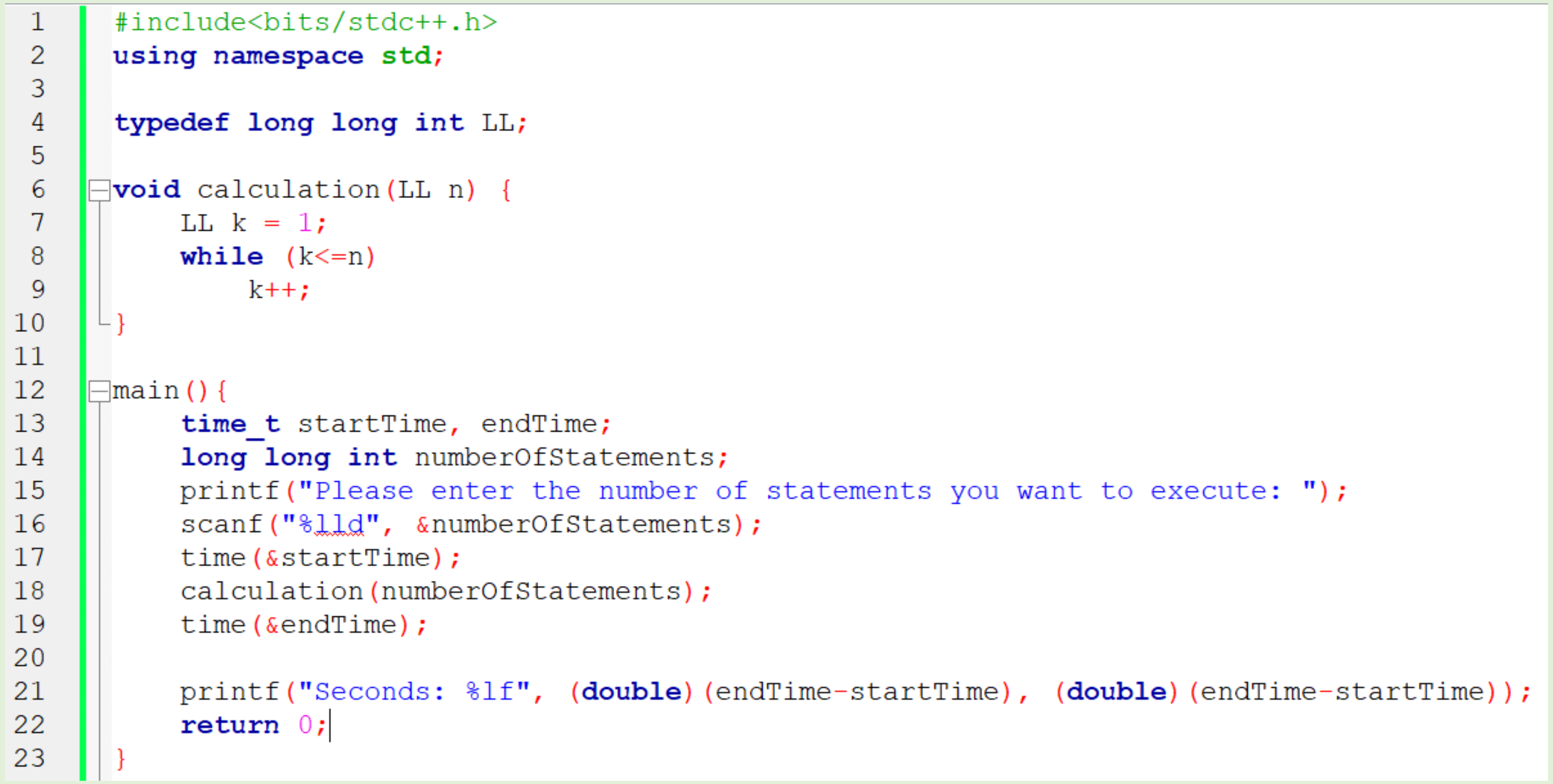


```
5 50
4 45
4 40
3 30
1 10
2 28
2 24
2 20

11
8
3
```

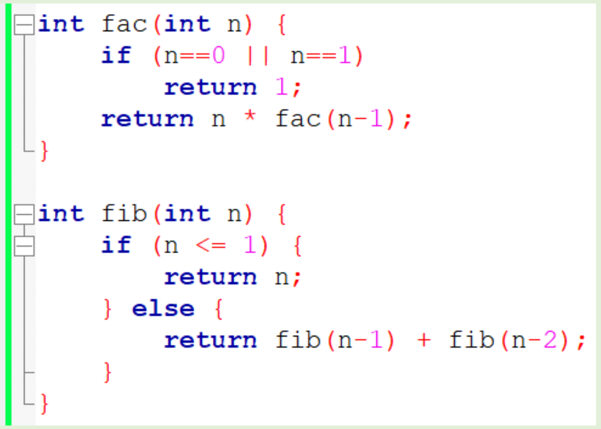


```cpp
int n=5;
int* p = &n;

cout<<p<<endl;
cout<<*p<<endl;

p++;
cout<<*p<<endl;
```

```
C:\Users\USER\
0x61ff08
5
6422284
```

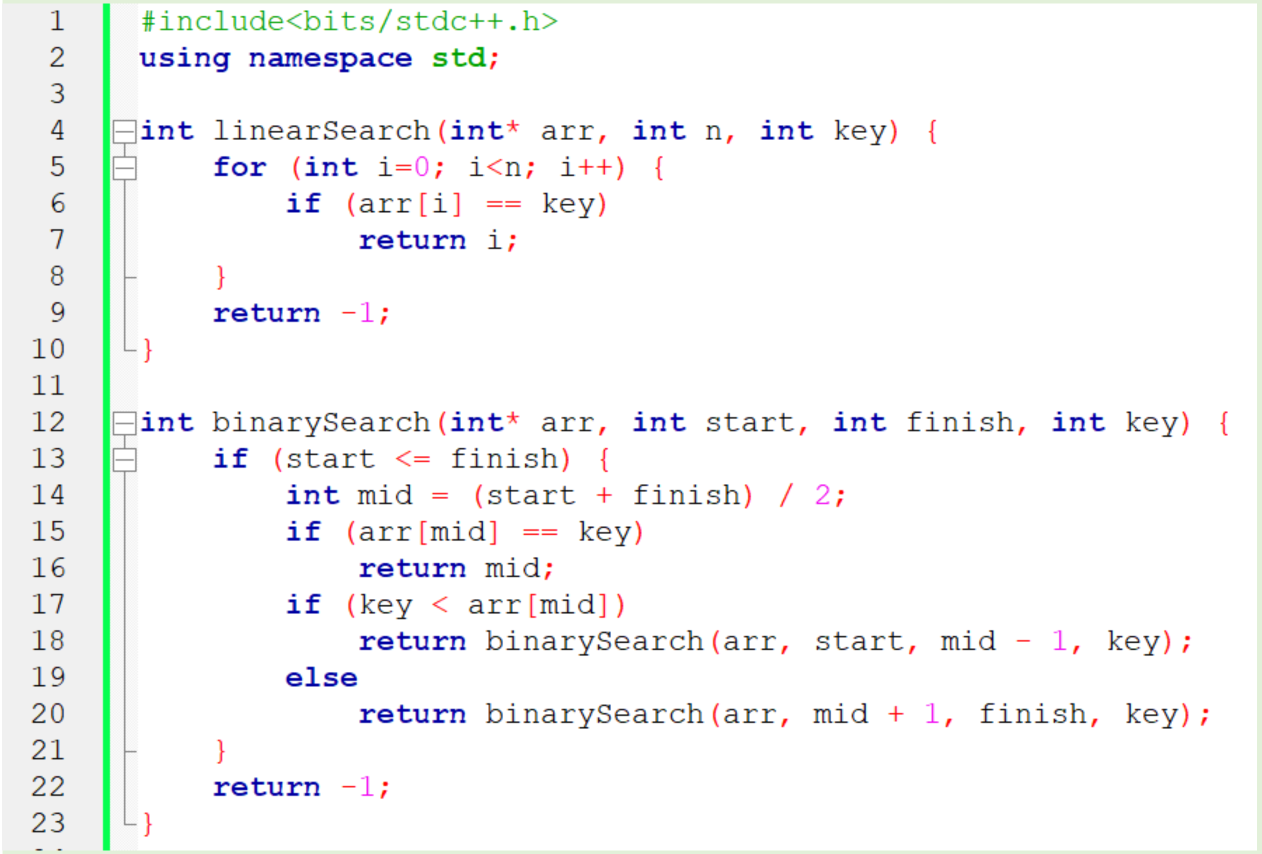


```cpp
#include<bits/stdc++.h>
using namespace std;

typedef long long int LL;

void calculation(LL n) {
    LL k = 1;
    while (k<=n)
        k++;
}

main(){
    time_t startTime, endTime;
    long long int numberOfStatements;
    printf("Please enter the number of statements you want to execute: ");
    scanf("%lld", &numberOfStatements);
    time(&startTime);
    calculation(numberOfStatements);
    time(&endTime);

    printf("Seconds: %lf", (double)(endTime-startTime), (double)(endTime-startTime));
    return 0;
}
```

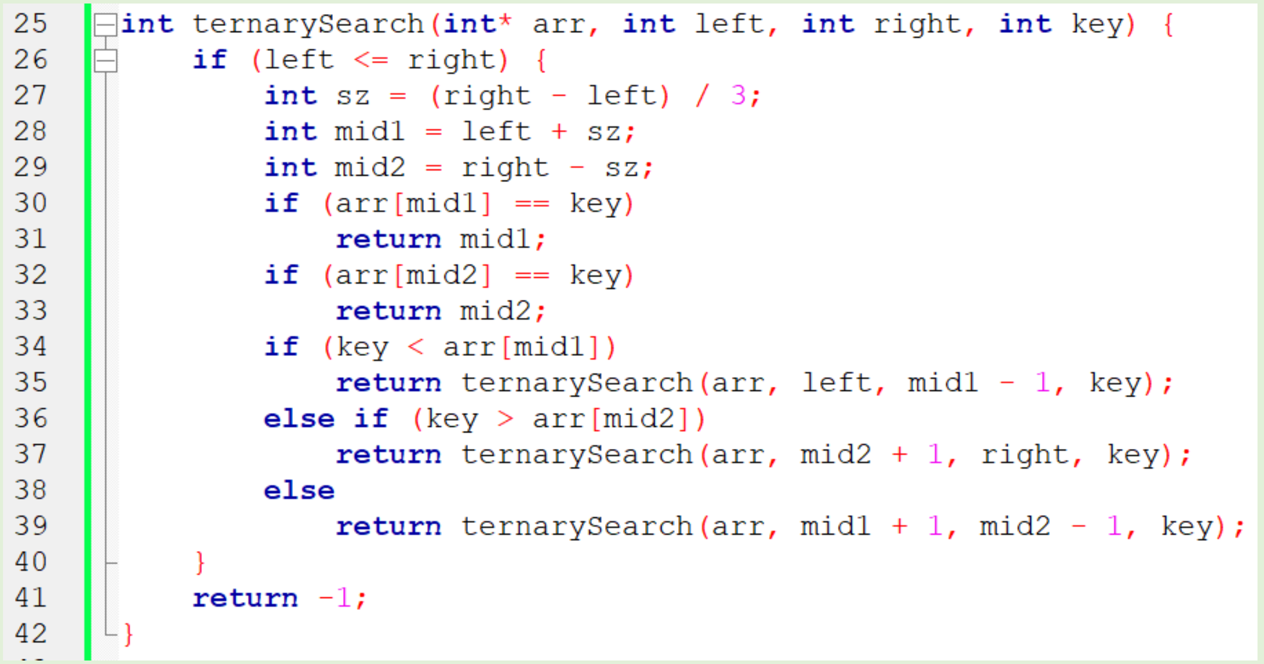


```cpp
int fac(int n) {
    if (n==0 || n==1)
        return 1;
    return n * fac(n-1);
}

int fib(int n) {
    if (n <= 1) {
        return n;
    } else {
        return fib(n-1) + fib(n-2);
    }
}
```

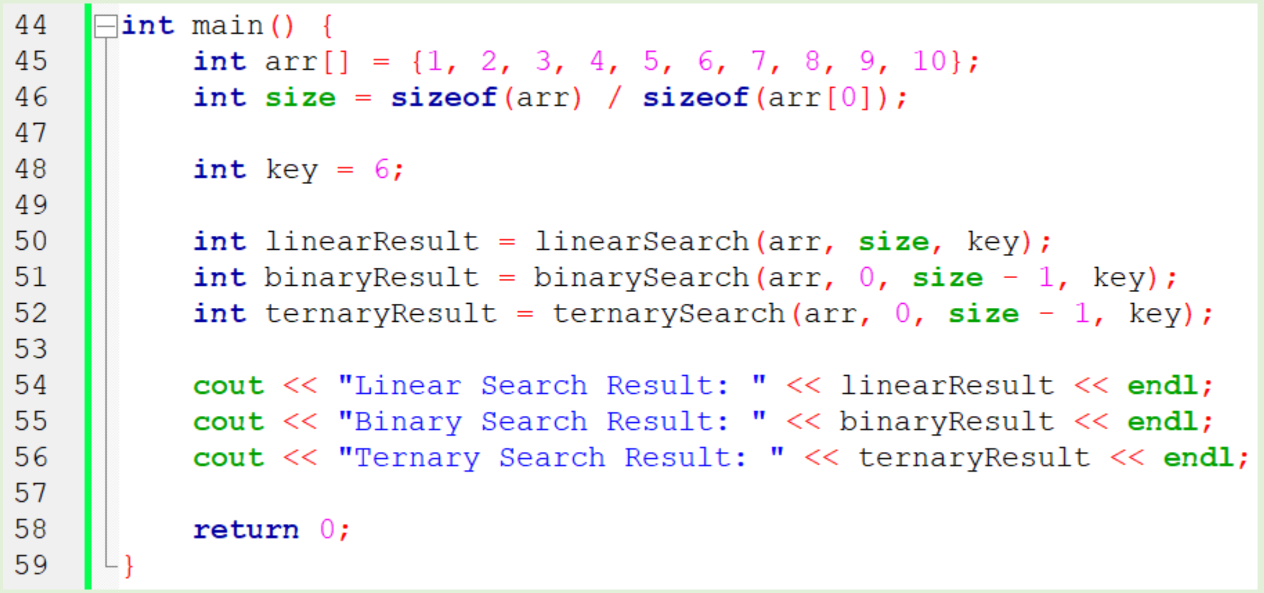
Searching Algorithm



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

int linearSearch(int* arr, int n, int key) {
    for (int i=0; i<n; i++) {
        if (arr[i] == key)
            return i;
    }
    return -1;
}

int binarySearch(int* arr, int start, int finish, int key) {
    if (start <= finish) {
        int mid = (start + finish) / 2;
        if (arr[mid] == key)
            return mid;
        if (key < arr[mid])
            return binarySearch(arr, start, mid - 1, key);
        else
            return binarySearch(arr, mid + 1, finish, key);
    }
    return -1;
}
```



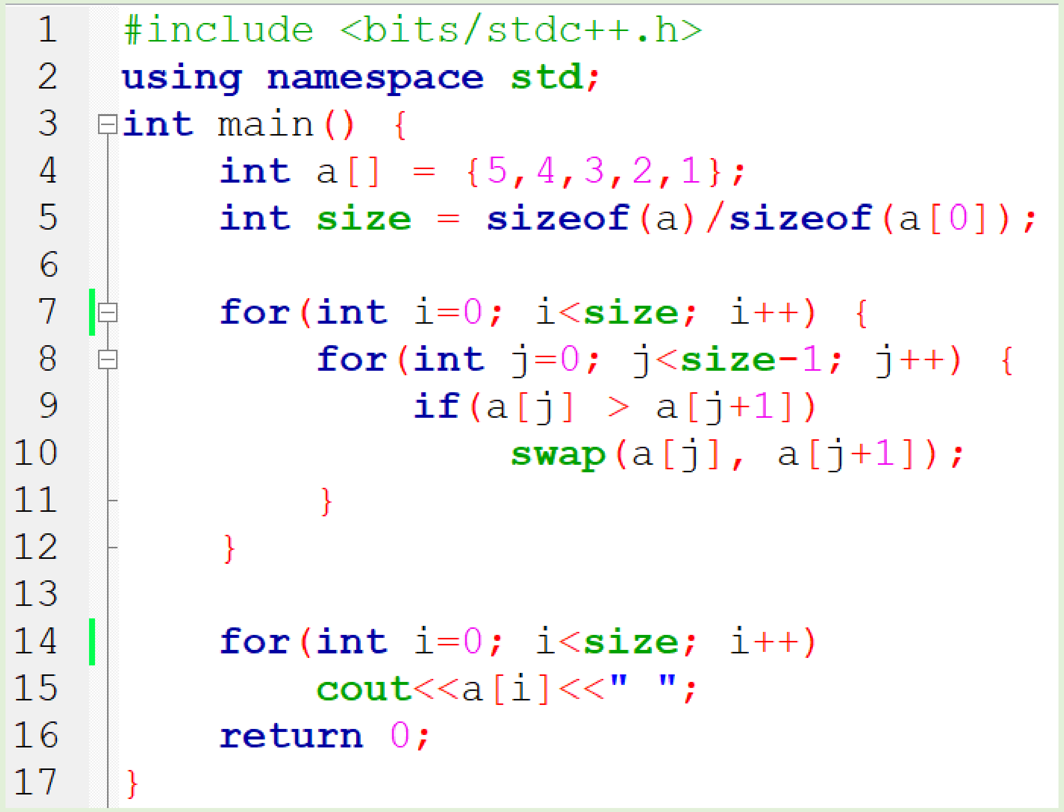
```cpp
int ternarySearch(int* arr, int left, int right, int key) {
    if (left <= right) {
        int sz = (right - left) / 3;
        int mid1 = left + sz;
        int mid2 = right - sz;
        if (arr[mid1] == key)
            return mid1;
        if (arr[mid2] == key)
            return mid2;
        if (key < arr[mid1])
            return ternarySearch(arr, left, mid1 - 1, key);
        else if (key > arr[mid2])
            return ternarySearch(arr, mid2 + 1, right, key);
        else
            return ternarySearch(arr, mid1 + 1, mid2 - 1, key);
    }
    return -1;
}
```



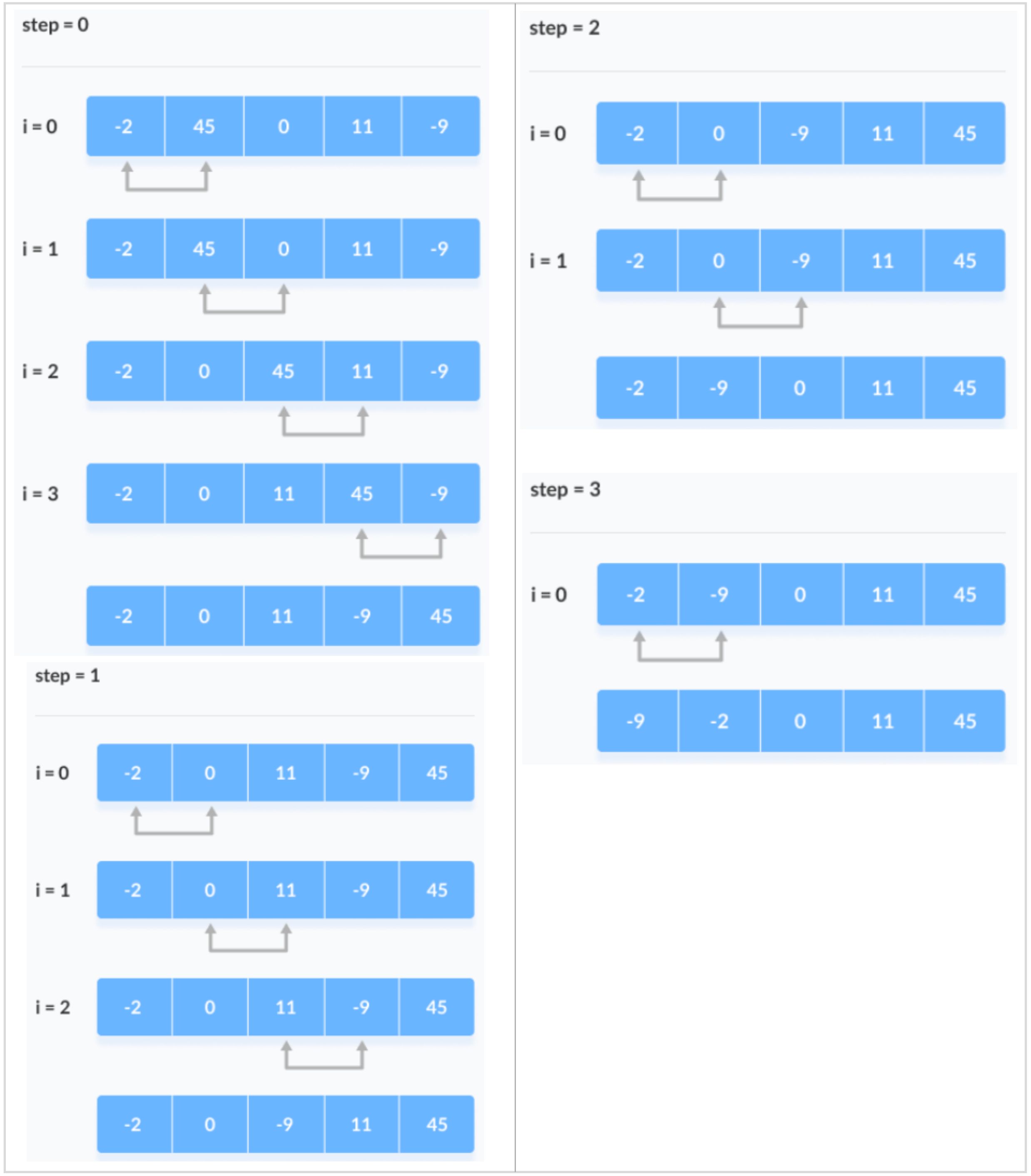
```cpp
int main() {
    int arr[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int size = sizeof(arr) / sizeof(arr[0]);

    int key = 6;

    int linearResult = linearSearch(arr, size, key);
    int binaryResult = binarySearch(arr, 0, size - 1, key);
    int ternaryResult = ternarySearch(arr, 0, size - 1, key);

    cout << "Linear Search Result: " << linearResult << endl;
    cout << "Binary Search Result: " << binaryResult << endl;
    cout << "Ternary Search Result: " << ternaryResult << endl;

    return 0;
}
```

Bubble Sort

Insertion Sort (programiz.com)

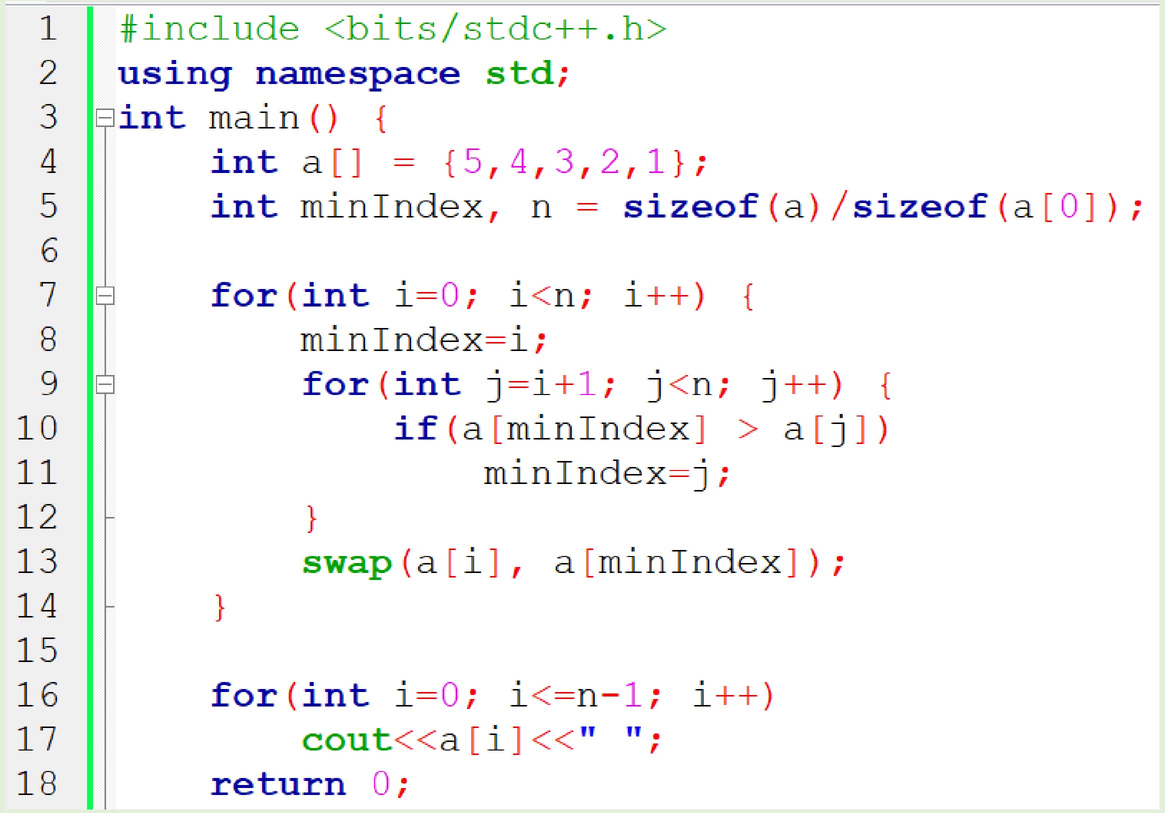


```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int a[] = {5,4,3,2,1};
    int size = sizeof(a)/sizeof(a[0]);

    for(int i=0; i<size; i++) {
        for(int j=0; j<size-1; j++) {
            if(a[j] > a[j+1])
                swap(a[j], a[j+1]);
        }
    }

    for(int i=0; i<size; i++)
        cout<<a[i]<<" ";
    return 0;
}
```

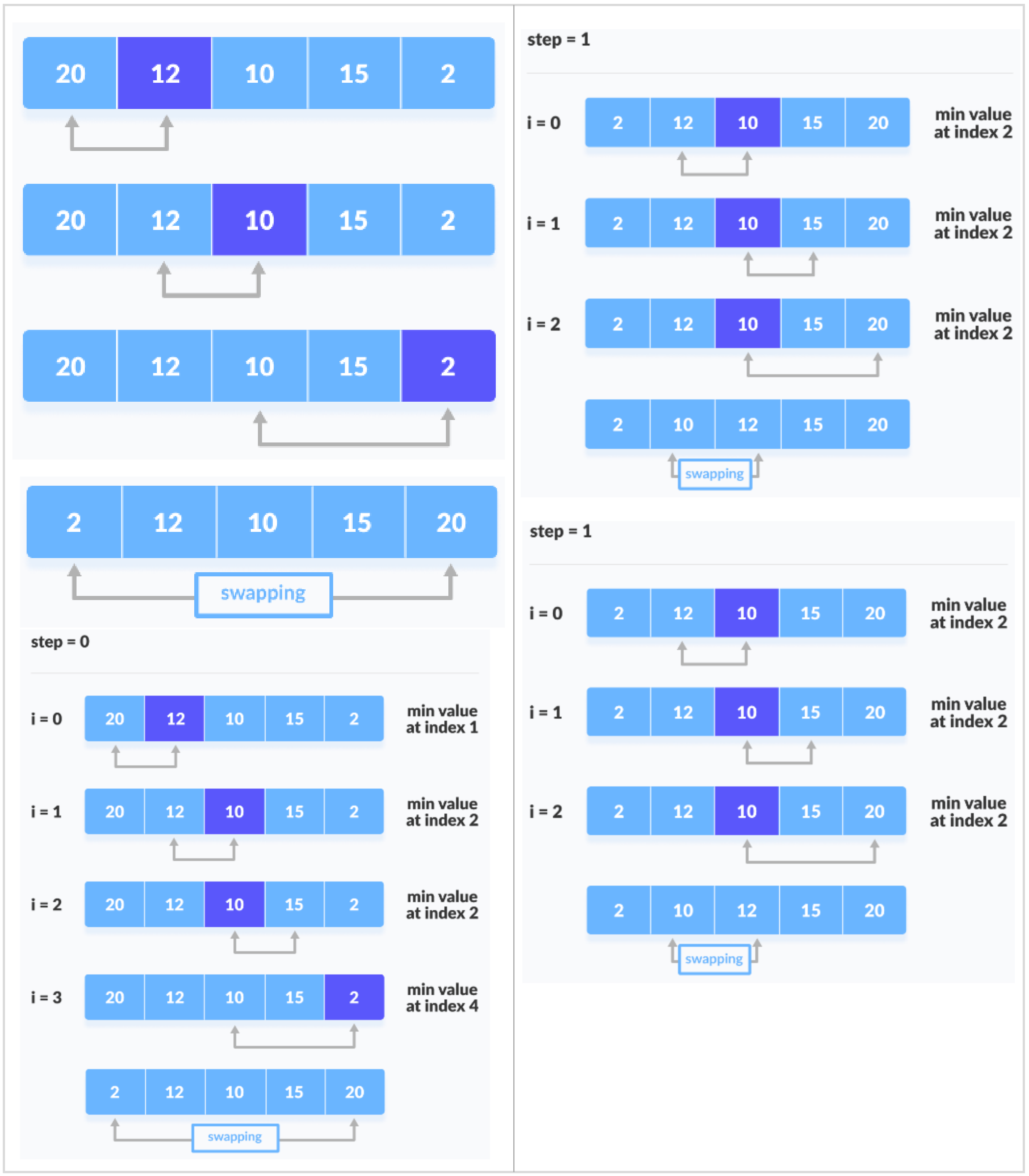


Selection Sort

Selection Sort (programiz.com)

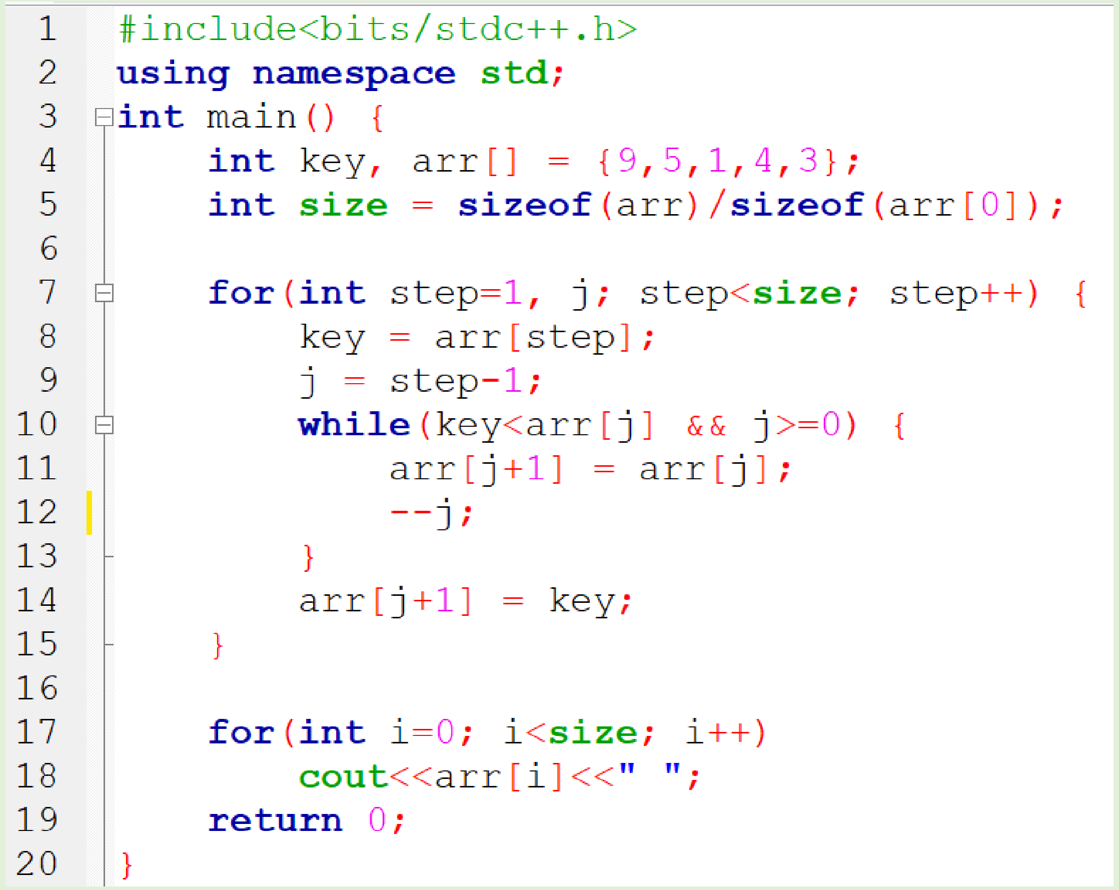


```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    int a[] = {5,4,3,2,1};
    int minIndex, n = sizeof(a)/sizeof(a[0]);

    for(int i=0; i<n; i++) {
        minIndex=i;
        for(int j=i+1; j<n; j++) {
            if(a[minIndex] > a[j])
                minIndex=j;
        }
        swap(a[i], a[minIndex]);
    }

    for(int i=0; i<=n-1; i++)
        cout<<a[i]<<" ";
    return 0;
```

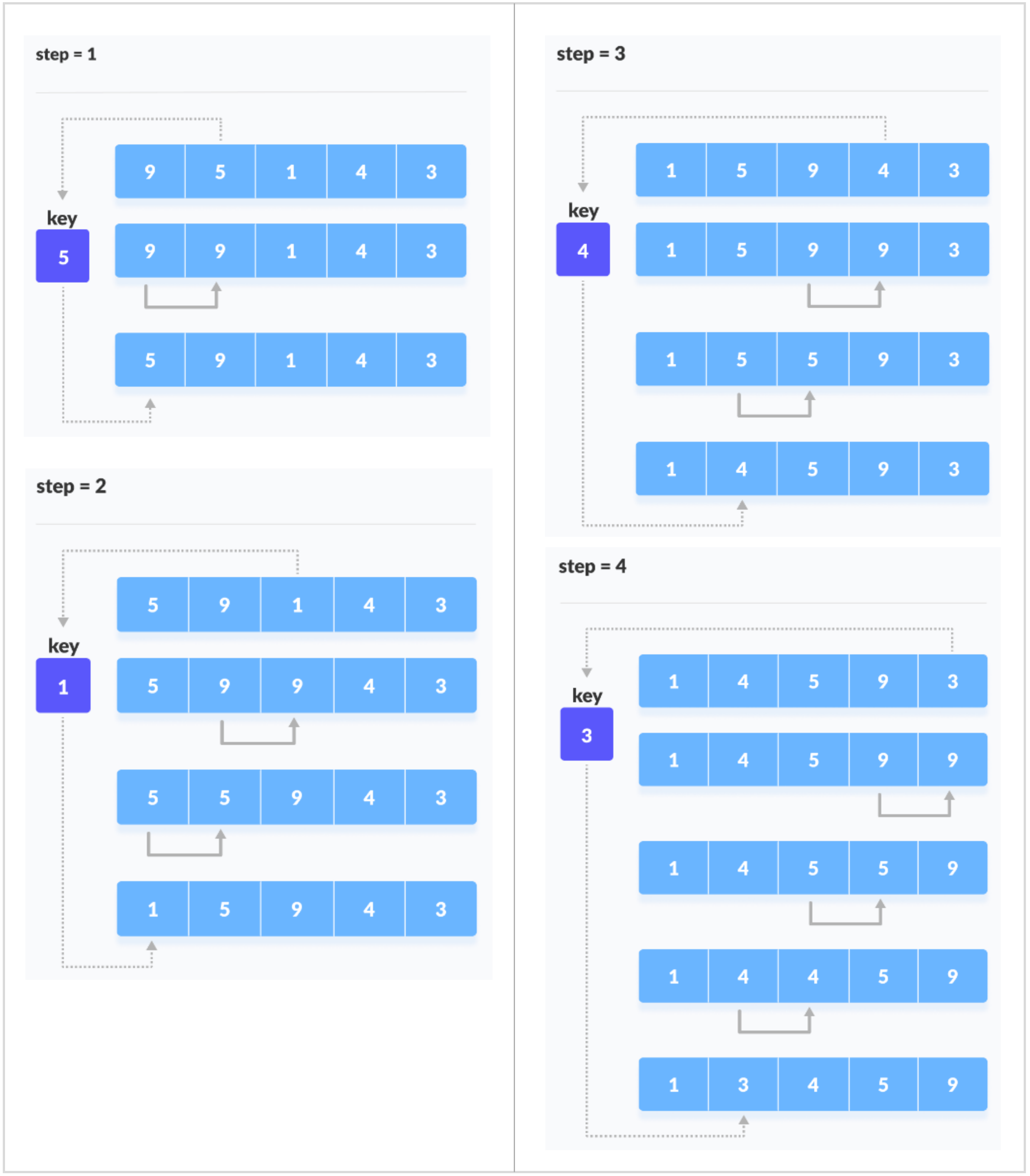


Insertion Sort

Insertion Sort (programiz.com)



```
#include<bits/stdc++.h>
using namespace std;
int main() {
    int key, arr[] = {9,5,1,4,3};
    int size = sizeof(arr)/sizeof(arr[0]);

    for(int step=1, j; step<size; step++) {
        key = arr[step];
        j = step-1;
        while(key<arr[j] && j>=0) {
            arr[j+1] = arr[j];
            --j;
        }
        arr[j+1] = key;
    }

    for(int i=0; i<size; i++)
        cout<<arr[i]<<" ";
    return 0;
}
```

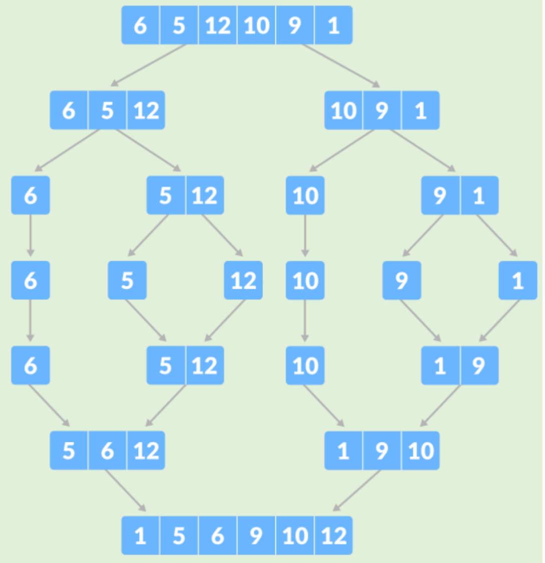


Merge Sort

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

void merge(int arr[], int start, int mid, int end){
    int n1 = (mid - start)+1;
    int n2 = end - mid;
    int L[n1], R[n2];

    for (int i = 0; i < n1; i++)
        L[i] = arr[start + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];

    int i=0, j=0, k=start;

    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
    while(i < n1) {
        arr[k] = L[i];
        i++; k++;
    }
    while(j < n2) {
        arr[k] = R[j];
        j++; k++;
    }
}
```

Merge Sort (programiz.com)

Merge Sort Algorithm – Bangla - Youtube

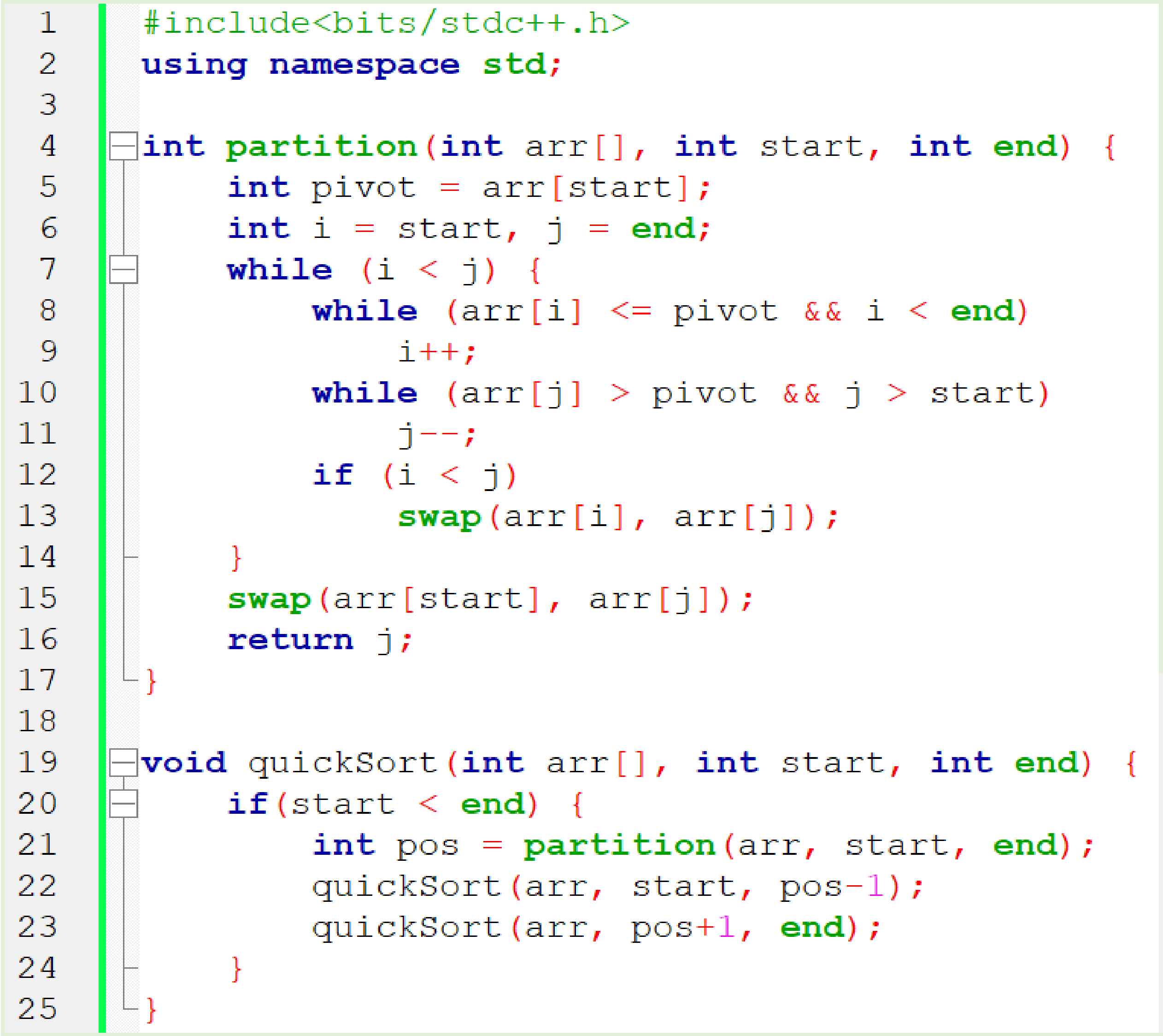


```
void mergeSort(int arr[], int start, int end) {
    if (start < end) {
        int mid = (start + end) / 2;
        mergeSort(arr, start, mid);
        mergeSort(arr, mid + 1, end);
        merge(arr, start, mid, end);
    }
}


int main() {
    int arr[] = {4,5,2,7,6,9,3};
    int size = sizeof(arr) / sizeof(arr[0]);

    mergeSort(arr, 0, size - 1);

    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

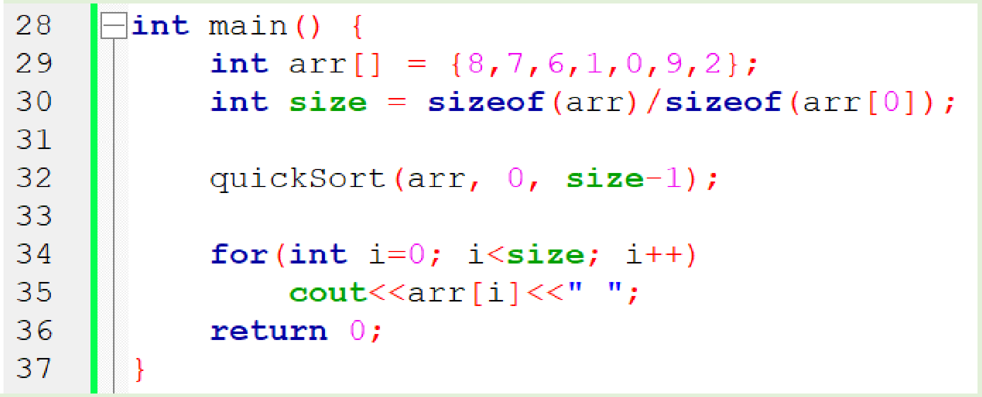
Quick Sort (Easy)

How Quick Sort Works – YouTube

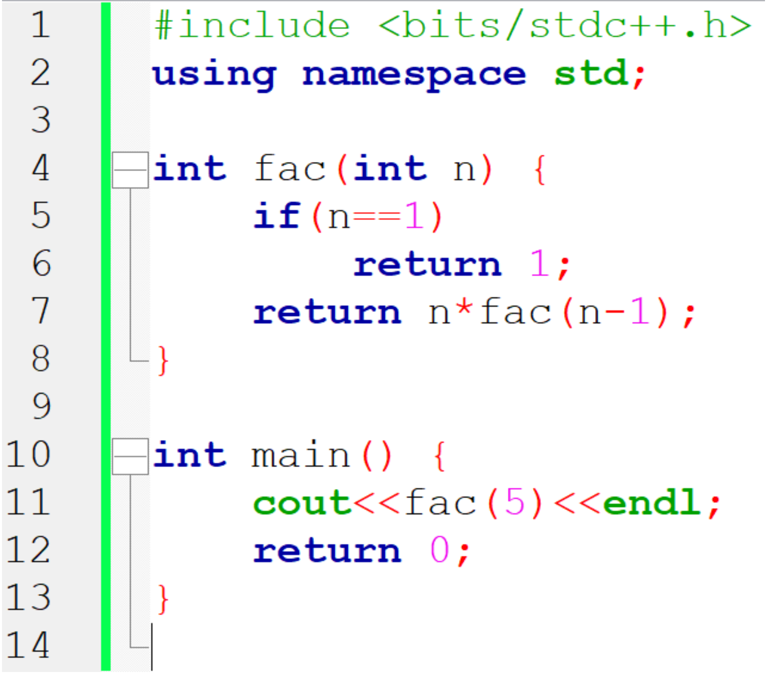
Quick Sort Algorithm | Bangla Tutorial - YouTube



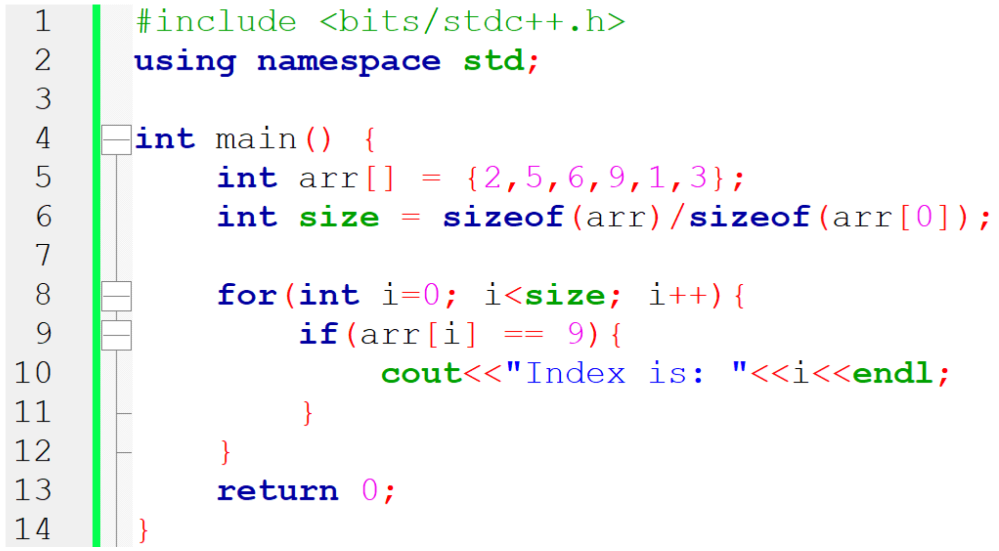
```cpp
#include<bits/stdc++.h>
using namespace std;

int partition(int arr[], int start, int end) {
    int pivot = arr[start];
    int i = start, j = end;
    while (i < j) {
        while (arr[i] <= pivot && i < end)
            i++;
        while (arr[j] > pivot && j > start)
            j--;
        if (i < j)
            swap(arr[i], arr[j]);
    }
    swap(arr[start], arr[j]);
    return j;
}

void quickSort(int arr[], int start, int end) {
    if(start < end) {
        int pos = partition(arr, start, end);
        quickSort(arr, start, pos-1);
        quickSort(arr, pos+1, end);
    }
}
```



```cpp
int main() {
    int arr[] = {8,7,6,1,0,9,2};
    int size = sizeof(arr)/sizeof(arr[0]);

    quickSort(arr, 0, size-1);

    for(int i=0; i<size; i++)
        cout<<arr[i]<<" ";
    return 0;
}
```



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

int fac(int n) {
    if(n==1)
        return 1;
    return n*fac(n-1);
}

int main() {
    cout<<fac(5)<<endl;
    return 0;
}

```



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

int main() {
    int arr[] = {2,5,6,9,1,3};
    int size = sizeof(arr)/sizeof(arr[0]);

    for(int i=0; i<size; i++){
        if(arr[i] == 9){
            cout<<"Index is: "<<i<<endl;
        }
    }
    return 0;
}
```