Esercizi di Programmazione ad Oggetti

Lista n. 3

Esercizio 1

Definire un template di classe contenitore `Dizionario<T>` i cui oggetti rappresentano una collezione di coppie (chiave,valore) dove la chiave è una stringa ed il valore è di tipo `T`. Ad una certa stringa può essere associato un solo valore di `T`. Si tratta quindi di definire un template di classe analogo al contenitore STL `map<string,T>`. Dovranno essere disponibili le seguenti funzionalità:

- inserimento: `bool insert(string, const T&)`
- rimozione: `bool erase(string)`
- ricerca per chiave: `T* findValue(string)`
- ricerca per valore: `vector<string> findKey(const T&)`
- overloading degli operatori di indicizzazione e output

Esercizio 2

```
class Z {
public:
  ...
};

template <class T1, class T2=Z>
class C {
public:
  T1 x;
  T2* p;
};

template<class T1,class T2>
void fun(C<T1,T2>* q) {
  ++(q->p);
  if(true == false) cout << ++(q->x);
  else cout << q->p;
  (q->x)++;
  if(*(q->p) == q->x) *(q->p) = q->x;
  T1* ptr = &(q->x);
  T2 t2 = q->x;
}

main(){
  C<Z> c1; fun(&c1); C<int> c2; fun(&c2);
}
```

Si considerino le precedenti definizioni. Fornire una dichiarazione (non è richiesta la definizione) dei membri pubblici della classe `Z` nel **minor numero possibile** in modo tale che la compilazione del precedente `main()` non produca errori. **Attenzione:** ogni dichiarazione in `Z` non necessaria per la corretta compilazione del `main()` sarà penalizzata.

```
class Z {
public:
  int operator++() {} // 1
  Z& operator++(int) {} // 2
  bool operator==(const Z& x) const {} // 3
  Z(int x=0) {} // 4
```

```
};

template<class T1,class T2>
void fun(C<T1,T2>* q) {
  ++(q->p); // nulla
  if(true == false) cout << ++(q->x); // 1
  else cout << q->p;  // nulla
  (q->x)++; // 2
  if(*(q->p) == q->x) // 3
    *(q->p) = q->x; // nulla
  T1* ptr = &(q->x); // nulla
  T2 t2 = q->x; // 4 conversione T1 => T2
}
```