Esercizi di Programmazione ad Oggetti

Lista n. 1

Esercizio 1

Definire una classe `IntMod` i cui oggetti rappresentano numeri interi modulo un dato intero n, che deve essere dichiarato come campo dati statico.
Definire metodi statici di `set_modulo()` e `get_modulo()` per tale campo dati statico.
Devono essere disponibili gli operatori di somma e moltiplicazione tra oggetti di `IntMod`.
Definire inoltre opportuni convertitori di tipo affiché questa classe sia liberamente usabile assieme al tipo primitivo `int` e valga la seguente condizione:
quando in una espressione compaiono interi e oggetti di `IntMod` il tipo dell'espressione dovrà essere intero.

Scrivere infine un programma d'esempio che utilizza tutti i metodi della classe.

Esercizio 2

Il seguente programma compila. Quali stampe produce la sua esecuzione?

```
#include<iostream>
using std::cout;

class A {
private:
  int x;
public:
  A(int k = 5): x(k) {cout << k <<  " A01 ";}
  A(const A& a): x(a.x) {cout << "Ac ";}
  A g() const {return *this;}
};

class B {
private:
  A ar[2];
  static A a;
public:
  B() {ar[1] = A(7); cout << "B0 ";}
  B(const B& b) {cout << "Bc ";}
};
A B::a = A(9);

A Fun(A* p, const A& a, B b) {
  *p = a;
  a.g();
  return *p;
};

main() {
  cout << ``ZERO\n'';
  A a1; cout << "UNO\n";
  A a2(3); cout << "DUE\n";
  A* p = &a1; cout << "TRE\n";
  B b; cout << "QUATTRO\n";

  a1 = Fun(p,a2,b); cout << "CINQUE\n";

  A a3 = Fun(&a1,*p,b); cout << "SEI";
}
```

Esercizio 3

Perché il seguente programma non compila? Modificare o eliminare una e soltanto una delle righe 1-8 in modo che il programmi compili.

```
class C {
public:                          // 1
  int *const p;                  // 2
  C(int a=0): p(new int(a))      // 3
  { }                            // 4
};

main() {
  C x(3);                        // 5
  C y;                           // 6
  x=y;                           // 7
  C z(y);                        // 8
}
```

Esercizio 4

Il seguente programma compila ed esegue correttamente. Quale stampa di output provoca?

```
#include<iostream>
#include<string>
using std::string; using std::cout;

class C {
private:
  int d;
public:
  C(string s=""): d(s.size()) {}
  explicit C(int n): d(n) {}
  operator int() {return d;}
  C operator+(C x) {return C(d+x.d);}
};

main() {
  C a, b("pippo"), c(3);
  cout << a << ' ' << 1+b << ' ' << c+4 << ' ' << c+b;
}
```

Esercizio 5

Definire, separando interfaccia ed implementazione, una classe `Raz` i cui oggetti rappresentano un numero razionale $\frac{num}{den}$ (naturalmente, i numeri razionali hanno sempre un denominatore diverso da 0). La classe deve includere:

1. opportuni costruttori;
2. un metodo `Raz inverso()` con il seguente comportamento: se l'oggetto di invocazione rappresenta $\frac{n}{m}$ allora `inverso` ritorna un oggetto che rappresenta $\frac{m}{n}$;
3. un operatore esplicito di conversione al tipo primitivo `double`;
4. l'overloading come metodi interni degli operatori di somma e moltiplicazione;

5. l'overloading come metodo interno dell'operatore di incremento postfisso, che, naturalmente, dovrà incrementare di 1 il razionale di invocazione;
6. l'overloading dell'operatore di output su `ostream`;
7. un metodo statico `Raz uno()` che ritorna il razionale 1.

Definire un esempio di `main()` che usi tutti i metodi della classe.

Esercizio 6

Il seguente programma compila ed esegue correttamente. Quali stampe provoca in output?

```
#include<iostream>
#include<string>
using std::string; using std::cout;

class B {
public:
  string s;
  B(char x='a', char y='b') {s += x; s += y; cout << "B012 ";}
  B(const B& obj): s(obj.s) {cout << "Bc "; }
};

class C {
private:
  B t;
  B* p;
  B u;
public:
  string s;
  C(char x='c', B y = B('d')): u(y), s(y.s) {
    s += x;
    cout << s[s.size()-2] << " C012 ";
  }
};

B F(B x, C& y) {
  (x.s) += (y.s)[0];
  return x;
}

main() {
  B b('e'); cout << "UNO\n";
  C c1('f',b); cout << "DUE\n";
  C c2; cout << "TRE\n";
  b=F(b,c2); cout <<"QUATTRO\n";
  cout << b.s  << " CINQUE";
}
```

Esercizio 7

Si consideri il seguente programma.

```
#include<iostream>
using std::cout;

class C {
```

```
public:
  int x;
  C(int k=5): x(k) {};
  C* m(C& c) {
    if((c.x != 5) && (x==5)) return &c;
    return this;
  }
};

main() {
  C a, b(2), c(a);
  cout << (b.m(b))->x << ' ' << (a.m(a))->x << ' ' << (b.m(c))->x
       << ' ' << c.m(a) << ' ' << c.m(c);
}
```

Il seguente programma compila correttamente? Se sì, al sua esecuzione quali stampe provoca in output?

Esercizio 8

Definire, separando interfaccia ed implementazione, una classe `Data` i cui oggetti rappresentano una data con giorno della settimana (lun-mar-...-dom). La classe deve includere:

- opportuni costruttori
- metodi di selezione per ottenere giorno della settimana, giorno, mese, anno di una data
- l'overloading dell'operatore di output esternamente alla classe
- l'overloading dell'operatore di uguaglianza
- l'overloading dell'operatore relazionale < che ignori il giorno della settimana
- un metodo `aggiungi_uno()` che avanza di un giorno la data di invocazione. Esempi: `lun 21/10/2002 => mar 22/10/2002`; `gio 31/1/2002 => ven 1/2/2002`; `mar 31/12/2002 => mer 1/1/2003`. Ignorare gli anni bisestili

Esemplificare l'uso della classe e di tutti i suoi metodi tramite un esempio di `main()`.

Esercizi di Programmazione ad Oggetti

Lista n. 2

Esercizio 1

Si consideri il seguente programma.

```
class S {
public:
  string s;
  S(string t): s(t) {}
};
class N {
private:
  S x;
public:
  N* next;
  N(S t, N* p): x(t), next(p) {cout << "N2 ";}
  ~N() {if (next) delete next; cout << x.s + "~N ";}
};
class C {
  N* pointer;
public:
  C(): pointer(0) {}
  ~C() {delete pointer; cout << "~C ";}
  void F(string t1, string t2 = "pippo") {
    pointer = new N(S(t1),pointer); pointer = new N(t2,pointer);
  }
};
main(){
  C* p = new C; cout << "UNO\n";
  p->F("pluto","paperino"); p->F("topolino"); cout <<"DUE\n";
  delete p; cout <<"TRE\n";
}
```

Il programma compila correttamente? Se sì, quali stampe provoca in output?

Esercizio 2

Si consideri il seguente programma.

```
#include<iostream>
using std::cout;

class It {
  friend class C;
public:
  bool operator<(It i) {return index < i.index;}
  It operator++(int) { It t = *this; index++; return t; }
  It operator+(int k) {index = index + k; return *this; }
private:
  int index;
};
class C {
public:
  C(int k) {
    if (k>0) {dim=k; p = new int[k];}
```

```
    for(int i=0; i<k; i++) *(p+i)=i;
  }
  It begin() { It t; t.index = 0; return t; }
  It end() { It t; t.index = dim; return t; }
  int& operator[](It i) {return *(p + i.index);}
private:
  int* p;
  int dim;
};

main() {
  C c1(4), c2(8);
  for(It i = c1.begin(); i < c1.end(); i++) cout << c1[i] << ' ';
  cout << "UNO\n";
  It i = c2.begin();
  for(int n=0; i < c2.end(); ++n, i = i+n) cout << c2[i] << ' ';
  cout << "DUE\n";
}
```

Il programma compila correttamente? Se sì, quali stampe provoca in output?

Esercizio 3

Si considerino le seguenti dichiarazioni e definizioni:

```
class Nodo {
private:
  Nodo(string st="***", Nodo* s=0, Nodo* d=0): info(st), sx(s), dx(d) {}
  string info;
  Nodo* sx;
  Nodo* dx;
};
class Tree {
public:
  Tree(): radice(0) {}
  Tree(const Tree&); // dichiarazione costruttore di copia
private:
  Nodo* radice;
};
```

Quindi, gli oggetti della classe `Tree` rappresentano *alberi binari ricorsivamente definiti di stringhe*. Si ridefinisca il costruttore di copia di `Tree` in modo che esegua copie profonde. Scrivere esplicitamente eventuali dichiarazioni `friend` che dovessero essere richieste da tale definizione.

Esercizio 4

Definire una classe `Vettore` i cui oggetti rappresentano array di interi. `Vettore` deve includere un costruttore di default, l'overloading dell'uguaglianza, dell'operatore di output e dell'operatore di indicizzazione. Inoltre deve includere un costruttore di copia "profonda" e l'overloading dell'assegnazione come assegnazione "profonda".

Esercizio 5

```
class C {
public:
    C(): size(1), a(new int[1]) {a[0]=0;}
    C& operator=(const C& x) {
```

```
          if(this!=&x){
            size=x.size;
            a=new int[size];
            for(int i=0;i<size;i++) a[i]=x.a[i];
          }
          return *this;
    }
    void add(int k) {
      int *b=a;
          a=new int[size+1];
          ++size;
          a[0]=k;
          for(int i=1;i<size;i++) a[i]=b[i-1];
          delete[] b;
    }
    int& operator[](int i) const {return a[i];}
    void stampa() const {
          for(int i=0;i<size;i++) cout<<a[i]<< ' ';
    }
    ~C() {stampa(); cout<<"~C "; delete[] a;}
private:
    int size;
    int* a;
};

main(){
    C v; v.add(1);
    C w=v; w[1]=2;
    v.stampa(); cout<<"UNO\n";
    w.stampa(); cout<<"DUE\n";
    C* p=new C; p->add(3);
    *p=v;
    (*p)[0]=4; v[1]=5;
    v.stampa(); cout<<"TRE\n";
    w.stampa(); cout<<"QUATTRO\n";
    p->stampa(); cout<<"CINQUE\n";
    w=*p;
    w[1]=6; v[0]=7;
    v.stampa(); cout<<"SEI\n";
    w.stampa(); cout<<"SETTE\n";
    p->stampa(); cout<<"OTTO\n";
    delete p; cout<<"NOVE\n";
}
```

Il precedente programma compila correttamente. Quali stampe provoca la sua esecuzione?

Esercizi di Programmazione ad Oggetti

Lista n. 3

Esercizio 1

Definire un template di classe contenitore `Dizionario<T>` i cui oggetti rappresentano una collezione di coppie (chiave,valore) dove la chiave è una stringa ed il valore è di tipo `T`. Ad una certa stringa può essere associato un solo valore di `T`. Si tratta quindi di definire un template di classe analogo al contenitore STL `map<string,T>`. Dovranno essere disponibili le seguenti funzionalità:

- inserimento: `bool insert(string, const T&)`
- rimozione: `bool erase(string)`
- ricerca per chiave: `T* findValue(string)`
- ricerca per valore: `vector<string> findKey(const T&)`
- overloading degli operatori di indicizzazione e output

Esercizio 2

```
class Z {
public:
  ...
};

template <class T1, class T2=Z>
class C {
public:
  T1 x;
  T2* p;
};

template<class T1,class T2>
void fun(C<T1,T2>* q) {
  ++(q->p);
  if(true == false) cout << ++(q->x);
  else cout << q->p;
  (q->x)++;
  if(*(q->p) == q->x) *(q->p) = q->x;
  T1* ptr = &(q->x);
  T2 t2 = q->x;
}

main(){
  C<Z> c1; fun(&c1); C<int> c2; fun(&c2);
}
```

Si considerino le precedenti definizioni. Fornire una dichiarazione (non è richiesta la definizione) dei membri pubblici della classe `Z` nel **minor numero possibile** in modo tale che la compilazione del precedente `main()` non produca errori. **Attenzione:** ogni dichiarazione in `Z` non necessaria per la corretta compilazione del `main()` sarà penalizzata.

```
class Z {
public:
  int operator++() {} // 1
  Z& operator++(int) {} // 2
  bool operator==(const Z& x) const {} // 3
  Z(int x=0) {} // 4
```