Metodologie di Programmazione
(prova in itinere)

Corso di Laurea in "Informatica"

25 novembre 2014

Note

1. Su tutti i fogli contenenti le soluzioni indicare, IN STAMPATELLO, la data della prova ed il proprio cognome, nome e numero di matricola.
2. Non è consentita la consultazione di alcunché.
3. L'orario di consegna scritto alla lavagna è tassativo.
4. Il testo del compito va consegnato insieme a tutti i fogli; marcare in modo evidente i fogli di brutta (che vanno consegnati insieme ai fogli contenenti le soluzioni).

Esercizi

1. Mostrare il processo di risoluzione dell'overloading per le seguenti chiamate di funzione. Per ogni chiamata, indicare l'insieme delle funzioni candidate, l'insieme delle funzioni utilizzabili e, se esiste, la migliore funzione utilizzabile.

```
namespace N {
  struct C {
    C(int);                // funzione #1
    C(const C&);           // funzione #2
  };

  void f(double d);        // funzione #3
  void f(const C& c);      // funzione #4
  void g(int i, double d); // funzione #5
  void g(int i, int j);    // funzione #6
  void h(C* pc);           // funzione #7

} // namespace N


void f(char);              // funzione #8

int h(const char* s = 0);  // funzione #9
int h(const N::C* pc);     // funzione #10


int main() {
  N::C c(5);    // chiamata A

  f(5);         // chiamata B
  f(c);         // chiamata C
  N::f('a');    // chiamata D

  g(5, 3.7);    // chiamata E
  N::g(2.3, 5); // chiamata F
  N::g(5, 2.3); // chiamata G

  h(&c);        // chiamata H
}
```

2. Scrivere un esempio, minimale ma completo, di codice che causa un errore di compilazione a causa dell'assenza delle guardie contro l'inclusione ripetuta. Spiegare brevemente il motivo dell'errore.

3. Si supponga che il metodo booleano `check_inv()` restituisca il valore `true` se e solo se l'oggetto sul quale è invocato soddisfa la proprietà invariante della classe `C`. Indicare, nel codice seguente, come ed in quali punti è opportuno inserire il controllo di tale invariante allo scopo di semplificare le attività di testing e debugging della classe (si assuma che il codice non contenga nessuna occorrenza di `const_cast`).

```
class C {
public:
  bool check_inv() const {
    // Codice che implementa il controllo di invariante.
  }

  C(int a, int b) {
    // Implementazione del costruttore.
  }

  void foo(C& y) {
    // Codice che implementa il metodo foo.
  }

  void bar(const C& y) {
    // Codice che implementa il metodo bar.
  }

  void ying(const C& y) {
    // Codice che implementa il metodo ying.
  }

  void yang(const C& y) const {
    // Codice che implementa il metodo yang.
  }

  ~C() {
    // Codice che implementa il distruttore.
  }

  static void zen(int i, double d) {
    // Codice che implementa il metodo zen.
  }

  // ... altro codice ...

}; // class C
```

4. Si forniscano il prototipo e l'implementazione della funzione generica `transform` i cui parametri specificano: (a) due sequenze di ingresso (non necessariamente dello stesso tipo); (b) una sequenza di uscita (potenzialmente di un tipo ancora diverso); (c) una funzione binaria applicabile a coppie di elementi in ingresso (il primo elemento preso dalla prima sequenza, il secondo elemento preso dalla seconda sequenza) e che restituisce un elemento assegnabile alla sequenza di uscita.

   La funzione `transform` applica la funzione binaria agli elementi in posizione corrispondente nelle due sequenze di ingresso, assegnando il risultato ad elementi consecutivi della sequenza di uscita, ed arrestandosi quando si raggiunge la fine di una qualunque delle due sequenze di ingresso (Nota Bene: le due sequenze in ingresso hanno lunghezze arbitrarie).

   Utilizzando la funzione suddetta, scrivere una funzione che, date due liste di numeri floating point $[a_1, \ldots, a_n]$ e $[b_1, \ldots, b_n]$, aggiunge in coda ad una terza lista le medie aritmetiche $\left[\frac{a_1+b_1}{2}, \ldots, \frac{a_n+b_n}{2}\right]$.

5. Il codice seguente non si comporta bene, dal punto di vista della gestione delle risorse, in presenza di eccezioni. Individuare almeno un problema, indicando la sequenza di operazioni che porta alla sua occorrenza. Fornire quindi due soluzioni alternative:

   - la prima basata sull'utilizzo dei blocchi `try` ... `catch`;
   - la seconda senza utilizzare alcun blocco `try` ... `catch`.

   Per entrambe le soluzioni si richiede che l'ordine di allocazione e deallocazione delle risorse di tipo `A` sia lo stesso codificato nel codice originale (per il caso di esecuzione senza eccezioni).

```
void foo() {
  A* a1 = new A(1);
  A* a2 = new A(2);
  job1(a1, a2)
  delete a2;
  A* a3 = new A(3);
  job2(a1, a3)
  job3(a3)
  delete a3;
  delete a1;
}
```