Lab Session: STL containers

Bruno Guindani

Politecnico di Milano

bruno.guindani@polimi.it

The Problem

- Write a program that manages the list of players of an arcade video-game and their scores. Create a function that can find and return a player with a given username
- As a starting point, you are provided with the implementation based on vector to collect players. Also a binary_search function has been implemented
- In the provided example, the search is profiled in case the player is in the vector or he/she is not
- Implement your solution once by using a set (ordered) to collect players, and another time using map (unordered)

The Questions

- Add to search.cc:
  - bool set_search (
    const std::set<Arcade::Player> & player_set,
    unsigned username);
  - bool map_search (
    const std::unordered_map<unsigned, Arcade::Player> & player_map,
    unsigned username);
- Implement stl_binary_search() for the initial vector-based solution, by relying on the STL binary_search which is:
  - bool binary_search(
    player_vec_type::const_iterator first,
    player_vec_type::const_iterator last,
    const Arcade::Player & value);
- Complete the main

Solution - 1

1) Write a class for players. Each player has a username and a vector of scores. In your main function, define a collection of players.

   ```
   typedef std::set<Player> player_set_type;
   player_set_type player_set;
   ```

   Hint: you need to overload operator< for class Player. Why?!

   ```
   bool operator<(const Player &lhs, const Player &rhs)
   ```

2) Try inserting some new elements in your set, using container specific insert.

   ```
   player_set.insert(random_player);
   ```

3) Define a search function that takes the collection of players and a username and looks for username within the container.

   ```
   bool set_search(const player_set_type & stud_set, unsigned username)
   ```

Solution - 2

Do the same process by using
an unordered_map<id, player> instead of a set.

```
typedef std::unordered_map<unsigned, Player> player_map_type;
player_map_type player_map;

player_map.insert(make_pair(random_player.getUsername(),random_player));

bool map_search(const player_map_type & player_map, unsigned username);
```