Lesson 3 C++ STL Algorithm Summary

Contents

Binary Search with std::binary$_search$

```
#include <iostream>
#include <vector>
#include <array>
#include <algorithm> // for std::for_each
using namespace std;

void myPrint(int n){
    cout << n << " ";
}

// Custom comparator function (Descending Order)
bool customCompare(int a,int b){
    return a>b;
}

int main(){
    array<int,16> s{15,5,7,10,13,4,2,8,11,6,1,9,12,0,3,14};
    cout << "sorted with the default operator <: std::less\n";
    sort(s.begin(),s.end());
    for_each(s.begin(),s.end(),myPrint);cout << endl;
    auto result = binary_search(s.begin(),s.end(),4);
    cout << "Type of result: " << typeid(result).name() << endl;
}
```

Conditional Counting with std::count$_if$

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

bool is_Odd(int &n)
{
    return n%2 != 0;
}

int main(){
    vector<int> numbers = {1,2,3,4,5,6,7,8,9,10,11};

    // use foreach to apply doubleNumber to each elemnt in the vector
    int oddCount = count_if(numbers.begin(),numbers.end(),is_Odd);

    // Print the count of odd number
    cout << "There are " << oddCount << " odd numbers in the vector";
    cout << endl;
```

```
    int evenCount = count_if(numbers.begin(),numbers.end(),[](int n){
        return n % 2 == 0;
    });

    // Print the count of even number
    cout << "There are " << evenCount << " even numbers in the vector";
    cout << endl;

    return 0;
}
```

Finding Elements with std::find$_if$

```
#include <iostream>
#include <vector>
#include <array>
#include <algorithm> // for std::find
using namespace std;

void myPrint(int n){
    cout << n << " ";
}

bool is_Even(int &n)
{
    return n%2 == 0;
}

bool is_Odd(int &n)
{
    return n%2 != 0;
}


int main(){
    char find_value{};
    vector<int> vec = {15,5,7,10,13,4,2,8,11,9,12,14};
    //array<int,16> vec{15,5,7,10,13,4,2,8,11,6,1,9,12,0,3,14};
    cout << "sorted with the default operator <: std::less\n";
    sort(vec.begin(),vec.end());
    for_each(vec.begin(),vec.end(),myPrint);cout << endl;
    cout << "Find the first even or odd value ? (e/o): " << endl;
    cin >> find_value;
    int state = 1;

    while(state==1){
        if(find_value == 'e'){
        //auto it = find_if(vec.begin(),vec.end(),is_Even);
```

```
        auto it = find_if(vec.begin(),vec.end(),[](int n){return n%2 == 0;});
        if(it != vec.end()){
            cout << "Found even value " << *it << " at position " << (it-vec.begin())
        }else{
            cout << "No even values found " << endl;
        }
    }else if(find_value == 'o'){
        //auto it = find_if(vec.begin(),vec.end(),is_Odd);
        auto it = find_if(vec.begin(),vec.end(),[](int n){return n%2 != 0;});
        if(it != vec.end()){
            cout << "Found odd value " << *it << " at position " << (it-vec.begin())
        }else{
            cout << "No odd values found " << endl;
        }
    }
    cout << "Continue ? (y/n): " << endl;
    cin >> find_value;
    if(find_value == 'y'){
        int state = 1;
        cout << "Find the first even or odd value ? (e/o): " << endl;
        cin >> find_value;
    }else{
        state = 0;
    }

    }

    return 0;
}
```

Applying Functions with std::for$_e ach$

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

void doubleNumber(int &n)
{
    n *= 2;
}

int main(){
    vector<int> numbers = {1,2,3,4,5,6,7,8,9,10};
    cout << "Initial vector: \n";
    for_each(numbers.begin(),numbers.end(),[](int num){cout<< num << " ";});cout << e
    // use foreach to apply doubleNumber to each elemnt in the vector
    //for_each(numbers.begin(),numbers.end(),doubleNumber);
```

```
    for_each(numbers.begin(),numbers.end(),[](int &num){num *= 2;});

    // Print the modified number
    for(int number:numbers){
        cout << number << " ";
    }
    cout << endl;

    return 0;
}
```

Manual Iteration and Comparison

```
#include <iostream>
#include <vector>
#include <array>
#include <algorithm> // for std::find
using namespace std;

void myPrint(int n){
    cout << n << " ";
}

int main(){
    int find_value{};
    vector<int> vec = {15,5,7,10,13,4,2,8,11,6,1,9,12,0,3,14};
    //array<int,16> vec{15,5,7,10,13,4,2,8,11,6,1,9,12,0,3,14};
    cout << "sorted with the default operator <: std::less\n";
    sort(vec.begin(),vec.end());
    for_each(vec.begin(),vec.end(),myPrint);cout << endl;
    cout << "Pick a value to perform a search on: " << endl;
    cin >> find_value;

    auto it = find(vec.begin(),vec.end(),find_value);

    if(it != vec.end()){
        cout << "Found value " << *it << " at position " << (it-vec.begin()) << endl;
    }
    else{
        cout << "Value not found." << endl;
    }
    return 0;
}
```

Associative Container Demo: std::map

```
#include <iostream>
```

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



int main(){
    map<int,string> employees = {
        {101, "Alice"},
        {102, "Bob"},
        {103, "Charlie"},
        {104, "Diana"},
        {105, "Ethan"},
        {106, "Fiona"},
        {107, "George"},
        {108, "Hannah"},
        {109, "Ian"},
        {110, "Jasmine"}
    };
    cout << "First Employee: " << employees.at(101) << endl;
    cout << "Last Employee: " << employees.at(110) << endl;

    int index{105};
    if (employees.count(index)){
        cout << "Employee #"<< index <<" is in the system: " << employees.at(index) <
    }else{
        cout << "Employee #"<< index <<" is NOT in the system: " << endl;
    }

    int id{};string name{};
    cout << "enter the number of employees to add to the system: ";
    cin >> id;

    if(id<0){
        cout << "No new employees added" << endl;
    }else{
        int count = id;
        for (int i = 0; i < count; i++) {
            cout << "Please enter the new employee id: ";
            cin >> id;
            cout << "Please enter the new employee name: ";
            cin >> name;
            employees.insert(make_pair(id,name));
          }
    }

    cout << "Employee Records" << endl;
    for(auto it= employees.begin(); it != employees.end();++it){
```

```
        cout << "ID: " << it->first << "\nName: " << it->second << "\n---------" << e
    }

    char ans{};
    cout << "Do you want to delete any employee ? (y/n): ";
    cin >> ans;

    if(ans == 'y'){
        cout << "which empolyee id ?: ";
        cin >> id;
        employees.erase(id);
        cout << "Employee Records" << endl;
        for(auto it= employees.begin(); it != employees.end();++it){
            cout << "ID: " << it->first << "\tName: " << it->second << endl;
            cout << "ID: " << (*it).first << "\tName: " << (*it).second << "\n-------
        }

    }
    // employees[id] = name;

    // if (employees.count(id)){
    //     cout << "Employee #"<< id <<" has been added to the the system: " << emplo
    // }else{
    //     cout << "Employee #"<< id <<" has NOT been added to the system: " << endl;
    // }


    return 0;
}
```

Conditional Removal with std::remove$_if$

```
#include <iostream>
#include <vector>
#include <array>
#include <algorithm> // for std::find
using namespace std;

void myPrint(int n){
    cout << n << " ";
}

void myPrint(vector<int> v){
    for(int val:v){
        cout << val << " ";
    }
    cout << endl;
}
```

```
bool is_Even(int &n)
{
    return n%2 == 0;
}
// Use remove_if together with erase
int main(){
    vector<int> vec = {15, 5, 7, 9, 10, 13, 9, 4, 2, 8, 9, 11, 9, 9, 9, 12, 14};
    cout << "Initail vector" << endl;
    myPrint(vec);
    auto newEnd = remove_if(vec.begin(),vec.end(),is_Even);
    vec.erase(newEnd,vec.end()); // Erase the "removed elements"
    cout << "After calling: remove(vec.begin(),vec.end(),is_Even)" << endl;
    cout << "After calling: vec.erase(newEnd,vec.end())" << endl;
    myPrint(vec);


    return 0;
}
```

Value-based Removal with std::remove

```
#include <iostream>
#include <vector>
#include <array>
#include <algorithm> // for std::find
using namespace std;

void myPrint(int n){
    cout << n << " ";
}

void myPrint(vector<int> v){
    for(int val:v){
        cout << val << " ";
    }
    cout << endl;
}

// Use remove together with erase
int main(){
    vector<int> vec = {15, 5, 7, 9, 10, 13, 9, 4, 2, 8, 9, 11, 9, 9, 9, 12, 14};
    cout << "Initail vector" << endl;
    myPrint(vec);
    auto newEnd = remove(vec.begin(),vec.end(),9);
    cout << "After calling: remove(vec.begin(),vec.end(),9)" << endl;
    myPrint(vec);
    vec.erase(newEnd,vec.end()); // Erase the "removed elements"
    cout << "After calling: vec.erase(newEnd,vec.end())" << endl;
```

```
    myPrint(vec);


    return 0;
}
```

Conditional Replace Copy with std::replace$_copy_if$

```
#include <iostream>
#include <vector>
#include <array>
#include <algorithm> // for std::find
using namespace std;

void myPrint(int n){
    cout << n << " ";
}

void myPrint(vector<int> v){
    for(int val:v){
        cout << val << " ";
    }
    cout << endl;
}

bool is_Even(int &n)
{
    return n%2 == 0;
}

// Use unique together with erase
int main(){
    vector<int> vec = {1, 5, 3, 7, 5, 2, 8, 5, 9, 4, 6, 5, 10};
    vector<int> result(vec.size());

    // Replace all even numbers with 0
    cout << "Original vec vector" << endl;
    myPrint(vec);
    cout << "Original result vector" << endl;
    myPrint(result);
    // Copy vec to result replacing all '5's with '99's
    replace_copy_if(vec.begin(),vec.end(),result.begin(),is_Even,0);
    cout << "After copying from result vec to result vector, replacing even numbers w
    cout << "vec vector" << endl;
    myPrint(vec);
    cout << "result vector" << endl;
    myPrint(result);
    return 0;
```

```
}
```

Value Replace Copy with std::replace$_copy$

```
#include <iostream>
#include <vector>
#include <array>
#include <algorithm> // for std::find
using namespace std;

void myPrint(int n){
    cout << n << " ";
}

void myPrint(vector<int> v){
    for(int val:v){
        cout << val << " ";
    }
    cout << endl;
}

// Use unique together with erase
int main(){
    vector<int> vec = {1, 5, 3, 7, 5, 2, 8, 5, 9, 4, 6, 5, 10};
    vector<int> result(vec.size());

    // Replace all instances of 5s with 99
    cout << "Original vec vector" << endl;
    myPrint(vec);
    cout << "Original result vector" << endl;
    myPrint(result);
    // Copy vec to result replacing all '5's with '99's
    replace_copy(vec.begin(),vec.end(),result.begin(),5,99);
    cout << "After copying from result vec to result vector, replacing 5's with 99" <
    cout << "vec vector" << endl;
    myPrint(vec);
    cout << "result vector" << endl;
    myPrint(result);
    return 0;
}
```

In-place Conditional Replace with std::replace$_if$

```
#include <iostream>
#include <vector>
#include <array>
#include <algorithm> // for std::find
```

```
using namespace std;

void myPrint(int n){
    cout << n << " ";
}

void myPrint(vector<int> v){
    for(int val:v){
        cout << val << " ";
    }
    cout << endl;
}

bool is_Even(int &n)
{
    return n%2 == 0;
}

// Use unique together with erase
int main(){
    vector<int> vec = {1, 5, 3, 7, 5, 2, 8, 5, 9, 4, 6, 5, 10};

    // Replace all instances of 5s with 99
    cout << "Original vector" << endl;
    myPrint(vec);
    replace_if(vec.begin(),vec.end(),is_Even,0);
    myPrint(vec);
    return 0;
}
```

In-place Value Replace with std::replace

```
#include <iostream>
#include <vector>
#include <array>
#include <algorithm> // for std::find
using namespace std;

void myPrint(int n){
    cout << n << " ";
}

void myPrint(vector<int> v){
    for(int val:v){
        cout << val << " ";
    }
    cout << endl;
}
```

```
// Use unique together with erase
int main(){
    vector<int> vec = {1, 5, 3, 7, 5, 2, 8, 5, 9, 4, 6, 5, 10};

    // Replace all instances of 5s with 99
    cout << "Original vector" << endl;
    myPrint(vec);
    replace(vec.begin(),vec.end(),5,99);
    myPrint(vec);
    return 0;
}
```

Sorting with std::sort (ascending, descending, custom)

```
#include <iostream>
#include <vector>
#include <array>
#include <algorithm> // for std::for_each
using namespace std;

void myPrint(int n){
    cout << n << " ";
}

// Custom comparator function (Descending Order)
bool customCompare(int a,int b){
    return a>b;
}

int main(){
    array<int,16> s{15,5,7,10,13,4,2,8,11,6,1,9,12,0,3,14};
    cout << "sorted with the default operator <: std::less\n";
    sort(s.begin(),s.end());
    for_each(s.begin(),s.end(),myPrint);cout << endl;
    cout << "sorted with the standard library compare function: std::greater\n";
    sort(s.begin(),s.end(),std::greater<int>());
    for_each(s.begin(),s.end(),myPrint);cout << endl;
    s = {15,5,7,10,13,4,2,8,11,6,1,9,12,0,3,14};
    cout << "Reset the array with default initial sort state\n";
    for_each(s.begin(),s.end(),myPrint);cout << endl;
    cout << "sorted with the custom function that emulates the standard library compa
    sort(s.begin(),s.end(),std::greater<int>());
    for_each(s.begin(),s.end(),myPrint);cout << endl;
}
```

Finding Min/Max with STL Algorithms

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

int main(){
    vector<int> numbers = {42, 7, 18, 99, 23, 5, 66};
    vector<char> letters = {'d', 'a', 'z', 'm', 'b', 'e'};

    cout << "Numbers" << endl;
    for (auto& num : numbers){
        cout << num <<", ";
    }
    cout << endl;

    auto minIt = min_element(numbers.begin(),numbers.end());
    auto maxIt = max_element(numbers.begin(),numbers.end());

    cout << "Minimium element in vector: " << *minIt << endl;
    cout << "Maximum element in vector: " << *maxIt << endl;

    // for both min and max elements
    auto result = minmax_element(numbers.begin(),numbers.end());
    cout << "Min and Max elements are: " << *result.first << " and " << *result.secon

    cout << "Letters" << endl;
    for (auto& num : letters){
        cout << num <<", ";
    }
    cout << endl;

    auto min_It = min_element(letters.begin(),letters.end());
    auto max_It = max_element(letters.begin(),letters.end());

    cout << "Minimium element in vector: " << *min_It << endl;
    cout << "Maximum element in vector: " << *max_It << endl;

    // for both min and max elements
    auto resultt = minmax_element(letters.begin(),letters.end());
    cout << "Min and Max elements are: " << *resultt.first << " and " << *resultt.sec

    return 0;
}
```

Removing Adjacent Duplicates with std::unique

```
// iterator unique(iterator1, iterator2)
// Removing adjacent duplicates? → No sort needed
// Removing all duplicates?     → Yes, sort first

#include <iostream>
#include <vector>
#include <array>
#include <algorithm> // for std::find
using namespace std;

void myPrint(int n){
    cout << n << " ";
}

void myPrint(vector<int> v){
    for(int val:v){
        cout << val << " ";
    }
    cout << endl;
}

// Use unique together with erase
int main(){
    vector<int> vec =
       {1,
        2, 2,
        3, 3, 3,
        4, 4, 4, 4,
        5, 5, 5, 5, 5,
        6, 6, 6, 6, 6, 6,
        7, 7, 7, 7, 7, 7, 7,
        8, 8, 8, 8, 8, 8, 8, 8,
        9, 9, 9, 9, 9, 9, 9, 9, 9,
       10,10,10,10,10,10,10,10,10,10,
       11,11,11,11,11,11,11,11,11,11,11};
    cout << "Initail vector" << endl;
    myPrint(vec);
    auto newEnd = unique(vec.begin(),vec.end());
    vec.erase(newEnd,vec.end()); // Erase the "removed elements"
    cout << "After calling: unique(vec.begin(),vec.end())" << endl;
    cout << "After calling: vec.erase(newEnd,vec.end())" << endl;
    myPrint(vec);


    return 0;
}
```