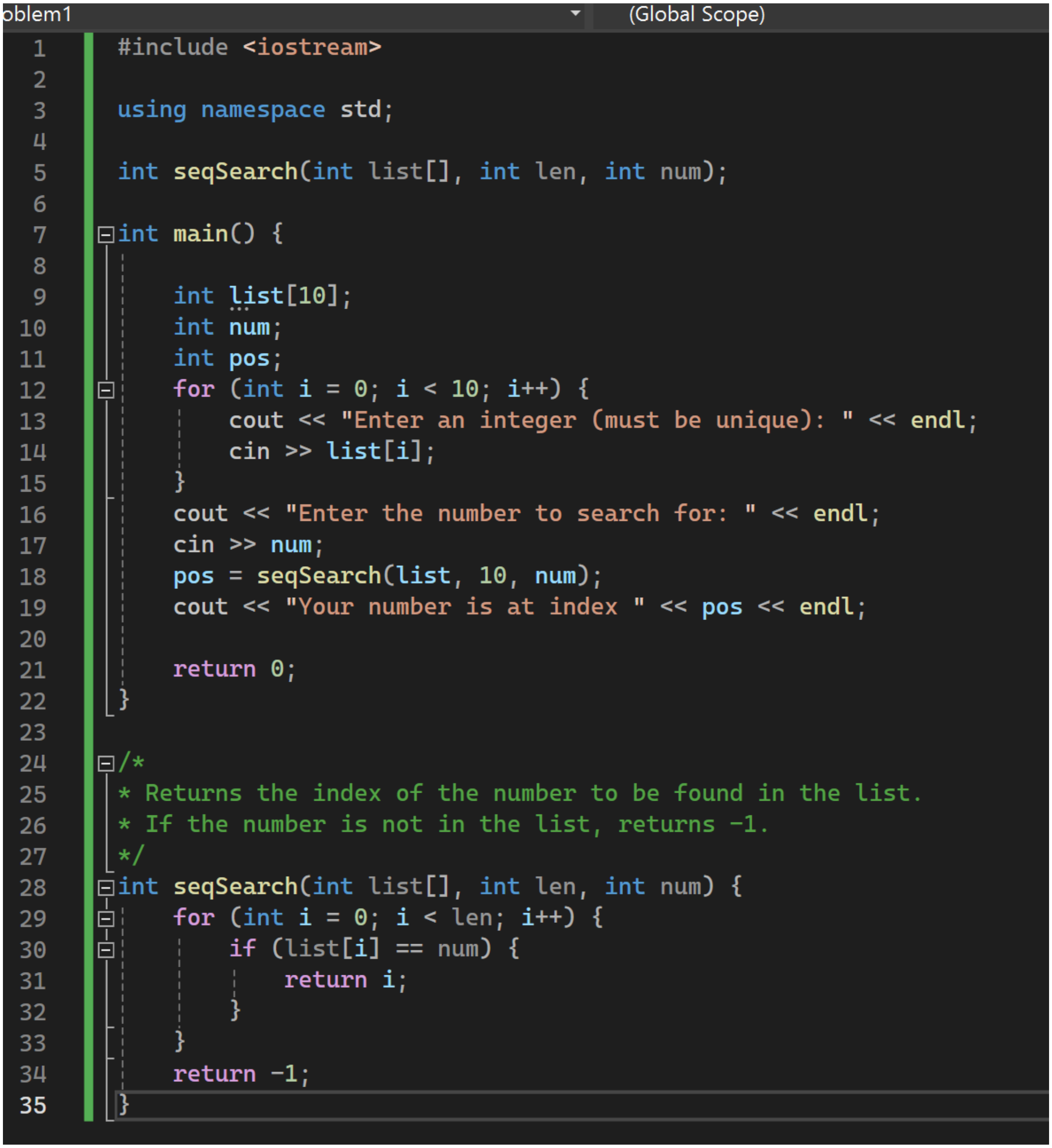
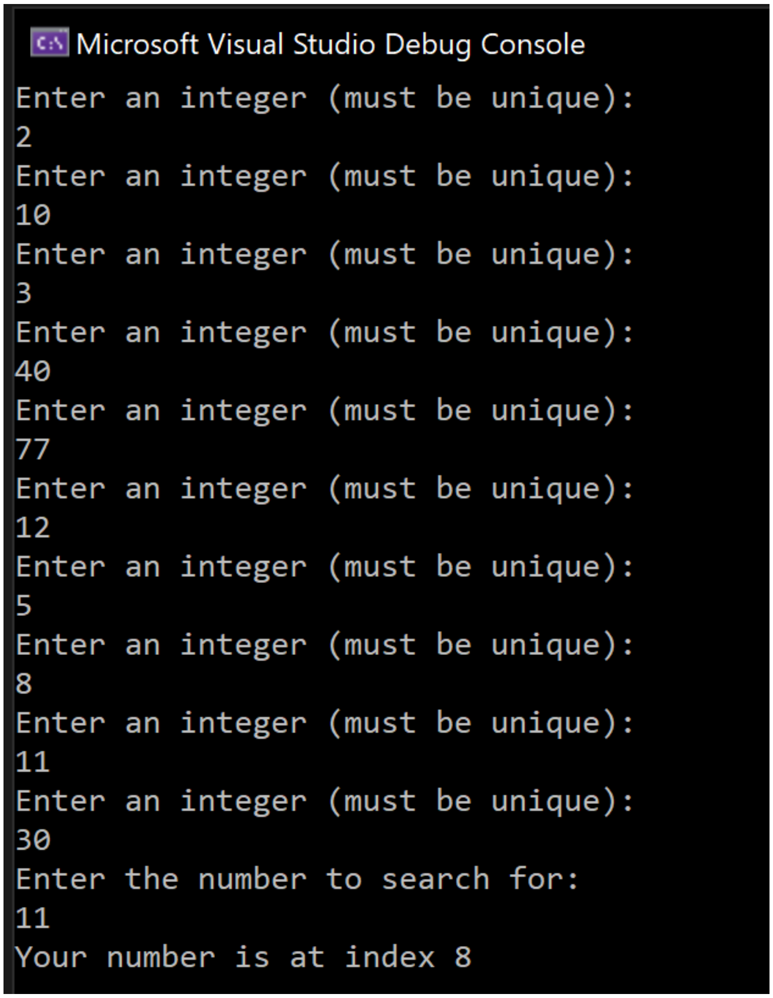
1. Linear search

Code:



```
#include <iostream>

using namespace std;

int seqSearch(int list[], int len, int num);

int main() {

    int list[10];
    int num;
    int pos;
    for (int i = 0; i < 10; i++) {
        cout << "Enter an integer (must be unique): " << endl;
        cin >> list[i];
    }
    cout << "Enter the number to search for: " << endl;
    cin >> num;
    pos = seqSearch(list, 10, num);
    cout << "Your number is at index " << pos << endl;

    return 0;
}

/*
* Returns the index of the number to be found in the list.
* If the number is not in the list, returns -1.
*/
int seqSearch(int list[], int len, int num) {
    for (int i = 0; i < len; i++) {
        if (list[i] == num) {
            return i;
        }
    }
    return -1;
}
```

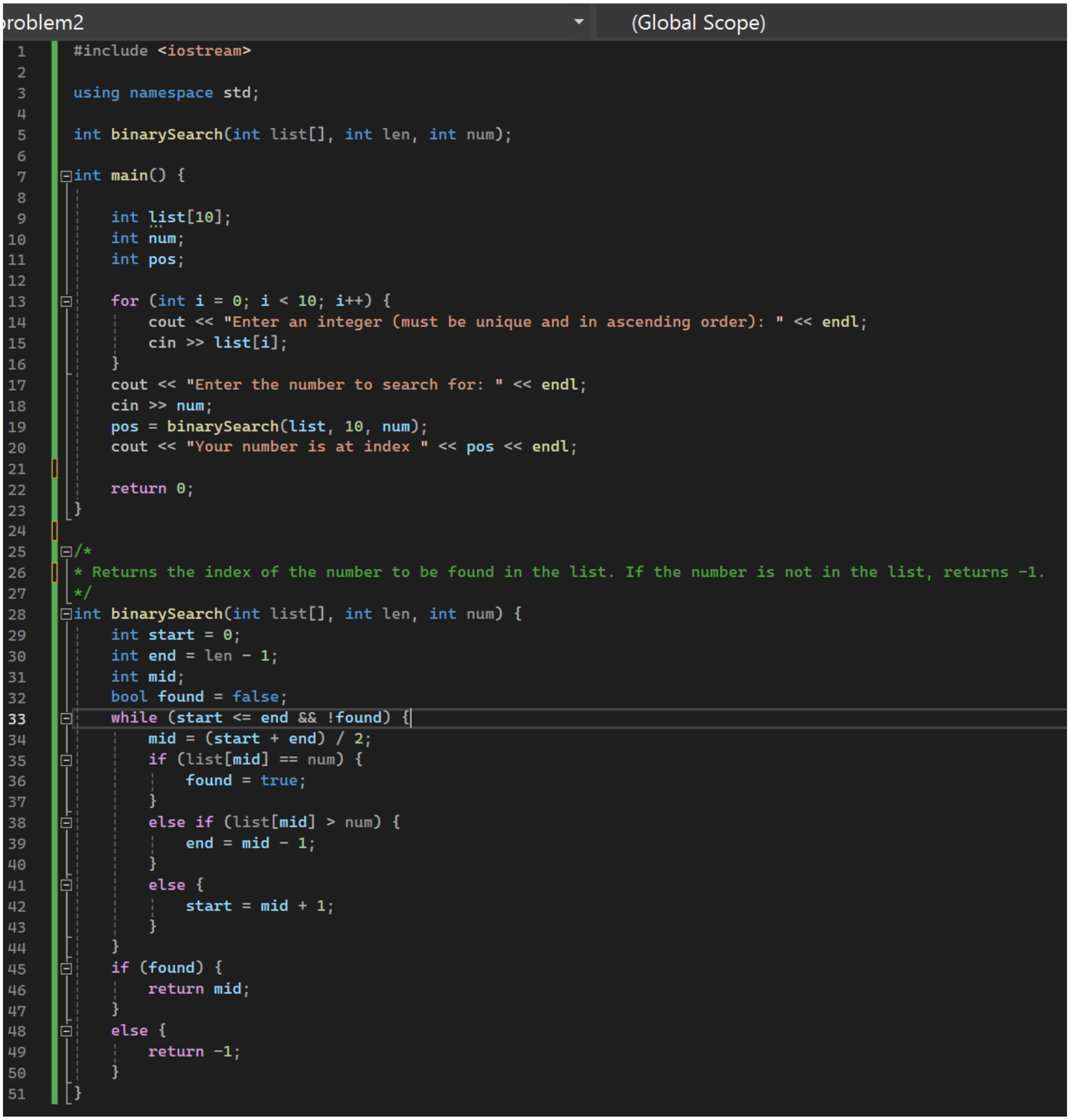
Output:



```
Enter an integer (must be unique):
2
Enter an integer (must be unique):
10
Enter an integer (must be unique):
3
Enter an integer (must be unique):
40
Enter an integer (must be unique):
77
Enter an integer (must be unique):
12
Enter an integer (must be unique):
5
Enter an integer (must be unique):
8
Enter an integer (must be unique):
11
Enter an integer (must be unique):
30
Enter the number to search for:
11
Your number is at index 8
```

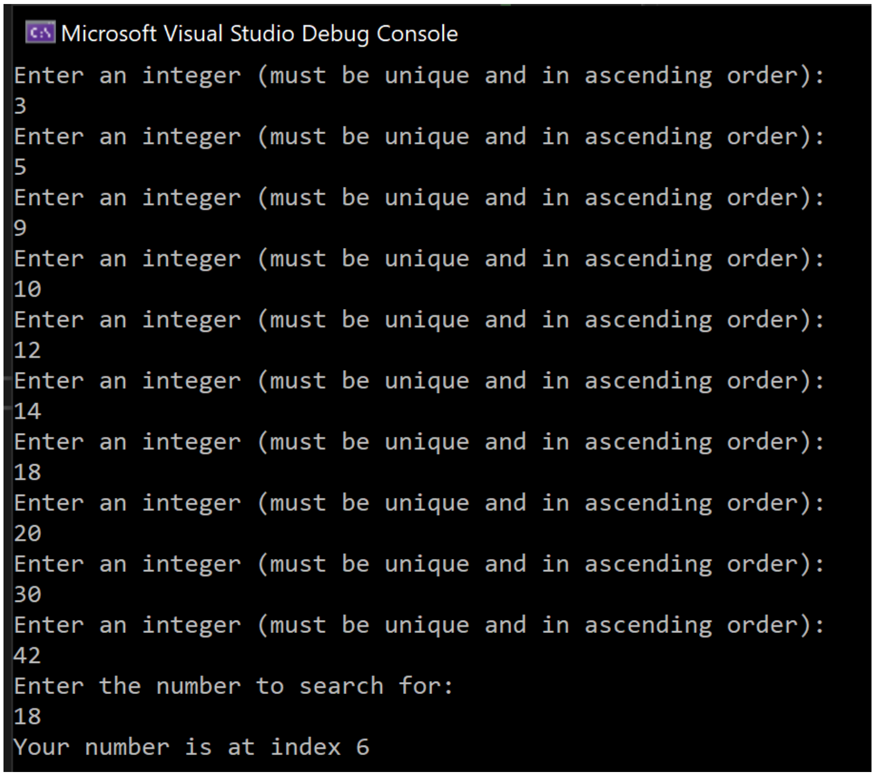
2. Binary search

Code:



```cpp
#include <iostream>

using namespace std;

int binarySearch(int list[], int len, int num);

int main() {

    int list[10];
    int num;
    int pos;

    for (int i = 0; i < 10; i++) {
        cout << "Enter an integer (must be unique and in ascending order): " << endl;
        cin >> list[i];
    }
    cout << "Enter the number to search for: " << endl;
    cin >> num;
    pos = binarySearch(list, 10, num);
    cout << "Your number is at index " << pos << endl;

    return 0;
}

/*
* Returns the index of the number to be found in the list. If the number is not in the list, returns -1.
*/
int binarySearch(int list[], int len, int num) {
    int start = 0;
    int end = len - 1;
    int mid;
    bool found = false;
    while (start <= end && !found) {
        mid = (start + end) / 2;
        if (list[mid] == num) {
            found = true;
        }
        else if (list[mid] > num) {
            end = mid - 1;
        }
        else {
            start = mid + 1;
        }
    }
    if (found) {
        return mid;
    }
    else {
        return -1;
    }
}
```

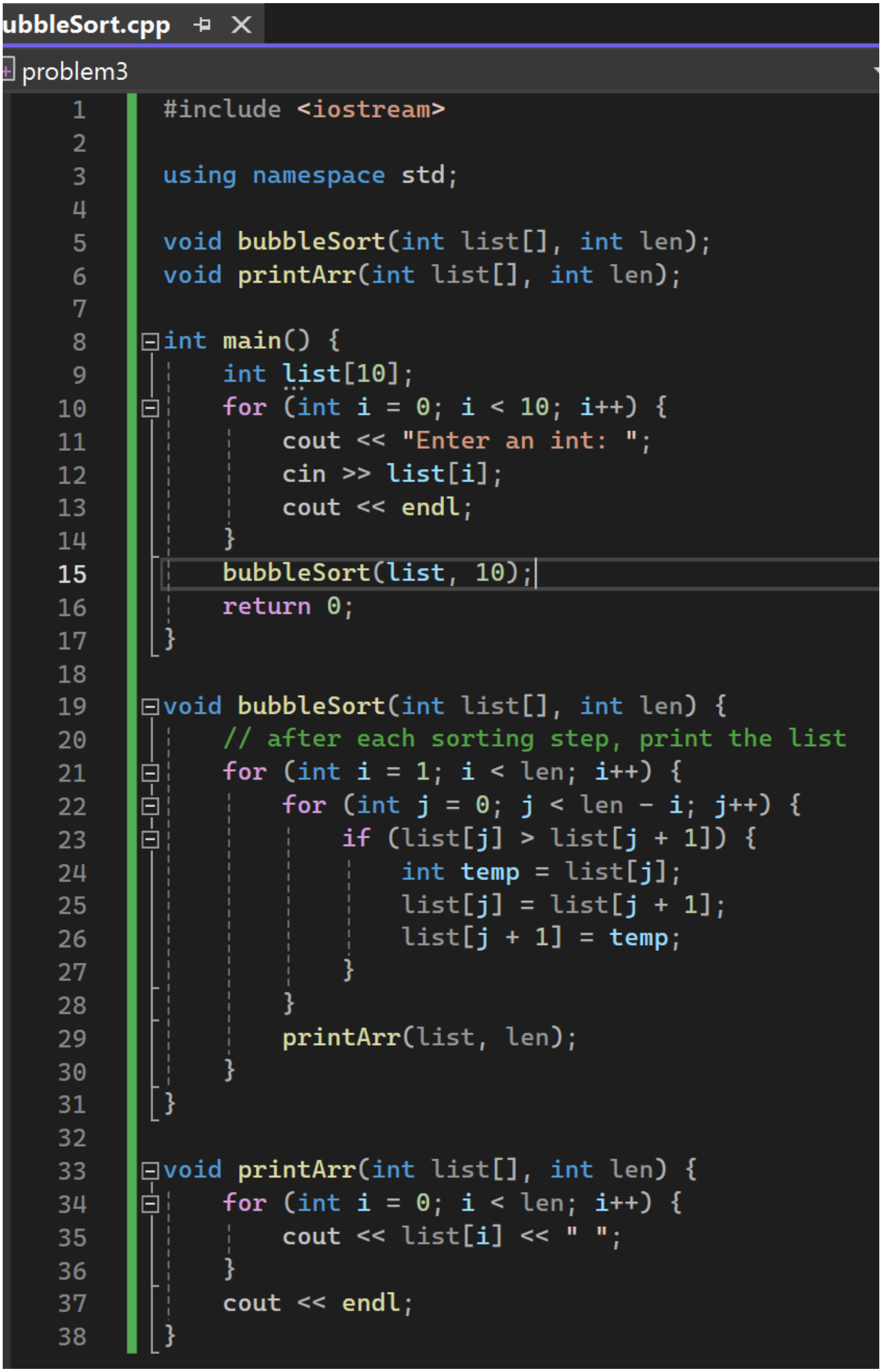
Output:



```
Enter an integer (must be unique and in ascending order):
3
Enter an integer (must be unique and in ascending order):
5
Enter an integer (must be unique and in ascending order):
9
Enter an integer (must be unique and in ascending order):
10
Enter an integer (must be unique and in ascending order):
12
Enter an integer (must be unique and in ascending order):
14
Enter an integer (must be unique and in ascending order):
18
Enter an integer (must be unique and in ascending order):
20
Enter an integer (must be unique and in ascending order):
30
Enter an integer (must be unique and in ascending order):
42
Enter the number to search for:
18
Your number is at index 6
```

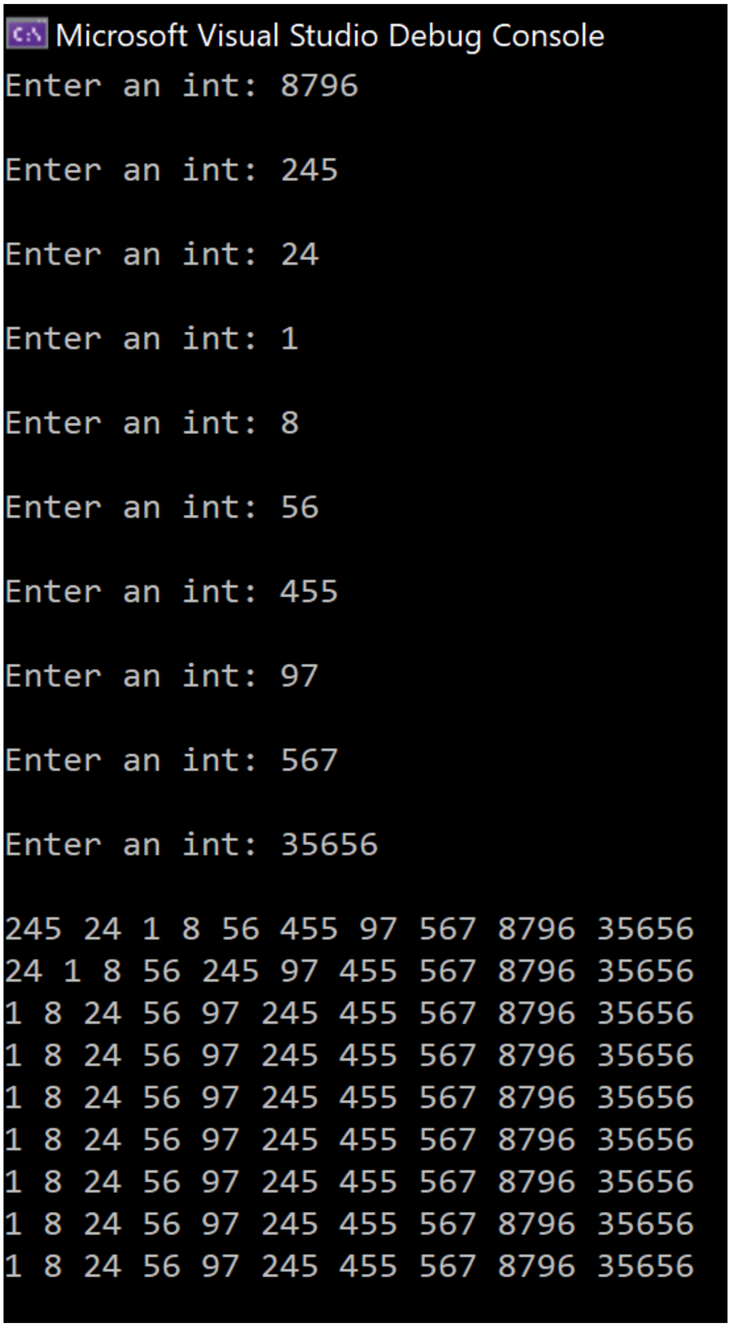
3. Bubble sort

Code:



```cpp
#include <iostream>

using namespace std;

void bubbleSort(int list[], int len);
void printArr(int list[], int len);

int main() {
    int list[10];
    for (int i = 0; i < 10; i++) {
        cout << "Enter an int: ";
        cin >> list[i];
        cout << endl;
    }
    bubbleSort(list, 10);
    return 0;
}

void bubbleSort(int list[], int len) {
    // after each sorting step, print the list
    for (int i = 1; i < len; i++) {
        for (int j = 0; j < len - i; j++) {
            if (list[j] > list[j + 1]) {
                int temp = list[j];
                list[j] = list[j + 1];
                list[j + 1] = temp;
            }
        }
        printArr(list, len);
    }
}

void printArr(int list[], int len) {
    for (int i = 0; i < len; i++) {
        cout << list[i] << " ";
    }
    cout << endl;
}
```

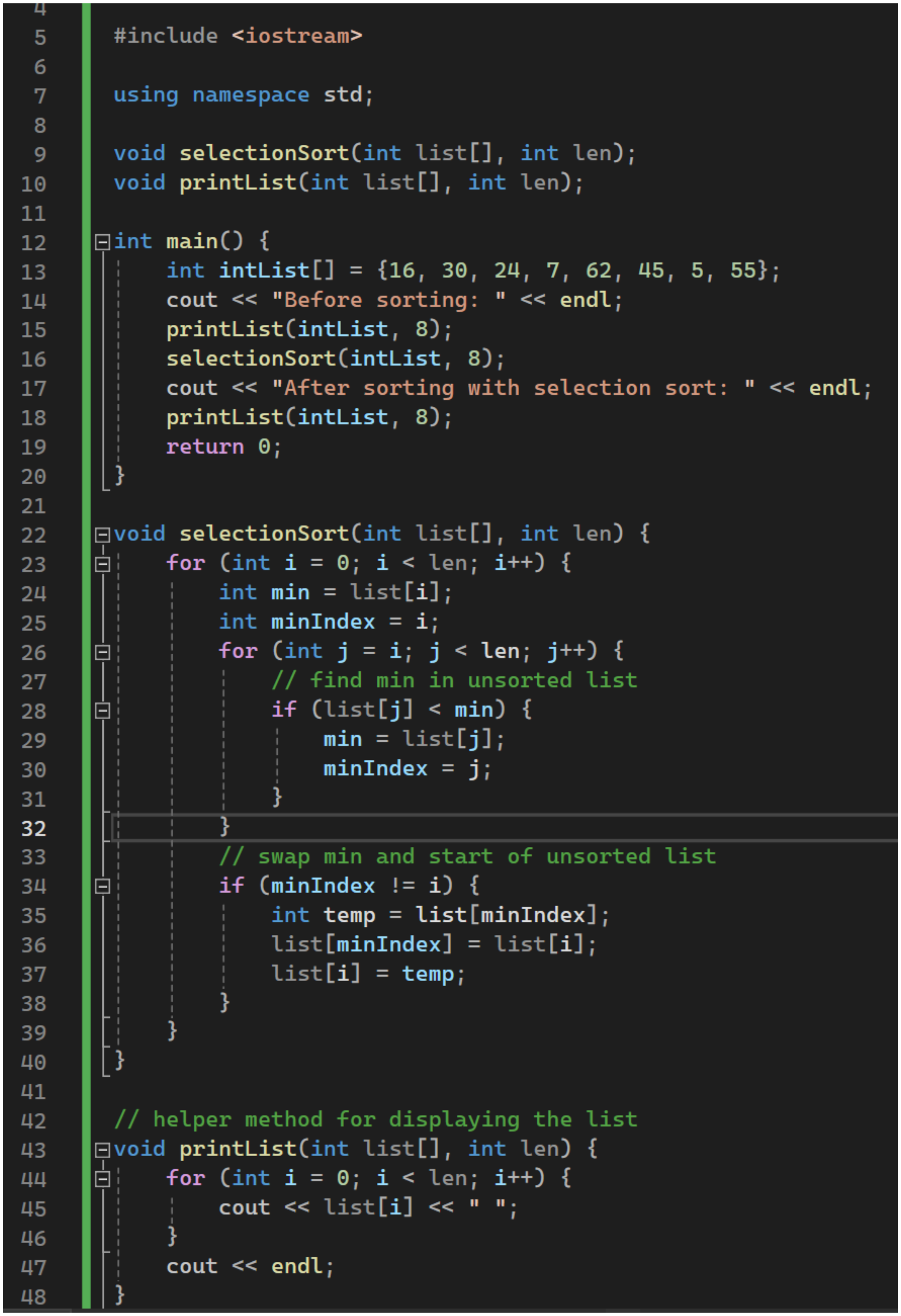
Output:



```
Enter an int: 8796

Enter an int: 245

Enter an int: 24

Enter an int: 1

Enter an int: 8

Enter an int: 56

Enter an int: 455

Enter an int: 97

Enter an int: 567

Enter an int: 35656

245 24 1 8 56 455 97 567 8796 35656
24 1 8 56 245 97 455 567 8796 35656
1 8 24 56 97 245 455 567 8796 35656
1 8 24 56 97 245 455 567 8796 35656
1 8 24 56 97 245 455 567 8796 35656
1 8 24 56 97 245 455 567 8796 35656
1 8 24 56 97 245 455 567 8796 35656
1 8 24 56 97 245 455 567 8796 35656
1 8 24 56 97 245 455 567 8796 35656
```

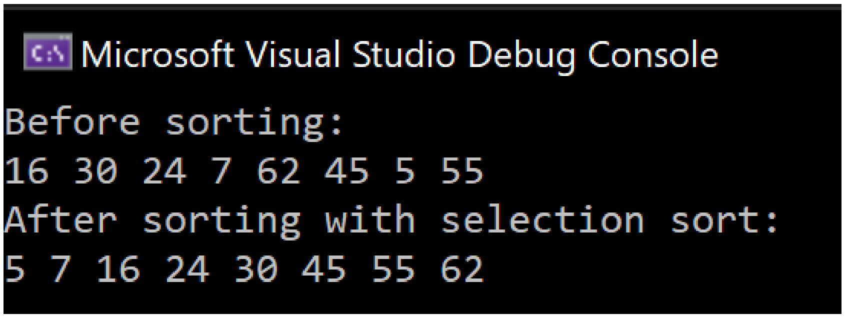
4. Selection sort

Code:



```cpp
#include <iostream>

using namespace std;

void selectionSort(int list[], int len);
void printList(int list[], int len);

int main() {
    int intList[] = {16, 30, 24, 7, 62, 45, 5, 55};
    cout << "Before sorting: " << endl;
    printList(intList, 8);
    selectionSort(intList, 8);
    cout << "After sorting with selection sort: " << endl;
    printList(intList, 8);
    return 0;
}

void selectionSort(int list[], int len) {
    for (int i = 0; i < len; i++) {
        int min = list[i];
        int minIndex = i;
        for (int j = i; j < len; j++) {
            // find min in unsorted list
            if (list[j] < min) {
                min = list[j];
                minIndex = j;
            }
        }
        // swap min and start of unsorted list
        if (minIndex != i) {
            int temp = list[minIndex];
            list[minIndex] = list[i];
            list[i] = temp;
        }
    }
}

// helper method for displaying the list
void printList(int list[], int len) {
    for (int i = 0; i < len; i++) {
        cout << list[i] << " ";
    }
    cout << endl;
}
```

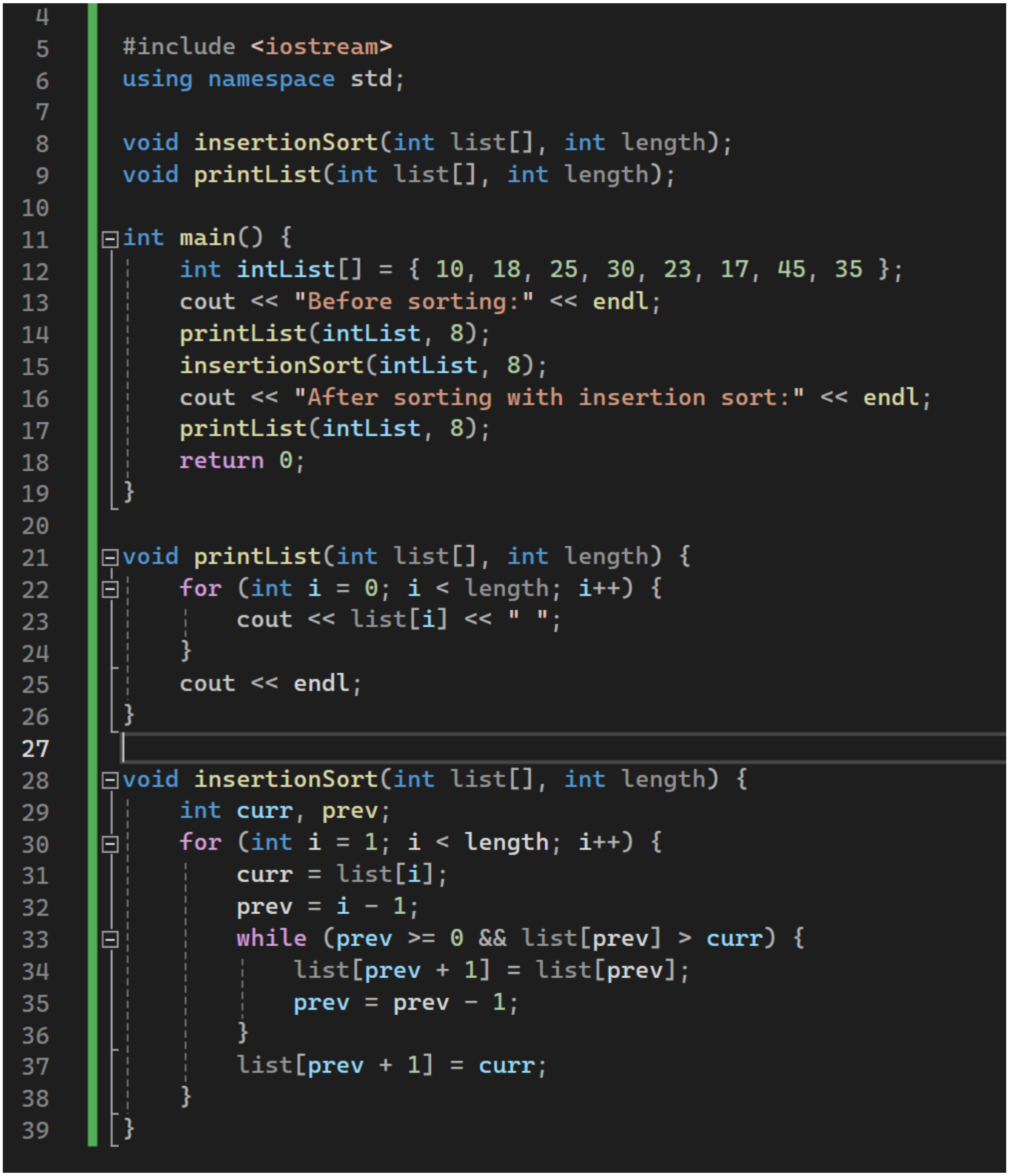
Output:



```
Before sorting:
16 30 24 7 62 45 5 55
After sorting with selection sort:
5 7 16 24 30 45 55 62
```

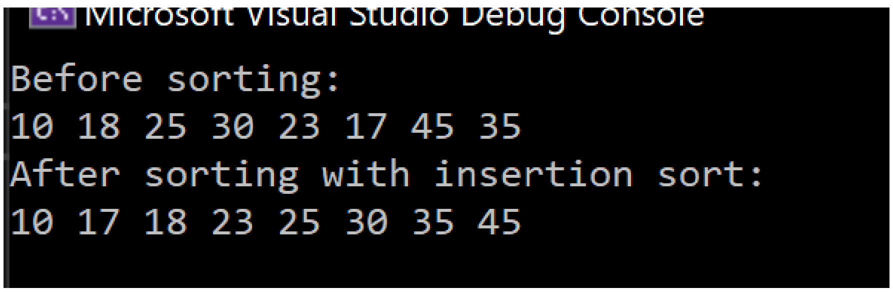
5. Insertion sort

Code:



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

void insertionSort(int list[], int length);
void printList(int list[], int length);

int main() {
    int intList[] = { 10, 18, 25, 30, 23, 17, 45, 35 };
    cout << "Before sorting:" << endl;
    printList(intList, 8);
    insertionSort(intList, 8);
    cout << "After sorting with insertion sort:" << endl;
    printList(intList, 8);
    return 0;
}

void printList(int list[], int length) {
    for (int i = 0; i < length; i++) {
        cout << list[i] << " ";
    }
    cout << endl;
}

void insertionSort(int list[], int length) {
    int curr, prev;
    for (int i = 1; i < length; i++) {
        curr = list[i];
        prev = i - 1;
        while (prev >= 0 && list[prev] > curr) {
            list[prev + 1] = list[prev];
            prev = prev - 1;
        }
        list[prev + 1] = curr;
    }
}
```

Output:



```
Before sorting:
10 18 25 30 23 17 45 35
After sorting with insertion sort:
10 17 18 23 25 30 35 45
```