SCSSE, University of Wollongong

CSCI204/MCS9204 Object and Generic Programming in C++

Laboratory Exercise Week 12

Task One: String split

In this task, you will define a class that stores a string and can split the string object into tokens according to the delimiters that given.
Define a class **stringSplit** in a **namespace MyLib** in a file **stringSplit.h**. The class contains following data members: a string object; a delimiter object; a suitable (STL) container (such as vector, deque, list) object for tokens; an iterator object for the container which points the current token.
Define proper constructor(s), set functions to set string and delimiters for the class **stringSplit**. Define the following member functions in the class stringSplit:

- Constructor(s).
- **setString(const std::string &)**: The function set string for the class.
- **setDelimiters(const std::string &)**: The function set delimiters for the class.
- **void split(bool)**: The function can split the string into tokens according to the delimiters. When the bool value is true, the tokens contain delimiters; when it is false, the tokens only contain non delimiter tokens. The iterator object initially points to the first token.
- **bool hasMore()**: The function will return true if more tokens left; return false when there is no more token.
- **std::string next()**: The function will return the token that the iterator points to, and the iterator points to the next token.

Implement the member functions in a file **stringSplit.cpp**. You can download the file **task1Main.cpp** to test the member functions above in the class stringSplit.

Testing:

You can compile your program

```
CC –o task1 task1Main.cpp stringSplit.cpp
```

Then run the program

```
./task1
```

The output data can be found in a text file **task1.txt**.

Task Two:

The program **vectorProc.cpp** intends to implement and test a template function:

```
template<class T>
bool same_elements(const vector<T>  &val1, const vector<T> &val2);
```

that checks whether two vector containers have the same elements with the same multiplicities. For example,

"1 4 9 16 9 7 4 9 11" and "11 1 4 9 16 9 7 4 9" would be considered identical, but "1 4 9 16 9 7 4 9 11" and "11 11 7 9 16 4 1" would not.
And a template function that removes duplicates from a vector container:

```
template<class T>
void remove_duplicates(vector<T> &val);
```

Complete the implementation of these two functions. You will probably need one or more helper (template) functions for the implementation. Place the helper functions in the file **vectorProc.cpp**. Complete the **main()** function and test your implementation.

You can download test files **input1.txt**, **input2.txt** and **input3.txt** from the web site to test your program.

Testing:
You can compile your program

```
CC –o task2 vectorProc.cpp
```

Then run the program

```
./task2 < input1.txt
Input number of elements for integer v1: 9
Input data: 1
Input data: 4
Input data: 9
Input data: 16
Input data: 9
Input data: 7
Input data: 4
Input data: 9
Input data: 11
Input number of elements for integer v2: 9
Input data: 11
Input data: 1
Input data: 4
Input data: 9
Input data: 16
Input data: 9
Input data: 7
Input data: 4
Input data: 9
v1 and v2 contain same elements.

After remove duplicates from v1, now it contains:
1 4 9 16 7 11

After remove duplicates from v2, now it contains:
11 1 4 9 16 7

Input number of elements for double v1: 5
Input data: 1.5
Input data: 4.3
Input data: 9.6
Input data: 16.2
Input data: 9.6
Input number of elements for double v2: 5
Input data: 9.6
Input data: 16.2
Input data: 1.5
Input data: 9.6
Input data: 4.3
v1 and v2 contain same elements.

After remove duplicates from v1, now it contains:
1.5 4.3 9.6 16.2

After remove duplicates from v2, now it contains:
9.6 16.2 1.5 4.3
```