Metodologie di Programmazione

Corso di Laurea in "Informatica"

22 febbraio 2005

Note

1. Su tutti i fogli contenenti le soluzioni indicare, IN STAMPATELLO, la data dell'appello ed il proprio cognome, nome e numero di matricola
2. Leggere attentamente i quesiti e fornire risposte sintetiche: di solito, una risposta troppo verbosa sta ad indicare che si stanno fornendo particolari non richiesti.
3. Non è consentita la consultazione di alcunché.
4. L'orario di consegna scritto alla lavagna è tassativo.
5. Il testo del compito va consegnato insieme ai fogli con le soluzioni.

1. Per ognuna delle seguenti chiamate, dire se la risoluzione dell'overloading va a buon fine e, nel caso, indicare quale funzione viene effettivamente invocata.

```
void f(const char* s);    // funzione #1
template <typename T>
void f(T t);              // funzione #2

template <typename T>
void f(T t1, T t2);       // funzione #3
template <typename T, typename U>
void f(T t, U u);         // funzione #4
template <typename T>
void f(T* pt, T t);       // funzione #5
template <typename T, typename U>
void f(T* pt, U u);       // funzione #6

template <typename T>
void g(T t, double d);    // funzione #7
template <typename T>
void g(T t1, T t2);       // funzione #8

int test() {
  f('a');         // chiamata #1
  f("aaa");       // chiamata #2
  int i;
  f(i);           // chiamata #3
  f(i, i);        // chiamata #4
  f(i, &i);       // chiamata #5
  f(&i, i);       // chiamata #6
  double d;
  f(i, d);        // chiamata #7
  f(&d, i);       // chiamata #8
  long l;
  g(l, i);        // chiamata #9
  g(l, l) ;       // chiamata #10
  g(l, d) ;       // chiamata #11
  g(d, d) ;       // chiamata #12
}
```

2. Indicare l'output prodotto dal seguente programma.

```
#include <iostream>

class ZooAnimal {
public:
  ZooAnimal() {
    std::cout << "Constructor ZooAnimal" << std::endl;
  }
  virtual void print() {
    std::cout << "ZooAnimal::print" << std::endl;
  }
  virtual ~ZooAnimal() {}
};

class Bear : virtual public ZooAnimal {
public:
```

```
  Bear() {
    std::cout << "Constructor Bear" << std::endl;
  }
  void print() {
    std::cout << "Bear::print" << std::endl;
  }
  virtual ~Bear() {}
};

class Raccoon : virtual public ZooAnimal {
public:
  Raccoon() {
    std::cout << "Constructor Raccoon" << std::endl;
  }
  virtual ~Raccoon() {}
};

class Endangered {
public:
  Endangered() {
    std::cout << "Constructor Endangered" << std::endl;
  }
  void print() {
    std::cout << "Endangered::print" << std::endl;
  }
  virtual ~Endangered() {}
};

class Panda : public Endangered, public Bear, public Raccoon {
public:
  Panda() {
    std::cout << "Constructor Panda" << std::endl;
  }
  void print() {
    std::cout << "Panda::print" << std::endl;
  }
  virtual ~Panda() {}
};

int main() {
  Panda ying_yang;
  ying_yang.print();

  Bear b = ying_yang;
  b.print();

  ZooAnimal* pz = &ying_yang;
  pz->print();

  Endangered& re = ying_yang;
  re.print();

  return 0;
}
```

3. Individuare il problema che si potrebbe presentare nel codice seguente (si noti che si tratta di quattro file sorgente distinti). Fornire un'implementazione alternativa in grado di porvi soluzione.

```
/************ Inizio file Costante.hh ************/
class Costante {
public:
  static double pi;
  static double e;
  // ...
};

/************ Inizio file Costante.cc ************/
#include "Costante.hh"

Costante::pi = 3.1415;
Costante::e = 2.72;
// ...

/************ Inizio file Angolo.hh  ************/
class Angolo {
public:
  static double piatto;
  static double retto;
  static double giro;
  // ...
};

/************ Inizio file Angolo.cc  ************/
#include "Angolo.hh"
#include "Costante.hh"

Angolo::retto  = Costante::pi / 2;
Angolo::piatto = Costante::pi;
Angolo::giro   = Costante::pi * 2;
// ...
```

4. Fornire il prototipo e l'implementazione della funzione templatica `average`, che prende come unico argomento un (non meglio identificato) contenitore sequenziale della libreria standard e restituisce come risultato la media aritmetica degli elementi contenuti. Elencare i requisiti imposti da tale funzione sul tipo degli elementi del contenitore.

   Riscrivere la funzione `average` facendo in modo che prenda come argomenti due iteratori che identificano la sequenza in ingresso. Per quale motivo questa seconda versione è più generale della precedente?

5. Elencare le categorie di iteratori utilizzate nella libreria standard, mostrando la gerarchia di ereditarietà dei corrispondenti tag. Per ogni categoria, identificare un esempio di applicazione per il quale è opportuno utilizzare un iteratore della categoria suddetta, indicando almeno un motivo per il quale l'utilizzo di iteratori di un'altra categoria risulta inadeguato.

6. La classe templatica `Set` è intesa rappresentare un insieme di elementi di tipo `T`. L'implementazione della classe si basa sulla manipolazione di liste ordinate (senza duplicati). L'interfaccia della classe presenta numerosi problemi. Cercare di individuarne il maggior numero ed indicare come possono essere risolti (riscrivendo l'interfaccia).

```
template <typename T>
class Set : public std::list<T> {
public:
  // Costruisce l'insieme vuoto.
  Set();
  // Costruisce il singoletto che contiene t.
  Set(T t);
  Set(Set y);
  void operator=(Set y);
  virtual ~Set();

  unsigned int size();
  bool is_empty();
  bool contains(Set y);

  T& min();
  void pop_min();
  void insert(T z);
  void union_assign(Set y);
  void intersection_assign(Set y);

  void swap(Set y);
  std::ostream operator<<(std::ostream os);

private:
  // ...
};
```

7. La seguente implementazione della classe `Container`, indipendentemente da quali siano le funzionalità che tale tipo di dato deve fornire, è errata. Individuare i problemi, indicando la sequenza di operazioni che porta alla loro occorrenza. Modificare la classe per eliminare tutti i problemi identificati. Discutere (brevemente) se la soluzione proposta è corretta anche in presenza di eccezioni.

```
template <typename T>
class Container {
public:
  Container(int size)
    : sz(size), ps(new T[sz]) {
  }

  ~Container() {
    delete ps;
  }

private:
  T* ps;
  unsigned int sz;
};
```