



UNIVERSITÀ
DEGLI STUDI
FIRENZE



Esercitazione

STL map exercise

Obiettivo

- Il progetto CLion fornito contiene classi e scheletri di classi relative ad un gestore di attività (es. Promemoria di macOS).
- Scopo della presente esercitazione è:
 - Implementare una multimappa in grado di contenere Task associati sulla base della loro data di scadenza
 - Operare con algoritmo di ricerca su albero binario per cercare elementi o set di elementi nella multimappa
 - Iterare su tutti gli elementi della multimappa
 - Implementare un functor per confrontare date e consentire l'uso di oggetti di tipo Date come chiavi della multimappa

Schema del codice

- Il programma è composto da 2 classi di partenza:
- Date rappresenta una data.
 - E' possibile sapere se una data è minore di un'altra col codice:

```
if (first.getYear() > second.getYear())
    return false;
else if (first.getYear() == second.getYear()) {
    if (first.getMonth() > second.getMonth())
        return false;
    else if (first.getMonth() == second.getMonth()) {
        if (first.getDay() >= second.getDay())
            return false;
    }
}
return true;
```

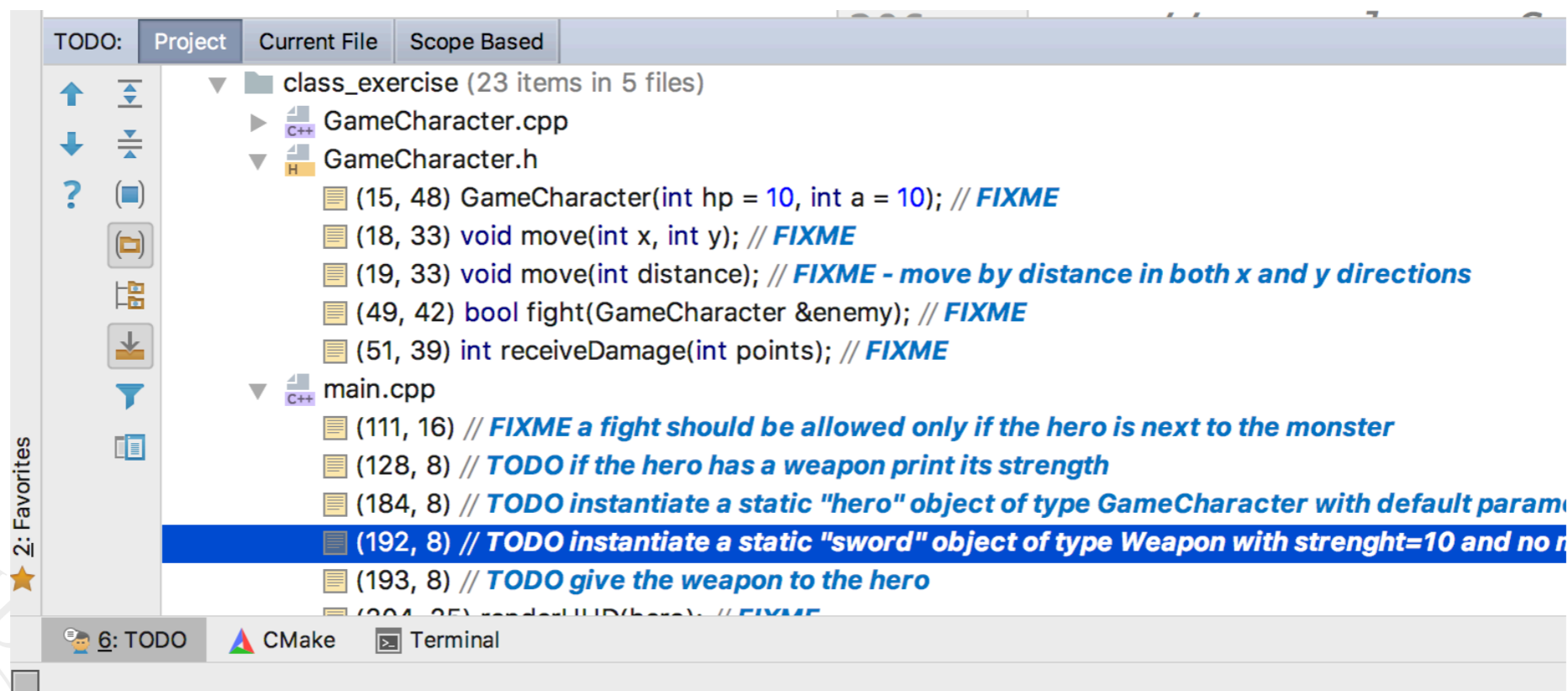
- Task rappresenta un'attività da compiere, con descrizione testuale, data di scadenza e booleano che ne indica il completamento

Schema del codice

- In questa esercitazione andremo ad aggiungere un functor per comparare date in: `Date.cpp/h`
- Modificheremo il codice di `main.cpp` per istanziare una multimappa che associa date a task.

Dove modificare il codice

- Le indicazioni precise sul codice da modificare sono fornite come commenti indicati con TODO e FIXME
- Per vedere tutti questi commenti selezionare la finestra TODO di CLion



Dove modificare il codice

The screenshot shows the CLion IDE interface. The main editor displays the code in `main.cpp` with the following content:

```
185 // find a legal start position
186 int startX = 0;
187 int startY = 0;
188 setupCharacterCell(startX, startY, map);
189 hero.setPosX(startX);
190 hero.setPosY(startY);
191 // create a weapon and give it to hero
192 // TODO instantiate a static "sword" object of type Weapon with strenght=10 and
193 // TODO give the weapon to the hero
194 // create an enemy with a low grade armor
195 GameCharacter enemy(20, 2);
196 // find monster position not too far from hero position
197 startX += 5;
198 startY += 3;
199 setupCharacterCell(startX, startY, map);
200 enemy.setPosX(startX);
201 enemy.setPosY(startY);
202
203 // render
204 renderHUD(hero); // FIXME
205 renderGame(map, hero, enemy);
```

The TODO list at the bottom of the IDE shows the following items:

- (15, 48) `GameCharacter(int hp = 10, int a = 10);` // FIXME
- (18, 33) `void move(int x, int y);` // FIXME
- (19, 33) `void move(int distance);` // FIXME - move by distance in both x and y directions
- (49, 42) `bool fight(GameCharacter &enemy);` // FIXME
- (51, 39) `int receiveDamage(int points);` // FIXME
- (111, 16) // FIXME a fight should be allowed only if the hero is next to the monster
- (128, 8) // TODO if the hero has a weapon print its strength
- (184, 8) // TODO instantiate a static "hero" object of type GameCharacter with default parameters
- (192, 8) // TODO instantiate a static "sword" object of type Weapon with strenght=10 and no magic
- (193, 8) // TODO give the weapon to the hero
- (204, 25) `renderHUD(hero);` // FIXME

The status bar at the bottom indicates the current context is `class_exercise [D]`.

Classe Date

- Implementare un Functor che rispetti i requisiti necessari alla multimappa.
- `bool operator()(const T& lhs, const T& rhs) const;`
true if `lhs < rhs`, false otherwise.

Main

- Dichiarare multimappa con Functor (come tipo).
- Inserire elementi come pair, usando `make_pair` o costruttore di `pair`:

```
template< class T1, class T2 >  
std::pair<T1,T2> make_pair( T1 t, T2 u );
```

```
pair( const T1& x, const T2& y );
```

- Usare algoritmo `find` per determinare se in una data è presente dell'attività:

```
iterator find( const Key& key );
```

Iterator to an element with key equivalent to `key`. If no such element is found, past-the-end (see `end()`) iterator is returned.



Main

- Usare algoritmo `equal_range` per trovare tutti gli elementi associati ad una certa chiave:

```
std::pair<iterator, iterator>  
equal_range( const Key& key );
```

Returns a range containing all elements with the given key in the container. The range is defined by two iterators, one pointing to the first element that is not less than key and another pointing to the first element greater than key.



Main

- Iterare su tutti gli elementi della mappa e stamparli.

