Lab 5 – Standard libraries and generic algorithms

Object-oriented programming in C ++

The use of generic algorithms and function objects.

Usage of Generic Algorithms using C++ STL

Generic Manipulation class for handling the operations on List Data structure

```
template<typename T>
class ListManipulator{
private:
    std::list<T> *theList;

public:
    ListManipulator(std::list<T> *aList);
    ~ListManipulator(){}

    void fillList();
    T sumList() const;
    T listAverage() const;
    bool findFirst1500_1900(T &num) const;
    void divideByTwo();
    void swapPlaces();
    void findMinMax(T &min, T &max) const;
    void sortList();
    void clearList();
    std::list<T> getList() const;
    void saveToFile() const;
    void readFromFile();
};
```

Generic Algorithm and Data structure in STL

Algorithm in C++ STL

```
#include <algorithm>
#include <random>
#include <ctime>
#include <numeric>
#include <functional>
#include <iterator>
```

Data Structure from STL C++

```
#include <list>
```

In STL there are a bundle of built-in data structures, from which we can use **List** along with a number of algorithms utilize from STL.

Algorithms going to be used from STL

For Uniform real number distribution within given range

```
std::uniform_real_distribution<double> random(1000, 2000);
```

For random number generation

```
std::default_random_engine generator(static_cast<unsigned>(std::time(0)));
```

For Populating my List with random numbers

```
std::generate(theList->begin(), theList->end(), random1());
```

For finding the average of a list, I calculated the sum using **accumulate function**

```
std::accumulate(start, end, T(0)) / theList->size();
```

Dividing the whole list by 2 using transform and bind function

```
// devide all numbers with two. Binds the value read from the list of arguments _1
std::transform(theList->begin(), theList->end(), theList->begin(), std::bind(std::divides<T>(),_1, 2));
```

Algorithms going to be used from STL(continued)

For finding the value with in given range, I used **find_if**, which returns the iterator to first occurrence.

```
auto it = find_if(theList->begin(), theList->end(), between1500and1900tester<T>());
if (it != theList->end()) { // value Found
    num = *it;
    return true;
}
else return false; // If no value is found returns false
```

Here, the **find_if** function takes 3 arguments, starting position, ending position and condition*.

*Condition could be a function (as mentioned above) or a *Lambda expression (as given below)*

```
auto it = find_if(theList->begin(), theList->end(), [](T a) {return a >= 1500 && a <= 1900; });
```

*For swapping the whole list, we used **iter_swap** which swaps the elements to which iterator is pointing.*

```
for (size_t i = 0; i < theList->size() / 2;i++) {
    std::iter_swap(theList->begin(), theList->end());
    fIt++; bIt--;
}
```

Algorithms going to be used from STL(continued)

For finding the minimum and maximum we used **min_element** and **max_element** functions.

```
T min = *std::min_element(theList->begin(), theList->end());
T max = *std::max_element(theList->begin(), theList->end());
```

For sorting the list in ascending order, we used **sort** function of List data structure.

```
theList->sort();
```

For deleting all the elements of list, we use **clear** functions of List data structure.

```
theList->clear();
```

For reading from file, we use the **ifstream** object with starting and ending **ifstream_iterator** to **copy** the whole list.

```
std::ifstream inFile("list.dat");
std::istream_iterator<T> eos;
std::istream_iterator<T> iit(inFile);
std::copy(iit, eos, std::back_inserter(*theList));
```

Test program

```
Choose type of list.
1. Int.
2. Double.
2

======== Menu ========
 1. Fill list with random numbers.
 2. Summerize the values in the list.
 3. average of numbers.
 4. find first number between 1500 and 1900
 5. Divide by two.
 6. Swap places.
 7. Find largest and smallest number.
 8. Sort.
 9. Clear list.
10. Write to file.
11. Read from file.
12. Print numbers.
13. Quit.
Make your choice: 1
The list has been filled with 20 random numbers.
```

```
Make your choice: 12
1874.18
1373.50
1733.29
1233.68
1563.12
1900.00
1638.21
1176.83
1016.81
1581.33
1446.64
1316.08
1758.27
1226.50
1596.89
1403.89
1771.82
1401.88
1837.67
1421.67
```

```
Make your choice: 6
The elements have swapped places.
```

```
Make your choice: 12
1421.67
1837.67
1401.88
1771.82
1403.89
1596.89
1226.50
1758.27
1316.08
1446.64
1581.33
1016.81
1176.83
1638.21
1900.00
1563.12
1233.68
1733.29
1373.50
1874.18
```