Lab 10: Maps & Sets – using the STL to organize data

CPSC 1021 – version 4/8/24

School of COMPUTING Clemson University

Lab Objective

- to familiarize students with the use of STL containers like
  1. std::map
  2. std::vector
  3. std::set
  4. std::pair
- along with introducing file handling
- object-oriented programming concepts
- handling cross-platform file format nuances

Collaboration Policy

Work with your assigned team members from Lab 0. ***Please make sure to integrate your team members who – because of COVID requirements – have to participate remotely.*** You can use the MS Teams video conferencing feature to collaborate with a remote team member. Use the search bar on the top of the MS Teams app to search for your team member with their CU user name, then click the video icon in the top-right corner of the screen to start a video call.

You and your partners can work on the assignments together.

Submission Policy

Code solutions to all challenges **can be submitted as teams** this week. This means only one member from each team needs to submit the code files on Gradescope (this person must then add all their other team members to the submission). However, **you are required to work on all problems together**.

Introduction

In this lab, we will again learn how to use maps, sets and vectors of pairs to solve a portion of Project 4 by reading in a similar data file. Your group will write a program that reads movie reviews from a file, where each review consists of a movie title, reviewer name, and rating. The goal is to process this data to generate various analytics (which will happen in Project 4).

Part 1: Introduction to STL Containers

1. **Maps in C++**:

- **Explanation**: Maps are associative containers that store elements formed by a combination of a key value and a mapped value, following a specific order. In C++, std::map is commonly used for this purpose.

- **Demonstration**: Create a simple program that uses std::map<std::string, int> to count the number of occurrences of words in a sentence. Input a sentence from the user, split it into words, and use the words as keys with their counts as values.

```
/* This program demonstrates the use of a std::map. When you run the
program, it will prompt you to enter a sentence. After entering the
sentence, it will output each word in the sentence along with the number
of times it appears.

NOTE: a few pieces of the program have been removed for you to figure
out and thereby learn maps. */

#include <iostream>
#include <map>
#include <sstream>

int main() {
    std::string sentence = "In the beginning was the Word and the Word
was with God and the Word was God the same was in the beginning with
God";

    [declare a map which consists of a string as key and int as value,
called wordcount]
    std::istringstream iss(sentence);
    std::string word;

    while (iss >> word) {
        [increment the map's int using word as the key]
    }

    for (const auto& pair : wordCount) {
        std::cout << [access the first element of pair] << ": " <<
pair.second << std::endl;
    }

    return 0;
}
```

Submission Instructions

- Submit your complete and correct program to Gradescope. Your submission must include the following file: main.cpp

- Make sure your program compiles and runs on the SoC Linux machines; the autograder automatically assigns 0 points to programs that do not compile.

2. **Vector of Pairs**:

- **Explanation**: A std::vector is a sequence container that encapsulates dynamic size arrays. A pair is a simple container consisting of two elements. Combining them,

std::vector<std::pair<std::string, std::vector<int>>> can represent complex data structures.

- **Demonstration**: Create a program that initializes a std::vector of pairs, where each pair consists of a string (key) and a vector of integers (values). Populate it with sample data and display the contents.

```
/* In this program, each pair represents a student's name along with their
scores in different subjects. The program populates the vector with sample
student data and then displays each student's name followed by their
corresponding scores.*/

#include <iostream>
#include <vector>
#include <utility>
using namespace std;

int main() {
    vector<pair<string, vector<int>>> studentData = {
        {"Alice", {85, 90, 88, 92, 95}},
        {"Bob", {78, 80, 85}},
        {"Charlie", {92, 95, 90, 87, 76, 90}},
        {"Diana", {80, 85}}
    };

    // Display the contents
    for (const auto& student : studentData) {
        cout << student.first << "'s scores: ";
        for (int score : student.second) {
            cout << score << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Modify the above program to output each student's name and their AVERAGE score rather than the individual scores. Be sure to take into account the number of scores for each student. Output should look like this:

```
Alice: 90
Bob: 81
Charlie: 88
Diana: 82
```

Submission Instructions

- Submit your complete and correct program to Gradescope. Your submission must include the following file: main.cpp
- Make sure your program compiles and runs on the SoC Linux machines; the autograder automatically assigns 0 points to programs that do not compile.

3. **Sets**:

- **Explanation**: A std::set is an associative container that contains a sorted set of unique objects of type Key. Adding members to a set has the side benefit of eliminating duplicates.

- **Demonstration**: Create a program that initializes a std::set. Populate it with sample data which contains some duplicates and display the contents.

```
/* In this program, the initial `std::vector` `stringData` contains strings, some
of which are duplicates. These strings are then added to a `std::set` to remove
duplicates. The unique elements are then transferred back to a new `std::vector`
called `uniqueStrings`, which is then displayed.

NOTE: you need to finish the code as specified in the comments
NOTE: std::set was added to C++ in verion 17 so you may need to compile with -
std=c++17 */

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

int main() {
    vector<string> stringData = {"apple", "banana", "apple", "orange", "banana",
"grape", "kiwi", "banana", "mango", "orange", "apple", "cherry", "mango", "kiwi",
"strawberry", "grape", "cherry", "apple", "orange", "banana"};

    // Create a set to remove duplicates
    set<string> stringSet(stringData.begin(), stringData.end());

    // Create a new vector with unique elements from the set
    vector<string> uniqueStrings(stringSet.begin(), stringSet.end());

    // Display the unique elements in the new vector using a single space to
separate each one. Use an endl at the end of the list.


    return 0;
}
```

Submission Instructions

- Submit your complete and correct program to Gradescope. Your submission must include the following file: main.cpp

- Make sure your program compiles and runs on the SoC Linux machines; the autograder automatically assigns 0 points to programs that do not compile.

Part 2: Putting it all together to read in movie review data

- **Explanation**: Write a program that will read in a text file of movie ratings and display first the list of movies followed by each reviewer's movie ratings such that the ratings are in the same order as the movies. IF a reviewer does not review a movie the rating needs to be set to zero in the ratings vector associated with each reviewer in the map.

- **Demonstration**: Here's working code to start with, but read the comments carefully to see there is a problem you need to correct.

```
/* This is an almost complete solution to the problem to the movie
   review problem. HOWEVER, you will notice an error in the output that
   shows the ratings from each reviewer. Identify this error and fix the
   code to work properly. */

#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <set>
#include <map>

using namespace std;

int main() {
    // Create a set to store unique movie titles
    set<string> uniqueMovieTitles;

    // Read the file to extract unique movie titles
    ifstream file("movie_reviews.txt");
    string movieTitle, reviewerName, rating;
    while (getline(file, movieTitle) && getline(file, reviewerName) &&
getline(file, rating)) {
        uniqueMovieTitles.insert(movieTitle);
    }
    file.close();

    // Copy unique movie titles from set to vector
    vector<string> movieTitles(uniqueMovieTitles.begin(), uniqueMovieTitles.end());

    // Create a map to store reviewer names as keys and vectors of ratings as
values
    map<string, vector<int>> reviewerRatings;

    // Read the file again to extract reviewer names and ratings
    file.open("movie_reviews.txt");
    while (getline(file, movieTitle) && getline(file, reviewerName) &&
getline(file, rating)) {
        reviewerRatings[reviewerName].push_back(stoi(rating));
    }
    file.close();

           // Output each movie title separated by /
           cout << "HERE ARE THE MOVIE TITLES\n";
```

```
        cout << "==========================\n";
    for (const auto& m : movieTitles) {
        cout << m << " / ";
    }
    cout << endl << endl;

    // Output each reviewer followed by their ratings for each movie
        cout << "HERE ARE THE MOVIE RATINGS\n";
        cout << "==========================\n";

    for (const auto& pair : reviewerRatings) {
        cout << pair.first << ": ";
            auto ratings = pair.second;
            auto it = ratings.begin();
            while (it != ratings.end()) {
                cout << *it;
                ++it;
                if (it != ratings.end()) {
                    cout << ",";
                }
            }
            cout << endl;
    }

    return 0;
}
```

This is what the correct output should look like. Notice the 0's any time a movie was NOT rated by a reviewer.

```
HERE ARE THE MOVIE TITLES
==========================
Avatar / Blade Runner / Blade Runner 2049 / Cars / Dunkirk / Finding Nemo / Ince
ption / Interstellar / Jurassic Park / La La Land / Mad Max: Fury Road / Moonris
e Kingdom / Pulp Fiction / Ratatouille / The Dark Knight / The Grand Budapest Ho
tel / The Lion King / The Lord of the Rings / The Matrix / The Social Network /
Toy Story / Up / WALL-E /

HERE ARE THE MOVIE RATINGS
==========================
Alice Johnson: 0,0,0,0,0,0,-3,4,0,0,0,0,0,0,4,-4,5,0,0,0,0,0,4
Bob Smith: 0,0,5,0,0,0,-2,0,0,0,0,3,-1,0,0,0,0,4,0,0,0,4,0
Carol Lee: -3,0,0,0,0,0,0,4,0,0,-5,0,0,5,0,0,0,0,0,2,0,0,0
Dan Brown: 2,0,0,3,0,0,3,-4,0,5,0,0,5,0,0,0,0,0,0,0,0,0,0
Eve Davis: 0,0,0,0,-2,4,0,0,0,0,4,0,0,0,5,0,0,0,2,0,0,0,0
Frank Miller: -5,-2,5,3,-3,0,0,4,0,0,3,0,0,0,0,5,0,0,0,0,0,0,0
Grace White: 0,4,0,0,0,5,4,0,3,0,0,-4,3,0,0,0,5,0,0,4,0,0,0
Harry Adams: 0,0,0,0,0,0,0,0,3,4,0,0,0,4,5,0,0,0,-2,0,-1,3,-4
```

Submission Instructions

- Submit your complete and correct program to Gradescope. Your submission must include the following file: main.cpp
- Make sure your program compiles and runs on the SoC Linux machines; the autograder automatically assigns 0 points to programs that do not compile.