Metodologie di Programmazione

Corso di Laurea in "Informatica"

7 febbraio 2024

1. (Risoluzione overloading) 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; la migliore funzione utilizzabile (se esiste).

```
struct D;

struct C {
  C() {}
  C(const D&) {}
};

struct D {
  D() {}
  D(const C&) {}
};

void f(double d);                       // funzione #1
void f(int i, C c = C());               // funzione #2

void g(C c, D d);                       // funzione #3
void g(D d, C c = C());                 // funzione #4

void test(C c, D d) {
  f(3.2);          // chiamata A
  f('a');          // chiamata B
  f('a', c);       // chiamata C

  g(c, d);         // chiamata D
  g(d, c);         // chiamata E
  g(c);            // chiamata F
}
```

2. (Conversioni implicite) Scrivere la dichiarazione di una funzione avente 4 parametri e una chiamata della funzione stessa che comporti le seguenti conversioni implicite sui 4 argomenti (nell'ordine): conversione di qualificazione const, promozione, conversione standard, conversione definita dall'utente.

3. (Funzione generica) Scrivere l'implementazione dell'algoritmo generico `min_element`, che prende come input una sequenza ed un criterio di ordinamento (un predicato binario) e restituisce l'iteratore all'elemento della sequenza di valore minimo (secondo tale criterio di ordinamento); l'algoritmo si comporta in modo standard nel caso di sequenza vuota. Elencare in modo esaustivo i requisiti imposti dall'implementazione sui parametri di tipo e sugli argomenti della funzione. In particolare, individuare le categorie di iteratori che *non* possono essere utilizzate per istanziare il template, motivando la risposta.

4. (Gestione risorse) Il codice seguente presenta alcuni problemi relativi alla gestione delle risorse. Evidenziare questi problemi, differenziando tra: (a) errori che si verificano in assenza di eccezioni; (b) errori che si verificano in presenza di eccezioni.

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

   Fornire una soluzione basata sull'idioma RAII che si comporti correttamente sia in assenza che in presenza di eccezioni.

5. (Domande a risposta aperta)

   (a) Fornire un esempio di violazione della ODR (One Definition Rule) che sia causato dalla assenza delle guardie contro l'inclusione ripetuta degli header file.

   (b) Spiegare brevemente la differenza tra `static_cast`, `const_cast` e `dynamic_cast`, fornendo per ognuno un semplice esempio di utilizzo.

   (c) Con riferimento alla programmazione per contratto, spiegare la differenza tra contratti narrow e contratti wide; fornire un semplice esempio per ognuno dei due casi.

6. (Progettazione object-oriented) Uno strumento software utilizza codice come il seguente allo scopo di gestire la produzione automatica di documentazione secondo diversi formati di stampa.

```
// Manual_Generator.hh
struct Manual_Generator {
  virtual void put(const std::string& s) = 0;
  virtual void set_boldface() = 0;
  virtual void reset_boldface() = 0;
  virtual ~Manual_Generator() = default;
};

// HTML_Generator.hh
#include "Manual_Generator.hh"
struct HTML_Generator : public Manual_Generator {
  void put(const std::string& s) override;
  void set_boldface() override;
  void reset_boldface() override;
  void hyperlink(const std::string& uri, const std::string& text);
};

// ASCII_Generator.hh
#include "Manual_Generator.hh"
struct ASCII_Generator : public Manual_Generator {
  void put(const std::string& s) override;
  void set_boldface() override;
  void reset_boldface() override;
  void page_break();
};
```

All'interno del codice utente, le classi suddette vengono usate nel modo seguente

```
#include "Manual_Generator.hh"
#include "HTML_Generator.hh"
#include "ASCII_Generator.hh"
void user_code(Manual_Generator* mg_p) {
  HTML_Generator* html_p = dynamic_cast<HTML_Generator*>(mg_p);
  if (html_p)
    html_p->hyperlink("http://www.unipr.it", "UNIPR");
  else
    mg_p->put("PPL (http://www.cs.unipr.it/ppl)");
  ASCII_Generator* ascii_p = dynamic_cast<ASCII_Generator*>(mg_p);
  if (ascii_p)
    ascii_p->page_break();
  else
    mg_p->put("<HR>"); // Simulo il cambio pagina in HTML.
}
```

Quali principi della programmazione orientata agli oggetti sono violati dal codice precedente? Fornire una soluzione alternativa che rispetti i principi in questione, mostrando le modifiche da apportare al codice (sia il codice delle classi, che il codice della funzione `user_code`.