Lesson 2 C++ Activity Summary

Contents

Set Usage with Integer and String Collections **2**

Searching in a Set with .find() **2**

Unordered Set with Find Operation **3**

Vector Manipulations and Capacity Tracking **4**

Set Usage with Integer and String Collections

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

// Method to print numbers
template<typename T>
void printNumbers(const set<T>& elements,string command="Not provided") {
    if (command !=  "Not provided"){
        cout << "After Command: " << command << endl;
    }
    cout << "Set Contents" << endl;
    for (T elems : elements) {
        cout << elems << " ";
    }
    cout << endl;
}

int main(){
    set<int> numbers = {1,1,2,2,3,4,5,5,5};printNumbers(numbers);
    set<string> names = {"Joe","Karen","Lisa","Jackie"};printNumbers(names);

    return 0;
}
```

Searching in a Set with .find()

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

// Method to print numbers
template<typename T>
void printNumbers(const set<T>& elements,string command="Not provided") {
    if (command !=  "Not provided"){
        cout << "After Command: " << command << endl;
    }
    cout << "Set Contents" << endl;
    for (T elems : elements) {
        cout << elems << " ";
    }
    cout << endl;
}

int main(){
```

```
    set<string> names = {"Joe","Karen","Lisa","Jackie"};printNumbers(names);

    set<string>::iterator iter;

    // Find "Karen"
    iter = names.find("Karen");

    // Display the results
    if (iter != names.end()){
        cout << *iter << " was found.\n";
    }else{
        cout << "Karen was NOT found.\n";
    }

    return 0;
}
```

Unordered Set with Find Operation

```
#include <iostream>
#include <vector>
#include <unordered_set> // NOTE: using unordered_set
using namespace std;

// Method to print numbers
template<typename T>
void printNumbers(const unordered_set<T>& elements, string command = "Not provided") {
    if (command != "Not provided") {
        cout << "After Command: " << command << endl;
    }
    cout << "Unordered Set Contents" << endl;
    for (T elems : elements) {
        cout << elems << " ";
    }
    cout << endl;
}

int main() {
    unordered_set<string> names = {"Joe", "Karen", "Lisa", "Jackie"};
    printNumbers(names);

    unordered_set<string>::iterator iter;

    // Find "Karen"
    iter = names.find("Karen");

    // Display the results
    if (iter != names.end()) {
```

```
        cout << *iter << " was found.\n";
    } else {
        cout << "Karen was NOT found.\n";
    }

    return 0;
}
```

Vector Manipulations and Capacity Tracking

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

// Method to print numbers
void printNumbers(const vector<int>& numbers,string command="Not provided") {
    if (command !=  "Not provided"){
        cout << "After Command: " << command << endl;
    }
    cout << "Vector Contents" << endl;
    for (int num : numbers) {
        cout << num << " ";
    }
    cout << endl;
}

int main(){
    vector<int> vec;printNumbers(vec,"vector<int> vec");
    vec.push_back(10);printNumbers(vec,"vec.push_back(10)");
    vec.push_back(30);printNumbers(vec,"vec.push_back(30)");
    vec.pop_back();printNumbers(vec,"vec.pop_back()");
    cout << "Size: " << vec.size() << endl;
    cout << "Capacity: " << vec.capacity() << endl;printNumbers(vec,"vec.capacity()")
    vec.push_back(90);printNumbers(vec,"vec.push_back(90)");
    vec.push_back(100);printNumbers(vec,"vec.push_back(100)");
    vec.pop_back();printNumbers(vec,"vec.pop_back()");
    cout << "Size: " << vec.size() << endl;
    cout << "Capacity: " << vec.capacity() << endl;printNumbers(vec);
    vec.pop_back();printNumbers(vec,"vec.pop_back()");
    cout << "Size: " << vec.size() << endl;
    cout << "Capacity: " << vec.capacity() << endl;printNumbers(vec),"vec.size() and
    cout << "Is empty?: " << (vec.empty() ? "Yes":"No") << endl;printNumbers(vec,"vec
    vec.pop_back();printNumbers(vec,"vec.pop_back()");
    cout << "Is empty?: " << (vec.empty() ? "Yes":"No") << endl;printNumbers(vec,"vec
    vec.clear();printNumbers(vec);printNumbers(vec,"vec.clear()");
    cout << "Size after clear: " << vec.size() << endl;
    vec.push_back(10);printNumbers(vec,"vec.push_back(10)");
```

```
    vec.push_back(30);printNumbers(vec,"vec.push_back(30)");
    vec.insert(vec.begin()+1,20);printNumbers(vec,"vec.insert(vec.begin()+1,20)"); //
    vec.erase(vec.begin());printNumbers(vec,"vec.erase(vec.begin())"); // Removes the
    vec.push_back(40);printNumbers(vec,"vec.push_back(40)");
    vec.push_back(50);printNumbers(vec,"vec.push_back(50)");
    vec.erase(vec.begin(),vec.begin()+2);printNumbers(vec,"vec.erase(vec.begin(),vec.
    vec.reserve(10);// Reserves capacity for 10 elements (does not change size)
    cout << "Size: " << vec.size() << endl;
    cout << "Capacity: " << vec.capacity() << endl;
    vec.shrink_to_fit();
    cout << "Size: " << vec.size() << endl;
    cout << "Capacity: " << vec.capacity() << endl;
    vec.insert(vec.begin(),10);printNumbers(vec,"vec.insert(vec.begin(),10)");
    vec.insert(vec.end()-1,90);printNumbers(vec,"vec.insert(vec.end()-1,90)");
    return 0;
}
```