

Analisi e progettazione del software

Compito di metà corso

19 novembre 2010

Esercizio 1 (punti 8) Si consideri la classe `Pila` (di interi) come definita a lezione, cioè con i campi `dim`, `top` e `vet` e le funzioni `Push()`, `Pop()`, `Top()` ed `EstVuota()`, i costruttori e gli operatori proposti in classe.

Si aggiunga alla classe `Pila` l'operatore unario `!`, realizzato con una *funzione esterna non friend*, che restituisce la pila ottenuta dall'oggetto di invocazione invertendo l'ordine degli elementi della pila e cancellando gli elementi alternativamente (un elemento cancellato e uno no, a partire dall'elemento affiorante).

Ad esempio, se `p` contiene i valori (3, 4, 12, 11, 13, 16, 22) (con 22 elemento affiorante), la pila `!p` dovrà contenere i valori (16, 11, 4). Se `p` è vuota, allora anche `!p` sarà vuota. La pila `p` non deve essere modificata.

Soluzione.

```
Pila operator!(Pila p)
{
    Pila p1;

    while (!p.EstVuota())
    {
        p.Pop();
        if (!p.EstVuota())
        {
            p1.Push(p.Top());
            p.Pop();
        }
    }
    return p1;
}
```

Esercizio 2 (punti 8) Si consideri la classe `Impegno` con i campi `nome` (stringa), `inizio` e `fine` (date), i costruttori e i selettori visti in classe. Si scriva la funzione esterna (debitamente modularizzata)

```
bool StessoImpegno(Impegno v[], int n, string nome_file);
```

che riceve come parametri il vettore `v` di oggetti di tipo `Impegno` (e la sua lunghezza `n`) e il nome di un file al cui interno sono scritte due date nel formato letto dall'operatore di input della classe `Data`, separate da spazi e dal carattere `|` come nel seguente esempio.

26/4/2010 | 15/5/2010

La funzione deve restituire `true` se esiste nel vettore un impegno tale che le due date cadono all'interno di esso (estremi inclusi), `false` in caso contrario. Si assuma che la classe `Data` disponga di tutti gli operatori che si ritengono utili.

Soluzione.

```
bool DataInclusa(Data d, Impegno i)
{
    return i.Inizio() <= d && d <= i.Fine();
}

bool StessoImpegno(Impegno v[], int n, string nome_file)
{
    Data d1, d2;
    char ch;
    int i;
    ifstream is(nome_file);
```

```

is >> d1 >> ch >> d2;
for (i = 0; i < n; i++)
    if (DataInclusa(d1,v[i]) && DataInclusa(d2,v[i]))
        return true;
return false;
}

```

Esercizio 3 (punti 17) Si considerino la classi A e B e la funzione main() definite qui sotto.

```

class B
{public:
    B(int n) { num = n; }
    void Set(int n) { num = n; }
    int Get() const { return num; }
private:
    int num;
};
class A
{public:
    A(int v = 0) : b(v) { val = v; pb = new B(v); }
    void Set1(int h) { val = h; }
    void Set2(int h) { b.Set(h); }
    void Set3(int h) { pb->Set(h); }
    int C1() const { return val; }
    int C2() const { return b.Get(); }
    int C3() const { return pb->Get(); }
    void Flip() { delete pb; pb = &b; }
    void Flop() { pb = new B(val); }
private:
    int val;
    B* pb;
    B b;
};

int main()
{
    A a1(2);
    A a2;
    a2.val = 3;
    a1.Set1(5);
    a1.C2() += a1.C1();
    a1.Set2(a1.C1() + a2.C2() + 1);
    a2(5);
    a1.Flip();
    cout << a1.C1() << ' '
         << a1.C2() << ' '
         << a1.C3() << endl;
    return 0;
}

```

- 3.1 (3 punti)** Segnalare le istruzioni della funzione main() che danno errore in compilazione e spiegare brevemente il motivo.
- 3.2 (3 punti)** Riportare cosa stampa il programma una volta eliminate le istruzioni errate.
- 3.3 (4 punti)** Costruire un esempio di funzione main() in cui si presenta il fenomeno dell'interferenza.
- 3.4 (4 punti)** Scrivere il costruttore di copia della classe A in modo che eviti la condivisione di memoria (si tralasci l'operatore di assegnazione).
- 3.5 (3 punti)** Scrivere il distruttore della classe A che rilasci la memoria non più utilizzata.

Soluzione.

3.1

```

int main()
{
    A a1(2);
    A a2;
    // a2.val = 3; ERRORE: usa la parte privata
    a1.Set1(5);
    // a1.C2() += a1.C1(); ERRORE: a sinistra dell'assegnazione c'e' un valore
    a1.Set2(a1.C1() + a2.C2() + 1);
    // a2(5); ERRORE: invoca il costruttore su un oggetto gia' costruito
    a1.Flip();
    cout << a1.C1() << ' '
         << a1.C2() << ' '
         << a1.C3() << endl;
    return 0;
}

```

3.2

5/6/6

3.3

```
int main()
{
    A a1(3), a2;
    a2 = a1;
    a2.Set3(-5);
    cout << a1.C3(); // stampa -5 invece di 3
    ...
}
```

3.4

```
A::A(const A& a)
: b(a.b)
{
    val = a.val;
    if (a.pb == &a.b)
        pb = &b;
    else
        pb = new B(*(a.pb));
}
```

3.5

```
A::~~A()
{
    if (pb != &b)
        delete pb;
}
```