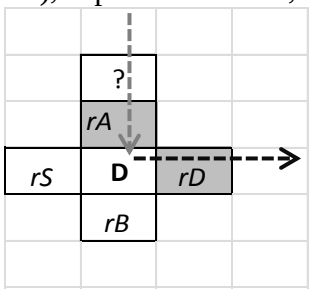
PROVA PRATICA 19 SETTEMBRE 2014

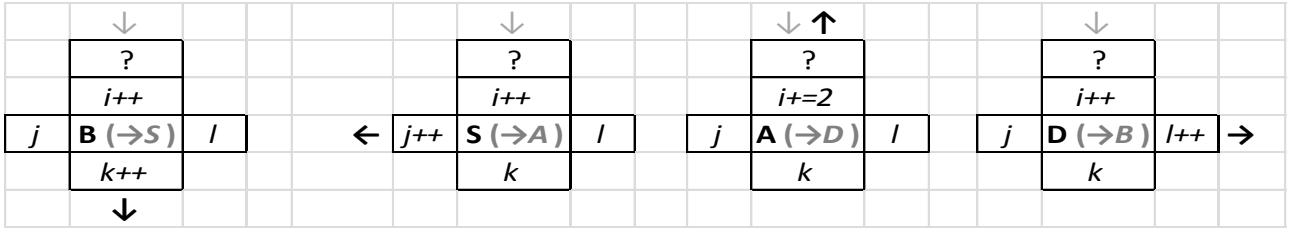
FONDAMENTI DI INFORMATICA I
FOND. DI INFORMATICA E PROGRAMMAZIONE A OGGETTI

Un `Deviatore` è un dispositivo capace di deviare, mediante uno specchio, la direzione di un raggio laser incidente su di esso. Per semplificare la descrizione del suo funzionamento, si immagini il deviatore come appeso al soffitto attraverso un gancio (il *punto interrogativo* di figura). Il laser colpisce il deviatore sempre dall'alto verso il basso, in corrispondenza del suo asse di simmetria verticale. Si supponga che lo specchio, posto al centro del deviatore, possa assumere solo 4 posizioni, a seconda di dove farà uscire il raggio laser: dal Basso (B), da Sinistra (S), dall'Alto (A) o da Destra (D). In corrispondenza di ciascuna direzione è posizionato un rilevatore (*rB*, *rS*, *rA* e *rD*), capace di contare, *indipendentemente dal verso*, quante volte esso viene attraversato dal laser.



In questo esempio lo specchio si trova in posizione D, ad indicare che farà uscire il laser verso destra.

I contatori *k*, *j*, *i* ed *l* contano quante volte un raggio laser ha attraversato il rilevatore *rB*, *rS*, *rA* ed *rD*, rispettivamente. All'inizio i contatori sono posti tutti a zero e lo specchio in posizione B (a significare che il primo raggio laser uscirà dal basso). Una volta che un raggio attraversa il deviatore, dopo aver aggiornato i contatori, la posizione dello specchio avanza di 90 gradi in senso orario (ossia la sua posizione cambia così: B→S, S→A, A→D, D→B). Nella figura sottostante sono mostrati la posizione dello specchio e i contatori all'arrivo dei primi 4 raggi laser.



La freccia grigia indica la direzione di ingresso del raggio laser, quella nera la direzione di uscita. In grassetto è mostrata la posizione delle specchio al momento del passaggio del laser (fra parentesi ed in corsivo la posizione successiva). k, j, i e l sono i contatori con i rispettivi incrementi.

PRIMA PARTE ***(qualora siano presenti errori di compilazione, collegamento o esecuzione in questa parte, l'intera prova verrà considerata insufficiente e pertanto non verrà corretta)***

- ✓ **`Deviatore d;`**
  Costruttore che crea un `Deviatore`. Inizialmente i contatori sono a zero e lo specchio è in B.
- ✓ **`d.laser();`**
  Operazione che simula l'arrivo di un nuovo raggio laser. L'operazione deve aggiornare il valore dei contatori e aggiornare la posizione dello specchio. Restituisce `true` nel caso in cui il raggio fuoriesca verso il basso, `false` negli altri casi.
- ✓ **`cout<<d;`**
  Operatore di uscita per il tipo `Deviatore`. Esempi di uscita sono:

`?`	`?`	`?`	`?`	`?`
`0`	`1`	`2`	`4`	`5`
`0B0`	`0S0`	`1A0`	`1D0`	`1B1`
`0`	`1`	`1`	`1`	`1`
all'inizio	*dopo primo laser*	*dopo secondo laser*	*dopo terzo laser*	*dopo quarto laser*

Come si può notare vengono mostrati sia la posizione corrente dello specchio che i contatori.

SECONDA PARTE ***(si invita a mettere sotto commento le operazioni di questa seconda parte che dovessero impedire la compilazione, il collegamento o la corretta esecuzione del codice)***

Più deviatori possono essere concatenati, a formare un `MultiDeviatore`. Sul `MultiDeviatore` debbono poter essere effettuate le seguenti operazioni:

- ✓ **`MultiDeviatore m(N);`**
  Costruttore che crea un `MultiDeviatore` composto da *N* `Deviatori`. Inizialmente tutti i deviatori hanno contatori a zero e specchi in B. *N* può essere arbitrario.
- ✓ **`m.multiLaser();`**
  Operazione che simula l'arrivo di un raggio laser nel primo deviatore. Qualora il raggio esca dal basso del primo, esso si propaga in ingresso al secondo (e così via per i successivi). La funzione restituisce `true` solo quando il raggio esce verso il basso dall'ultimo deviatore.
- ✓ **`cout<<m;`**
  Operatore di uscita per il tipo `MultiDeviatore`. Ecco alcuni esempi di uscita nel caso *N*=2:

? **0** **0B0** **0** **?** **0** **0B0** **0**	**?** **1** **0S0** **1** **?** **1** **0S0** **1**	**?** **2** **1A0** **1** **?** **1** **0S0** **1**	**?** **4** **1D0** **1** **?** **1** **0S0** **1**
situazione all'inizio	*dopo primo multiLaser*	*dopo secondo multiLaser*	*dopo terzo multiLaser*

Come si può notare vengono mostrati tutti i deviatori dal primo (quello più in alto) all'ultimo.

- ✓ **`m++;`**
  Operazione che appende un deviatore dopo l'ultimo, con contatori a zero e specchio in B.
- ✓ **`~MultiDeviatore();`**
  Distruttore.

Mediante il Linguaggio C++, realizzare i tipi di dato astratto **`Deviatore`** e **`MultiDeviatore`**, definiti dalle precedenti specifiche. **Gestire le eventuali situazioni di errore**.

NOTE SULLO SVOLGIMENTO DELLA PROVA PRATICA

AVVIO E IDENTIFICAZIONE

• Avviare la macchina in modalità diskless, scegliere "Fondamenti di Informatica I"
ed effettuare il login: **nome**: studenti **password**: studenti

• Aprire un terminale e al prompt spostarsi sulla cartella 'elaborato' (`$ cd ~/elaborato`). Si utilizzi il comando `pwd` per verificare che ci si trovi nella cartella corretta `/home/studenti/elaborato`.

• Sempre al prompt dare il comando `ident`, sempre da dentro la cartella. Lo script richiede i propri dati (cognome, nome, numero di matricola e password (la password **non va dimenticata** in quanto è indispensabile per scaricare da internet il proprio elaborato a consegna avvenuta). Il comando `ident` crea il file *matricola.txt* nella cartella corrente. Lo script può essere lanciato più volte, in tal caso il file *matricola.txt* viene sovrascritto. Per verificare che il file sia stato creato e che il contenuto sia quello giusto dare il comando (la password è codificata):
`$ cat /home/studenti/elaborato/matricola.txt`

• A questo punto il docente verifica che tutti gli studenti abbiamo effettuato l'identificazione, dopodiché provvede a inviare i seguenti file nella cartella `elaborato` del proprio PC: *compito.h, compito.cpp, main.cpp*.
Controllare pertanto che questi file, insieme al file *matricola.txt*, siano presenti sul proprio elaboratore.

SVOLGIMENTO DELLA PROVA

• Definire ed implementare il tipo di dato astratto richiesto e le relative funzioni nei file *compito.h* e *compito.cpp*. Il file *main.cpp* contiene la funzione principale `main()` ed è utilizzato dallo studente per testare la sua implementazione della classe. Il file *main.cpp* può essere modificato a piacere. In sede di valutazione dell'elaborato verrà considerato **esclusivamente il contenuto dei file compito.h e compito.cpp** ed è pertanto **vietato cambiare nome a tali file**.
Per compilare e linkare dare il comando:

`$ g++ main.cpp compito.cpp` (eseguibile invocabile tramite `$ ./a.out`)
(utilizzare `g++ -g` per includere le informazioni di debug qualora si intenda debuggare con `ddd`).

PER CONSEGNARE O RITIRARSI

Recarsi dal docente **dopo aver preso nota dell'identificativo della macchina** (esempi: g34, s23, c22, …).

USCITA CHE DEVE PRODURRE IL PROGRAMMA

```
Deviatore: test del costruttore e dell'op. di uscita
 ?
 0
0B0
 0

Deviatore: test della laser()
 ?
 1
0S0
 1

 ?
 2
1A0
 1

 ?
 4
1D0
 1

 ?
 5
1B1
 1

MultiDeviatore: test del costruttore e dell'op. di uscita
 ?
 0
0B0
 0
 ?
 0
0B0
 0

MultiDeviatore: test della multiLaser()
 ?
 1
0S0
 1
 ?
 1
0S0
 1

 ?
 2
1A0
 1
 ?
 1
0S0
 1

MultiDeviatore: test dell'operatore di incremento prefisso
 ?
 2
1A0
 1
 ?
 1
0S0
 1
 ?
 0
0B0
 0

MultiDeviatore: test del distruttore
```