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

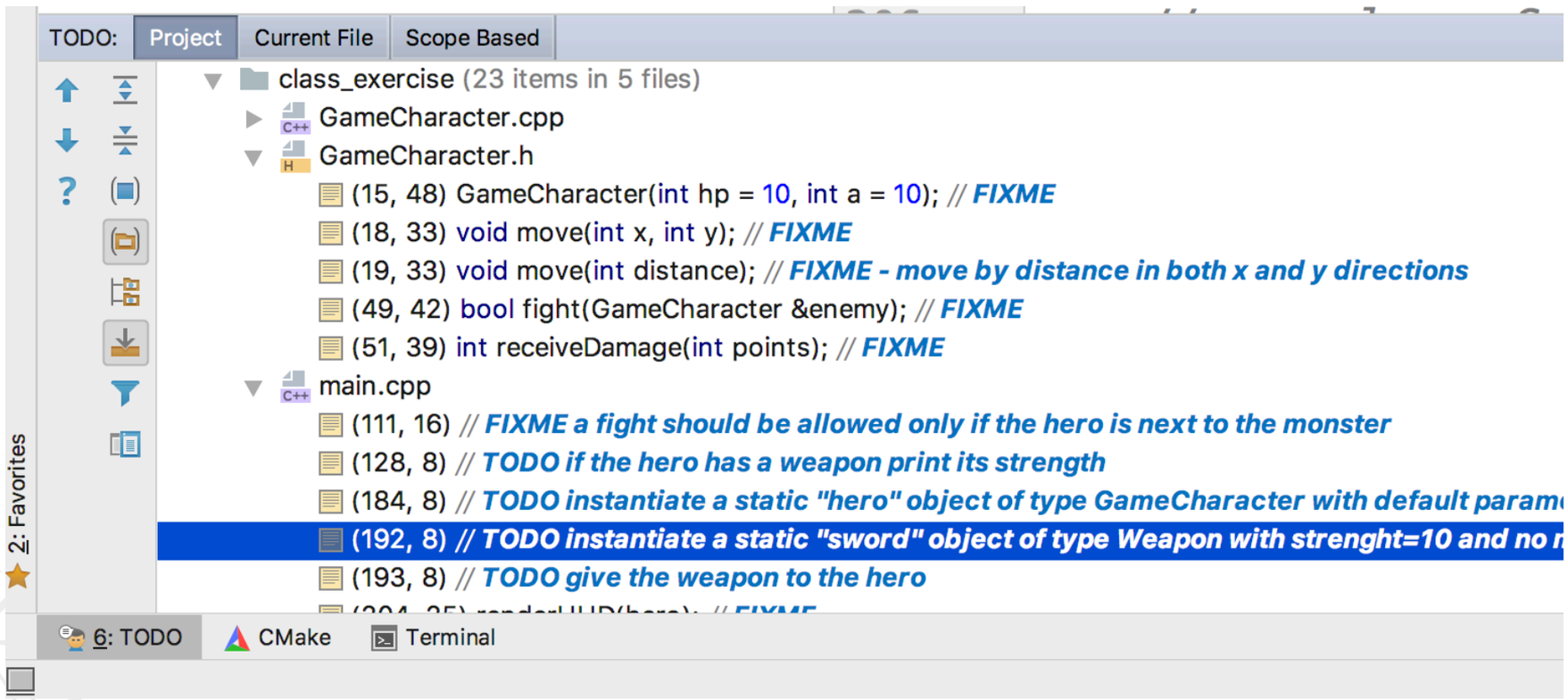
- In questa esercitazione andremo ad aggiungere un functor per comparare date in: `Date.cpp/h`

- Modificheremo il codice di main.cpp per istanziare una multimappa che associa date a task.

Dove modificare il codice

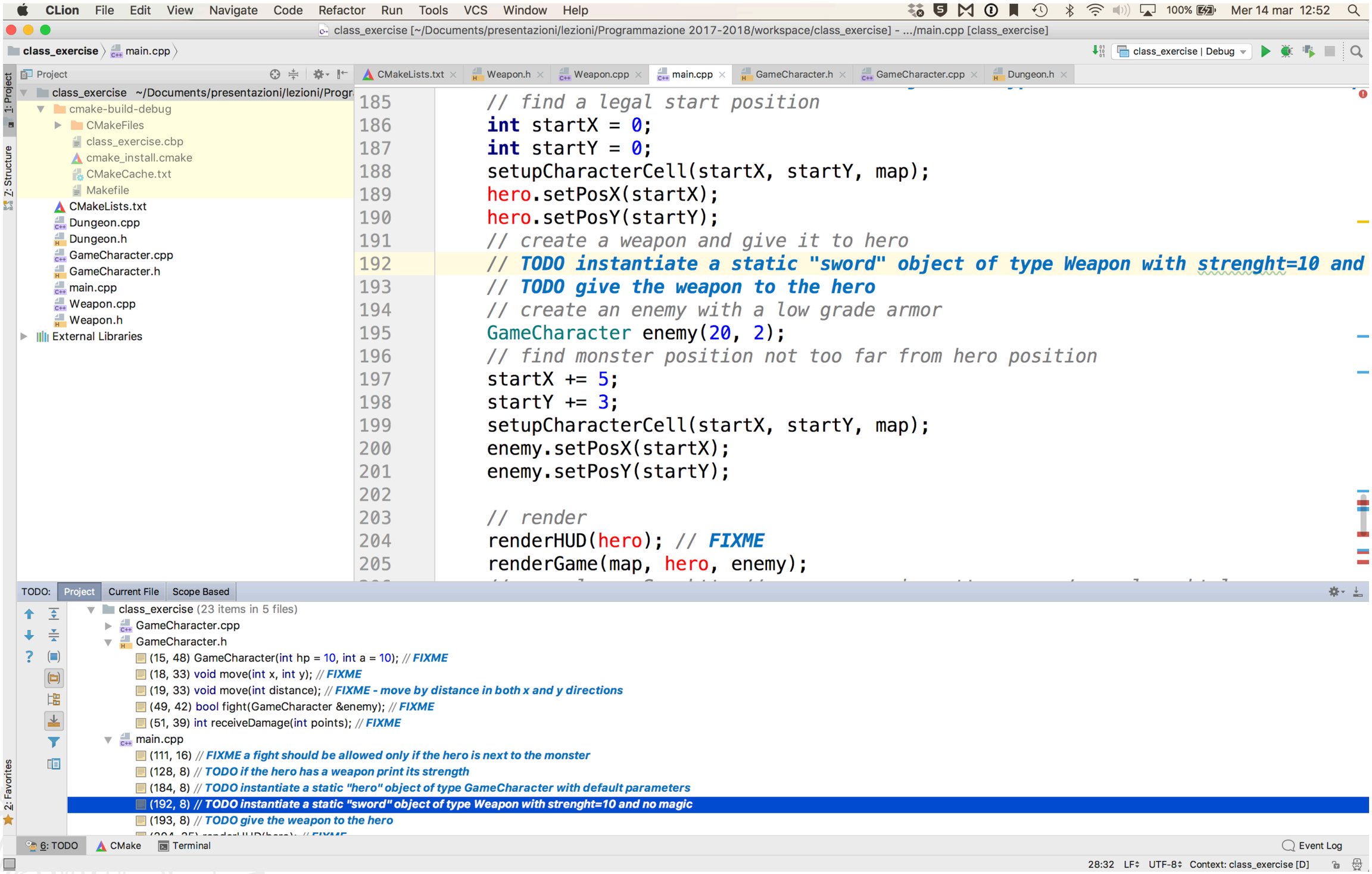
- Le indicazioni precise sul codice da modificare sono fornite come commenti indicati con `TODO` e `FIXME`

- Per vedere tutti questi commenti selezionare la finestra TODO di CLion



UNIVERSITÀ DEGLI STUDI FIRENZE

Dove modificare il codice



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.