

NOTE SULLO SVOLGIMENTO DELLA PROVA:

- PER SVOLGERE L'ELABORATO, APRIRE il Dev-C++ (dal Menù **Avvio** (o **Start**) nella barra degli strumenti in fondo allo schermo, selezionare Programmi e quindi Dev-C++);
- PRIMA DI INIZIARE LO SVOLGIMENTO DELL'ELABORATO, selezionare la voce **Identifica studente** nel menù **Strumenti** all'interno dell'ambiente Dev-C++ e inserire i dati richiesti;
- per svolgere l'elaborato, aprire il progetto *esaInf.dev* presente nel directory *c:\esame\esaInf* e scrivere le funzioni richieste nel file *compito.cpp*, già presente nel progetto;
- per una corretta stampa dell'elaborato bisogna mantenere il codice entro i margini imposti dall'ambiente Dev-C++ (linea verticale presente alla destra della pagina).

Immagine

Un'immagine in bianco e nero può essere rappresentata attraverso le seguenti costanti e strutture:

```
const int MAXN = 20;
const int DIMR = 8;
const int DIMC = 8;

enum Colore {Bianco, Nero};
enum Operatore {And, Or};

struct Immagine{
    char nome[MAXN];
    Colore pix[DIMR][DIMC];
};
```

La struttura `Immagine` è composta da due campi: un array di caratteri `nome` che contiene il nome dell'immagine; una matrice `pix` che memorizza il colore dei singoli pixel dell'immagine. Il tipo `Colore` definisce gli unici due colori possibili. Il tipo `Operatore` definisce due operatori che possono essere usati per modificare le immagini (come illustrato in seguito).

Scrivere il corpo delle seguenti funzioni C++.

1. **`bool carica(Immagine* pimm, const char nf[])`** che riempie l'oggetto puntato da `pimm` con i dati presenti nel file di nome `nf`. Il file contiene `DIMR` righe, ognuna composta da `DIMC` caratteri. Ogni carattere indica il colore del pixel posto in tale posizione secondo la seguente convenzione: `b` o `B` ad indicare bianco, `n` o `N`, ad indicare nero. La funzione provvede ad impostare il valore del campo `nome` dell'immagine puntata da `pimm` in modo tale che sia uguale al nome del file che contiene i dati (`nf`). La funzione restituisce *true* se il caricamento va a buon fine, *false* altrimenti

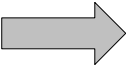
(file non trovato, dati insufficienti, presenza di caratteri diversi da quelli ammessi). Esempio (contenuto del file A.dat):

```
NNNBNNN
NNBNNBNN
NBNNNNBN
NBNNNNBN
NBBBBBBN
BNNNNNNB
BNNNNNNB
BNNNNNNB
```

2. **void stampa(const Immagine* pimm)** che stampa a video l'immagine puntata da pimm. I pixel bianchi vengono visualizzati usando il carattere X, mentre quelli neri vengono visualizzati con il carattere spazio. Il tutto è preceduto dal nome dell'immagine, come illustrato dal seguente esempio:


```
A.dat
  XX
  X X
 X  X
 X  X
XXXXXX
 X  X
 X  X
 X  X
```

3. **void ribalta(Immagine* pimm)** che ribalta l'immagine puntata da pimm lungo l'asse verticale.

<pre>3.dat XXXXXX X XX X XXX X X X X XXXXXX</pre>		<pre>3.dat XXXXXX XX X X XXX X X X X XXXXXX</pre>
---	---	---

4. **void modifica(Immagine* pim1, const Immagine* pim2, Operatore o)** che modifica l'immagine puntata da pim1 secondo il seguente criterio: ogni pixel di pim1 diventa pari all'and o all'or dei pixel di pim1 e pim2, a seconda del valore di o. And tra pixel: bianco solo se entrambi valgono bianco. Or tra pixel: bianco se almeno uno dei due vale bianco.

Esempio:

<pre>O.dat XXXXXX X X X X X X X X X X XXXXXX</pre>	<p>Or</p>	<pre>sm.dat X X XXXX</pre>		<pre>O.dat XXXXXX X X X X X X X X X X X XXXX X X X XXXXXX</pre>
---	-----------	-----------------------------	---	--

5. **int quanti(const Immagine* pimm, int x, int y, Colore c)** che conta quanti pixel di colore c ci sono intorno a quello di coordinate x, y.