Laurea in Informatica – Programmazione ad Oggetti – Appello d'Esame 15/02/2024

Nome..... Cognome..... Matricola.....

Esercizio Funzione.

Definire un template di funzione

`template <class T> list<const iostream*> compare(vector<ostream*>&, vector<const T*>&)`

con il seguente comportamento: in ogni invocazione `compare(v,w)`,

1. se `v` e `w` non contengono lo stesso numero di elementi allora viene sollevata una eccezione di tipo `string` che rappresenta la stringa vuota;
2. se `v` e `w` contengono lo stesso numero di elementi allora per ogni posizione `i` dentro i bounds dei due vettori `v` e `w`:
   (a) se `*v[i]` è un `fstream` ed è dello stesso tipo di `*w[i]` allora: (i) il puntatore `v[i]` viene inserito nella lista che la funzione deve ritornare; (ii) i puntatori `v[i]` e `w[i]` vengono rimossi dai vettori che li contengono;
   (b) se `*w[i]` è uno `stringstream` in stato good e `*v[i]` e `*w[i]` sono di tipo diverso allora il puntatore `w[i]` viene inserito nella lista che la funzione deve ritornare.

```
TEMPLATE <CLASS T> LIST<CONST IOSTREAM*>
    ... COMPARE(VECTOR<OSTREAM*>& V, VECTOR<CONST T*>& W){
LIST<CONST IOSTREAM*> RET;
IF(V.SIZE() != W.SIZE()) THROW STRING("");
ELSE{ FOR(INT i=0; i<V.SIZE(); ++i){
   IF( DYNAMIC_CAST<FSTREAM*>(*V[i]) && TYPEID(*V[i])==TYPEID(*W[i])){
      RET.PUSHBACK(DYNAMIC_CAST<CONST T*>(*V[i]));

      ERASE/DELETE (*V[i])/(*W[i]); V.ERASE(V.BEGIN()+i);
   }

IF(DYNAMIC_CAST<STRINGSTREAM>(CONST_CAST<T>(*W[i])) &&
  *W[i]->GOOD && TYPEID(*V[i]) != (*W[i])) {
     RET.PUSHBACK(DYNAMIC_CAST<CONST IOSTREAM*>(*W[i]));
}

SE AUTO IT = LIST.BEGIN() .....

ALLORA IT = LISTA.ERASE(IT)

SE HO UN CONTATORE NORMALE, COME IN QUESTO

CASO PARTICOLARE → V.ERASE(V.BEGIN()+i);

TUTTO QUESTO SOLO SE USO UN PUNTATORE
DI COPIA, ALTRIMENTI DOVREI UTILIZZARE ANCHE
DELETE
```

ERASE → TRONCO CODE

① QCHECKBOX * QCB = DYNAMIC_CAST
<QCHECKBOX*>(*IT);
DELETE QCB;
IT = V.ERASE(IT); ← OSA QCB PER POI FARE DELETE

② IF(DYNAMIC_CAST<QCHECKBOX*>(...
....
IT = V.ERASE(IT) ≠ NO! (DELETE)

CASI CON CONTATORE

① FOR(INT I=0; I<V.SIZE(); I++)...
....
V.ERASE(V.BEGIN()+I);

② FOR....
QCHECKBOX * QCB = DYN.CAST<QWIDGET*>(&V[I]);
V.ERASE(V.BEGIN()+I);

DELETE QCB:

ITERATORI → NAI CONST/
NON CONST

FUNZIONI CON VECTOR<CONST IOSTREAM*>
ES. SOLUZIONI PROPOSTE

① FSTREAM * F = DYNAMIC_CAST<FSTREAM*>
CONST_CAST<IOSTREAM**>(&IT);

② CONST FSTREAM * F =
DYNAMIC_CAST<CONST FSTREAM*>(IT);

③ IF(DYN_CAST<CONST FSTREAM*>(IT)
↑
COPIARE
& USO?

④ IF (DYN_CAST<FSTREAM**>(CONST_CAST
<IOSTREAM**>(&IT))
↑ COPIARE & USO?

↓

MORALE = USO SOLITARIO → * NOR LE
ANGOLARE
<...>

DI SOLITO USO ANCHE & PER LO

TOMBE → .... <(OSTREAM &> (*USI[*IT]);

TRANNE NEL CASO

FUNZIONE(CONST IOS& I);

↑

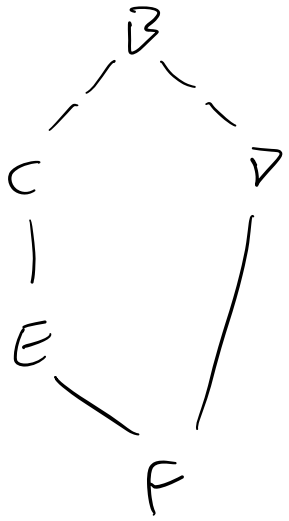
. . . . CASO NON COMUNISSIMO...

ES. FSTREAM & F = DYNAMIC_CAST<FSTREAM&>(I);

Esercizio Cosa Stampa

```
class B {
public:
  int x;
  B(int z=1): x(z) {}
  virtual void f() const {cout << x << " B::f() ";}
};

class D: virtual public B {
public:
  virtual void f() const {cout << "D::f() ";}
};


class F: public E, public D {
public:
  F(): B(3) {}
  virtual void f() const {cout << x << " F::f() ";}
  virtual void g() const {cout << "F::g() ";}
};

void Fun(const vector<B*>& v) {
  auto it1 = v.begin();
  vector<B*>::const_iterator it2;
  C* q;
  for(int i=1 ; it1 != v.end(); ++it1, ++i) {
    std::cout << "#" << i << " ";
    (*it1)->f();
    it2 = it1 + 1;
    if(it2 != v.end() && typeid(**it1) == typeid(**it2)) (*it2)->f();
    q = dynamic_cast<C*>(*it1);
    if(q) {static_cast<C*>(q)->g(); q->h();}
    cout << endl;
  }
}

int main() {
  B b; C c; D d; E e; F f;
  vector<B*> v = {  &d, &d, &e, &e, &b, &b, &f, &f, &e, &f, &c, &c };
  Fun(v);
}
```

```
class C: virtual public B {
public:
  virtual void g() const {cout << "C::g() ";}
  virtual void h() const {cout << "C::h() ";}
};


class E: public C {
public:
  virtual void f() const {cout << "E::f() ";}
  virtual void h() const {cout << "E::h() ";}
};
```



Le precedenti definizioni compilano correttamente ed il `main` esegue senza undefined behavior o errori run-time. Scrivere nell'apposito spazio relativo alla riga `#i` le stampe prodotte in output dall'iterazione `i`-esima del ciclo for della funzione `fun`, scrivendo **NESSUNA STAMPA** se in una iterazione non ci fossero stampe prodotte in output.

#1 D::f() D::f()

#2 D::f()

#3 E::f() E::f() C::g() E::h()

#4 E::f() C::g() E::h()

#5 7D::f() 7B::f()

#6 7B::f()

#7 3F::f() 3F::f() F::g() E::h()

#8 3F::f() F::g() E::h()

#9 E::f() C::g() E::h()

#10 3F::f() F::g() E::h()

#11 7B::f() 7D::f C::g() C::h()

#12 7B::f() C::g() C::h()