Corso Programmazione ad Oggetti
Corso di Laurea in Informatica
Esercitazione di Laboratorio del 27/05/2016

Esercizio 1

In un'applicazione in cui si richiede di manipolare i dati anagrafici di un gruppo di persone si fa uso delle classi: **DatiAnagrafici** e **VettoreDatiAnagrafici**. La prima classe consente di memorizzare le informazioni relative ad una persona, mentre la seconda implementa un array dinamico di oggetti di tipo **DatiAnagrafici**. Seguono le intestazioni delle classi:

```
class DatiAnagrafici {
      friend ostream& operator<<(ostream& out, const DatiAnagrafici& d);
private:
      string nome;
      string cognome;
      string indirizzo;
      string codiceFiscale;
      string dataDiNascita;
public:
      DatiAnagrafici();
      DatiAnagrafici(const string &n, const string &c, const string &ind,
const string &cf, const string &dN);
      DatiAnagrafici(const DatiAnagrafici& d);
      bool operator==(const DatiAnagrafici& d) const;
      const string& getNome() const;
      const string& getCognome() const;
      const string& getIndirizzo() const;
      const string& getCodiceFiscale() const;
      const string& getDataDiNascita() const;
      void setNome(const string& s);
      void setCognome(const string& s);
      void setIndirizzo(const string& s);
      void setCodiceFiscale(const string& s);
      void setDataDiNascita(const string& s);
};

class VettoreDatiAnagrafici
{
friend ostream& operator<<(ostream&, const VettoreDatiAnagrafici& d);
private:
      DatiAnagrafici* dati;
      int lunghezza;
public:
      VettoreDatiAnagrafici(int n);
      VettoreDatiAnagrafici(const VettoreDatiAnagrafici& d);
      VettoreDatiAnagrafici& operator=(const VettoreDatiAnagrafici& d);
      DatiAnagrafici& operator[](int i);
      int getLunghezza();
      ~VettoreDatiAnagrafici();
};
```

Si implementino i metodi membro delle due classi.

Nota: E' possibile aggiungere, qualora si ritenesse necessario, ulteriori membri pubblici o privati alle classi.

Corso Programmazione ad Oggetti
Corso di Laurea in Informatica
Esercitazione di Laboratorio del 27/05/2016

Esercizio 2

Data la classe **DatiAnagrafici** descritta nell'esercizio 1, si utilizzi l'ereditarietà per progettarne una estensione dal nome **DatiStudenteUniversitario**. La nuova classe deve essere in grado di rappresentare/manipolare, oltre alle informazioni già presenti nella classe **DatiAnagrafici**, ulteriori informazioni specifiche degli studenti universitari, quali la matricola (da rappresentare con un intero), il corso di laurea (da rappresentare con una stringa) e l'anno di corso (da rappresentare con un intero).

Esercizio 3

Rispondere ai seguenti quesiti.

Supponendo che la **classePippo** sia dotata di un metodo **metodo1()**, quali tra queste espressioni sono corrette (ed equivalenti) relativamente alla seguente dichiarazione : **classePippo *pObj;**

```
a) pObj->metodo1();      b) (*pObj)->medoto1();
c) (&pObj).metodo1();    d) (*pObj).metodo1();
```

Data la seguente dichiarazione di classe:

```
class Insieme {
    private:
            int* insieme;
            int dim;
    public:
            Insieme(int s){insieme = new int[dim];}
};
```

Quale tra le seguenti rappresenta una corretta implementazione del distruttore della classe **Insieme**?

```
a) ~Insieme(){delete insieme;}    b) ~Insieme(){delete [] insieme;}
c) ~Insieme(){delete this;}       d) ~Insieme(){}
```

Si considerino le seguenti dichiarazioni:

```
class A {
    public:
        virtual int metodo(){return 1;}
};
class B : public A {
    public:
        virtual int metodo(){return 2;}
};
class C : public B {

};
```

Quale stampa si ottiene dal seguente main?

```
int main(){
    A* a = new A();
    A* b = new B();
    A* c = new C();
    cout<<a->metodo()<<b->metodo()<<c->metodo();
}
```

```
a)111                b) 121
c)122                d) 222
```

Corso Programmazione ad Oggetti
Corso di Laurea in Informatica
Esercitazione di Laboratorio del 27/05/2016

Si considerino le seguenti dichiarazioni:

```
class A {
    public:
        void a1();
    protected:
        void a2();
    private:
        void a3();
};
```

```
class B: private A {
    public:
        void b1();
    protected:
        void b2();
    private:
        void b3();
};
```

Qual è l'insieme contenente tutte e sole istruzioni **CORRETTE**?

```
class C: public B {
    public:
        void c1() {
            a1(); /*n.1*/          a2(); /*n.2*/
            a3(); /*n.3*/          b1(); /*n.4*/
            b2(); /*n.5*/          b3(); /*n.6*/
        }
};
```

a) {1, 2, 3, 4, 5, 6}
b) {1, 4}
c) {1, 4, 5}
d) {4, 5}
e) {4, 5, 6}

Si considerino le seguenti dichiarazioni:

```
class A {
    public:
        void a1();
    protected:
        void a2();
    private:
        void a3();
};
```

```
class B: protected A {
    public:
        void b1();
    protected:
        void b2();
    private:
        void b3();
};
```

Qual è l'insieme contenente tutte e sole istruzioni **CORRETTE**?

```
class C: protected B {
    public:
        void c1() {
            a1(); /*n.1*/          a2(); /*n.2*/
            a3(); /*n.3*/          b1(); /*n.4*/
            b2(); /*n.5*/          b3(); /*n.6*/
        }
};
```

a) {1, 2, 3, 4, 5, 6}
b) {1, 2}
c) {1, 2, 4, 5}
d) {1, 4}
e) {1, 2, 4, 5, 6}