BANGLADESH ARMY UNIVERSITY OF SCIENCE AND TECHNOLOGY

Lab Report

Department : Computer Science and Engineering

Course Title : Object-Oriented Programming Language I Sessional.

Course No : CSE 1204

Experiment No : 11

Topic: Object-Oriented Programming STL Concepts in C++

Submitted By:

Name: Md Waliur Rafiq Samir

Roll: 0802410205101088

Level: 1

Term: II Section: B

Date of Experiment: 25-11-2024

Date of Submission: 02-12-2024

Submitted To:

Anite Halim Sagor
Lecturer, Dept. of CSE, BAUST

Md. Manirujjaman
Lecturer, Dept. of CSE, BAUST

Signature: .....

❖ **Topic: Different Ways to Print a Vector in C++**

```cpp
#include <iostream>
#include <vector>
using namespace std;
int main(){
    vector<int> empty;
    vector<int> value0_gor5(5, 0);
    cout << "**Print value0_gor5**" << endl;
    cout << "value0_gor5 useing []: ";

    for (int i = 0; i < value0_gor5.size(); i++)
        cout << value0_gor5[i] << " ";
    cout << "\nvalue0_gor5 useing .at(): ";

    for (int i = 0; i < value0_gor5.size(); i++)
        cout << value0_gor5.at(i) << " ";

    vector<int> v1 = {1, 2, 3, 4, 5};
    cout << "\n\n**Print v1**" << endl;
    cout << "v1 useing []: ";
    for (int i = 0; i < v1.size(); i++)
        cout << v1[i] << " ";

    cout << "\nv1 useing .at(): ";
    for (int i = 0; i < v1.size(); i++)
        cout << v1.at(i) << " ";

    cout << "v1 using range-based for loop: ";
    for (int i : v1)
        cout << i << " ";

    cout << "\nv1 using data() method: ";
    int *p = v1.data();
    for (int i = 0; i < v1.size(); i++)
        cout << *(p + i) << " ";

    vector<int> v2{6, 7, 8, 9, 10};
    cout << "\n\n**Print v2**" << endl;
    cout << "v2 using range-based for loop: ";
    for (int i : v2)
        cout << i << " ";
    cout << "\nv2 using data() method: ";
    p = v2.data();
    for (int i = 0; i < v2.size(); i++)
        cout << *(p + i) << " ";
    return 0;}
```

Output:

```
**Print value0_gor5**
value0_gor5 useing []: 0 0 0 0 0
value0_gor5 useing .at(): 0 0 0 0 0

**Print v1**
v1 useing []: 1 2 3 4 5
v1 useing .at(): 1 2 3 4 5 v1 using range-based for loop: 1 2 3 4 5
v1 using data() method: 1 2 3 4 5

**Print v2**
v2 using range-based for loop: 6 7 8 9 10
v2 using data() method: 6 7 8 9 10
```

❖ **Topic: Manipulating and Printing a `std::vector` in C++**

```cpp
#include <iostream>
#include <vector>
using namespace std;
int main(){
    vector<int> number;
    int n, num;
    cout << "Enter n : ";
    cin >> n;
    cout << "Enter numbers: ";
    for (int i = 0; i < n; i++){
        cin >> num;
        number.push_back(num);
    }
    cout << "Number: ";
    for (int i = 0; i < number.size(); i++)
        cout << number[i] << " ";

    cout << "\nAdd (153,48,78) : ";
    number.push_back(153);
    number.push_back(48);
    number.push_back(78);
    for (int i = 0; i < number.size(); i++)
        cout << number[i] << " ";

    cout << "\nPop last 2 element : ";
    number.pop_back();
    number.pop_back();
    for (int i = 0; i < number.size(); i++)
        cout << number[i] << " ";

    cout << "\nResize 10 and gap value 3 : ";
    number.resize(10, 3);
    for (int i = 0; i < number.size(); i++)
        cout << number[i] << " ";

    cout << "\nInsert 0 in last 3 position : ";
    number.insert(number.end() - 3, 0);

    for (int i = 0; i < number.size(); i++)
        cout << number[i] << " ";

    cout << "\nPrint use iterator: ";
    vector<int>::iterator it;
    for (it = number.begin(); it != number.end(); it++)
        cout << *it << " ";

    cout << "\nErase position 5-10 : ";
    number.erase(number.begin() + 5, number.end());
    for (it = number.begin(); it != number.end(); it++)
        cout << *it << " ";
    vector<int> id = {88, 87, 86, 85, 81};
    cout << "\nPrint id: ";
    for (it = number.begin(); it != number.end(); it++)
        cout << *it << " ";

    cout << "\nPrint Number after swap : ";
    swap(id, number);
    for (it = number.begin(); it != number.end(); it++)
        cout << *it << " ";
    cout << "\nPrint id after swap: ";
```

```
    for (it = id.begin(); it != id.end(); it++)
        cout << *it << " ";
    return 0;

}
```

Output:

```
Enter n : 5
Enter numbers: 11 12 13 14 15
Number: 11 12 13 14 15
Add (153,48,78) : 11 12 13 14 15 153 48 78
Pop last 2 element : 11 12 13 14 15 153
Resize 10 and gap value 3 : 11 12 13 14 15 153 3 3 3 3
Insert 0 in last 3 position : 11 12 13 14 15 153 3 0 3 3 3
Print use iterator: 11 12 13 14 15 153 3 0 3 3 3
Erase position 5-10 : 11 12 13 14 15
Print id: 11 12 13 14 15
Print Number after swap : 88 87 86 85 81
Print id after swap: 11 12 13 14 15
```

❖ Topic: Vector in class

```
#include <iostream>
#include <vector>
using namespace std;
class student{
public:
    int id;    string name;
    student(int i = 0, string n = "") : id(i), name(n) {}
};

int main(){
    vector<student> stdList = {
        {1, "Waliur Rafiq SAMI"},
        {2, "Siam Ahmed"},
        {3, "Milon Ahmed Mobashir"},
        {4, "Arif Sikdar"}};
    cout << "**Student Info : " << endl;
    vector<student>::iterator i;
    for (i = stdList.begin(); i != stdList.end(); i++)
        cout << "Id : " << i->id << ",  Name : " << i->name << endl;

    cout << "**After add a new element: " << endl;
    stdList.push_back({5, "SHAN"});
    for (i = stdList.begin(); i != stdList.end(); i++)
        cout << "Id : " << i->id << ",  Name : " << i->name << endl;

    cout << "**Remove last element: " << endl;
    stdList.pop_back();
    for (i = stdList.begin(); i != stdList.end(); i++)
        cout << "Id : " << i->id << ",  Name : " << i->name << endl;

    cout << "**Remove first element: " << endl;
    stdList.erase(stdList.begin());
    for (i = stdList.begin(); i != stdList.end(); i++)
        cout << "Id : " << i->id << ",  Name : " << i->name << endl;

    return 0;}
```

Output:

```
**Student Info :
Id : 1,  Name : Waliur Rafiq SAMI
Id : 2,  Name : Siam Ahmed
Id : 3,  Name : Milon Ahmed Mobashir
Id : 4,  Name : Arif Sikdar
**After add a new element:
Id : 1,  Name : Waliur Rafiq SAMI
Id : 2,  Name : Siam Ahmed
Id : 3,  Name : Milon Ahmed Mobashir
Id : 4,  Name : Arif Sikdar
Id : 5,  Name : SHAN
**Remove last element:
Id : 1,  Name : Waliur Rafiq SAMI
Id : 2,  Name : Siam Ahmed
Id : 3,  Name : Milon Ahmed Mobashir
Id : 4,  Name : Arif Sikdar
**Remove first element:
Id : 2,  Name : Siam Ahmed
Id : 3,  Name : Milon Ahmed Mobashir
Id : 4,  Name : Arif Sikdar
```

❖ Topic: STL Algorithm (sort, find, reverse)

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

int main()
{
    vector<int> intArray = {0, 5, 8, 9, 2, 4, 7, 3, 6, 1};
    cout << "Elements are : ";
    for (int i : intArray){cout << i << " ";}

    auto findElement = find(intArray.begin(), intArray.end(), 7);
    findElement != intArray.end()
        ? cout << "\nWOW! Element 7 Found\n"
        : cout << "\nSorry! Not Found\n";

    cout << "After short : ";
    sort(intArray.begin(), intArray.end());
    for (int i : intArray)
    for (int i : intArray){cout << i << " ";}

    cout << "\nAfter reverse : ";
    reverse(intArray.begin(), intArray.end());
    for (int i : intArray)
    for (int i : intArray){cout << i << " ";}

    return 0;
}
```

Output:

```
Elements are : 0 5 8 9 2 4 7 3 6 1
WOW! Element 7 Found
After short : 0 1 2 3 4 5 6 7 8 9
After reverse : 9 8 7 6 5 4 3 2 1 0
```

❖ Topic: STL SIMPLE SET

```cpp
#include <iostream>
#include <set>
using namespace std;
int main(){
    set<int> Set = {1, 2, 3, 1, 1, 4, 5, 6, 7, 7, 8, 1, 9};
    cout << "Visit SET: ";
    for (auto i : Set)
        cout << i << " ";

    cout << "\nAfter add 9 and 10: ";
    Set.insert(9); // allready add so don't add that time
    Set.insert(10);
    for (auto i : Set)
        cout << i << " ";

    cout << "\nAfter erase 3 and 9: ";
    Set.erase(3);
    Set.erase(9);
    set<int>::iterator i;
    for (i = Set.begin(); i != Set.end(); i++)
        cout << *i << " ";

    return 0;
}
```

Output:

```
Visit SET: 1 2 3 4 5 6 7 8 9
After add 9 and 10: 1 2 3 4 5 6 7 8 9 10
After erase 3 and 9: 1 2 4 5 6 7 8 10
```

❖ Topic: STL SET FOR CLASS

```cpp
#include <iostream>
#include <set>
using namespace std;
class student
{
public:
    int id;
    string name;
    student(int i = 0, string n = "") : id(i), name(n) {};
    bool operator<(const student &s) const
    {
        return id < s.id;
    }
};

int main()
{
    set<student> studentSet = {{3, "Milon Ahmed Mobashir"},
                               {1, "Waliur Rafiq SAMI"},
                               {2, "Siam Ahmed"}};

    cout << "**Visit studentSet: \n";
    for (auto i : studentSet)
        cout << "\tID: " << i.id << ", Name: " << i.name << endl;

    cout << "**After insert 2 student: \n";
```

```
    studentSet.insert({4, "MR. SAMI"});
    studentSet.insert({5, "Milon Khan"});
    for (auto i = studentSet.begin(); i != studentSet.end(); i++)
        cout << "\tID: " << i->id << ", Name: " << i->name << endl;

    cout << "**After remove 'Milon Ahmed Mobashir' student: \n";
    studentSet.erase(student(3, "Milon Ahmed Mobashir"));
    for (auto i = studentSet.begin(); i != studentSet.end(); i++)
        cout << "\tID: " << i->id << ", Name: " << i->name << endl;
    return 0;
}
```

Output:

```
**Visit studentSet:
        ID: 1, Name: Waliur Rafiq SAMI
        ID: 2, Name: Siam Ahmed
        ID: 3, Name: Milon Ahmed Mobashir
**After insert 2 student:
        ID: 1, Name: Waliur Rafiq SAMI
        ID: 2, Name: Siam Ahmed
        ID: 3, Name: Milon Ahmed Mobashir
        ID: 4, Name: MR. SAMI
        ID: 5, Name: Milon Khan
**After remove 'Milon Ahmed Mobashir' student:
        ID: 1, Name: Waliur Rafiq SAMI
        ID: 2, Name: Siam Ahmed
        ID: 4, Name: MR. SAMI
        ID: 5, Name: Milon Khan
```

Conclusion:

The lab report demonstrates a comprehensive understanding of various STL containers and their functionalities in C++, such as vector, set, and their integration with classes. It effectively showcases different techniques for iterating, manipulating, and accessing elements using loops, iterators, and member functions. The practical examples highlight the versatility and efficiency of STL in solving complex programming tasks, emphasizing their importance in modern C++ development. Through these implementations, the report provides valuable insights into optimizing code readability and performance, preparing students for real-world software engineering challenges.

THE END