

NOTE SULLO SVOLGIMENTO DELLA PROVA:

- Fare partire il Dev-C++ (dal Menù **Avvio** (o **Start**) nella barra degli strumenti in fondo allo schermo, selezionare Programmi e quindi Dev-C++);
- attraverso il menu **File->Open...** (o **apri** se il Dev-C++ è in italiano) aprire il progetto *APRIQUESTO.dev* presente nel directory *c:\esame\esaInf*;
- scrivere le funzioni richieste nel file *compito.cpp* già presente nel progetto;
- salvare spesso in modo da non perdere il lavoro nel caso in cui il PC abbia un malfunzionamento (per salvare premere il bottone a forma di dischetto del Dev-C++);
- 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).

Campo minato

Una versione semplificata del gioco “campo minato” (noto anche come “prato fiorito”) può essere realizzata a partire dalle seguenti costanti e dai seguenti tipi:

```
const int DIM = 6;

enum Casella {VUOTA, MINA, ESPLORATA_VUOTA, ESPLORATA_MINA};

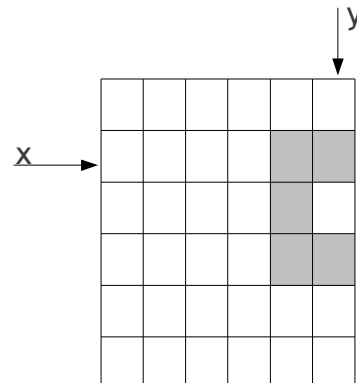
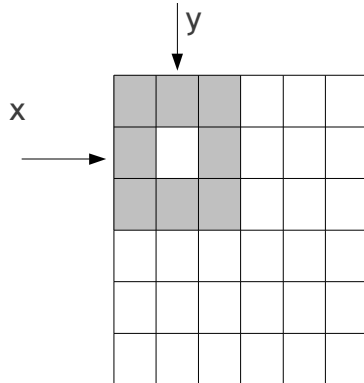
struct CampoMinato {
    int mosse;
    Casella campo[DIM][DIM];
};
```

Il tipo `CampoMinato` contiene le informazioni relative a una partita. In particolare, il membro `mosse` contiene il numero di mosse fatte dal giocatore dall'inizio della partita; `campo` invece contiene lo stato del campo di gioco. Il campo di gioco è una matrice `DIM x DIM` e ogni casella può trovarsi in uno degli stati specificati dal tipo `Casella`. Il significato dei valori del tipo `Casella` è il seguente: `VUOTA`, la casella non è ancora stata esplorata dal giocatore e non contiene una mina; `MINA`, la casella non è ancora stata esplorata dal giocatore e contiene una mina; `ESPLORATA_VUOTA`, la casella è stata esplorata dal giocatore e non contiene una mina; `ESPLORATA_MINA`, la casella è stata esplorata dal giocatore e contiene una mina.

Scrivere il corpo delle seguenti funzioni C++.

- **`void inizializza(CampoMinato* pm, int num)`** predispone il campo minato per l'inizio di una nuova partita. In particolare, il numero di mosse viene messo a zero e tutte le caselle sono inesplorate. `num` caselle contengono mine, mentre tutte le altre sono vuote. Le `num` caselle che contengono mine sono scelte a caso. Suggestimento: per generare un valore a caso usare la funzione `rand()`, che restituisce un intero compreso tra 0 e `RAND_MAX` (estremi inclusi).
- **`int suggerimento(const CampoMinato* pm, int vett[])`** riempie il vettore `vett` con gli indici di tutte le colonne contenenti almeno una mina. La funzione restituisce il numero di valori inseriti in `vett`.
- **`bool caricaPartita(CampoMinato* pm, const char nomef[])`** carica dal file di nome `nomef` lo stato della partita (supponiamo che lo stato sia stato salvato nel file in precedenza). Il file contiene nell'ordine: *i*) un intero che rappresenta il numero di mosse eseguite; *ii*) tanti caratteri quante sono le caselle del campo di gioco. In particolare, viene utilizzata la seguente corrispondenza: `v` per una casella vuota non esplorata; `m` per una casella contenente una mina e non ancora esplorata; `V` per una casella vuota esplorata; `M` per una casella contenente una mina e esplorata. I dati delle caselle sono memorizzati per righe. La funzione restituisce il valore `true` se l'operazione di caricamento va a buon fine, `false` altrimenti.

- **bool vicinoMina(const CampoMinato* pm, int x, int y)** verifica se intorno alla casella di riga *x* e colonna *y* è presente una mina. Il controllo deve essere eseguito sulle otto caselle che circondano quella di coordinate *x*, *y* (meno di otto se la casella con coordinate *x*, *y* si trova lungo un bordo o in un angolo). La funzione restituisce *true* se almeno una mina è presente, *false* altrimenti.



- **void stampa(const CampoMinato* pm)** stampa a video una rappresentazione dello stato del campo di gioco. Le caselle del campo vengono rappresentate dei seguenti caratteri: # a indicare le caselle non esplorate (indipendentemente dal fatto che contengano una mina o meno); * per indicare le caselle esplorate e contenenti una mina; le caselle prive di mine ed esplorate vengono rappresentate da un carattere spazio se non sono vicine a una mina, con un ! se sono vicine a una mina (si suggerisce di riutilizzare la funzione *vicinoMina* definita in precedenza). Esempio:

```
#####
##  ##!
## !#!#
#*#####
#!####
####!##
```