SCSSE, University of Wollongong

CSCI204/MCS9204 Object and Generic Programming in C++

Laboratory Exercise Week 11

Task One: String stream

Use **ostringstream** and **istringstream** to implement the following ***two functions*** in a file **stringIO.cpp** by assuming that the operators << and >> have been overloaded for type T.

```
template<class T>
string toString (T value); // convert a value into a string

template <class T>
T toValue(string str); // extract the value from a string
```

Defined a class **Student** in a file **Student.h** and implemented the member functions in a file **Student.cpp**. The class Student can be defined like following

```
class Student {
    private:
        string firstname;
        string lastname;
        int id; // student number
        float gpa;
    public:
        // all necessary functions to be defined here
        … …
};
```

Implement a main() function in **stringIO.cpp** to test the two functions in the cases where T is **integer**, **double** and **class Student.**

You need to define and implement the student class properly in order to test the two template functions and assume the input string for the student class is in the following format.

First-name:last-name:student-number:gpa

For example:
John:Anderson:1234567:6.35

Testing:

Compile the program in this task by:

```
CC –o task1 stringIO.cpp Student.cpp
```

Run the program (You may use different values)

```
./task1
Input an integer: 12345678
Integer to string: 12345678
String to integer: 12345678
```

Input a double: 3214.654
Double to string: 3214.65
String to double: 3214.65
Input a student record (first-name:last-name:number:gpa): David:Smith:1234567:3.65
Student to string:
David:Smith:1234567:3.65
String to Student:
David:Smith:1234567:3.65

Note: The outputs above indicate different types of data.

Task Two: Containers and generic functions

Implement C++ code in a file **task2.cpp** include main() function that read input data for students; add students' records in a **vector** container; use generic functions **copy** to copy the students' records from a **vector** to a **deque** container; display the records in the deque container by using its iterator to travel through the container; then use **sort** function to sort the students' records by their last names and first names inside the deque container; finally display the sorted records by using the generic **copy** function.

Hint: You should define comparison operator (<) for the class Student and define a function object to compare two students.

Testing:

Compile the program in this task by:
CC –o task2 task2.cpp Student.cpp

./task2
Number of Student records: 5
John:Smith:1234567:3.21
Taylor:Swift:7654321:5.43
Joan:Anderson:1122334:4.25
Bob:Cook:2211443:3.89
Marry:Brand:1234432:5.23

There are 5 records in the vector.

Copy Students from vector to deque:
John:Smith:1234567:3.21
Taylor:Swift:7654321:5.43
Joan:Anderson:1122334:4.25
Bob:Cook:2211443:3.89
Marry:Brand:1234432:5.23

Sorted records in deque:

Joan:Anderson:1122334:4.25
Marry:Brand:1234432:5.23
Bob:Cook:2211443:3.89
John:Smith:1234567:3.21
Taylor:Swift:7654321:5.43

You can download the testing file **input2.txt** from Moodle.

Task Three: Binary file and vector

Download the source files **Employee.h** and **Employee.cpp** from the web site. Implement main() function in a file **task3Main.cpp** that read employees' records from the given binary file from the command line and save the records into a **vector** container. Then use **iterator** to access each Employee record in the **vector** container and print out all the information.
You will consider there is no file name passed from the command line or the file not exists.

Testing:

Compile the program in this task by:
CC –o task3 task3Main.cpp Employee.cpp

When you run the program
./task3
Usage: ./task3 binary-file-name

When you run the program with a file name which does not exist.
./task3 mydata.dat
Error: The file mydata.dat not exists or cannot be opened.

When you test the program with a binary file (**edate.bin** can be downloaded from the web site),
./task3 edata.bin

The output will be

Employee ID: 100
First name: aaa
Last name: bbb
Pay rate/hour: $00105.28

Employee ID: 101
First name: bbb
Last name: ccc
Pay rate/hour: $00120.38