

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).

MARE

Supponiamo di voler realizzare una parte di un programma dedicato alla gestione di uno stabilimento balneare.

```
const int MAXS = 100;

const int NUMC = 5;
const int NUMR = 4;

const float prezzoOmbr = 10.0f;
const float prezzoSdraio = 6.5f;

struct Postazione {
    bool occupata;
    char nome[MAXS];
    bool ombrellone;
    int numSdraio;
};

struct Stabilimento {
    char nomeStab[MAXS];
    Postazione pos[NUMR][NUMC];
};
```

Uno stabilimento è rappresentato da una istanza della struttura `Stabilimento`. Tale struttura è composta da un campo `nomeStab` (che contiene il nome dello stabilimento) e da un campo `pos` (una matrice di oggetti di tipo `Postazione`). La riga 0 della matrice `pos` è quella più vicina al mare. La struttura `Postazione` contiene alcuni campi che indicano se una postazione è occupata o meno da un cliente, il nome del cliente, se il cliente usa un ombrellone o meno, e il numero di sdraio utilizzate dal cliente. I campi `nome`, `ombrellone`, e `numSdraio` sono significativi solo se la postazione è occupata.

Inizialmente tutte le posizioni dello stabilimento sono non occupate.

Scrivere il corpo delle seguenti funzioni C++.

1. **bool arrivo(Stabilimento* ps, const char n[], bool o, int s)** che rappresenta l'arrivo di un cliente. La funzione posiziona il cliente in una postazione libera. Tra le postazioni libere vengono assegnate prima quelle vicino al mare. I parametri *n*, *o* e *s* rappresentano rispettivamente il nome del cliente, se vuole un ombrellone, e il numero di sdraio. La funzione restituisce *false* se lo stabilimento è pieno, *true* in tutti gli altri casi.
2. **void stampaMappa(const Stabilimento* ps)** che stampa una mappa dello stabilimento nel modo indicato dal seguente esempio:

```

Bagno Maestrale
O3   X1   O2   O2   X4
O5           O1
           X3

```

Per ogni postazione occupata viene stampato un primo carattere che indica se viene usato un ombrellone (O) o meno (X). A seguire viene indicato il numero di sdraio utilizzate in tale postazione.

3. **float incasso(const Stabilimento* ps)** che calcola e restituisce l'incasso totale, supponendo che il costo di un ombrellone e quello di una sdraio siano quelli specificati dalle costanti *prezzoOmb* e *prezzoSdraio*.
4. **bool partenza(Stabilimento* ps, const char n[])** che rappresenta la partenza del cliente che ha il nome specificato dal parametro *n*. La posizione assegnata a tale cliente diventa libera. Tutti gli altri clienti situati nella medesima colonna e più lontani dal mare si spostano di una postazione verso il mare. La funzione restituisce *false* se un cliente con nome *n* non esiste, *true* altrimenti. Si supponga che non ci siano casi di omonimia.
5. **bool piuVicino(const Stabilimento* ps, int r, int c, int* pr, int* pc)** che trova la postazione libera più vicina a quella indicata dai parametri *r* e *c* (che indicano rispettivamente la riga e la colonna). Il numero di riga e di colonna della posizione libera trovata vengono restituiti attraverso *pr* e *pc* rispettivamente. La posizione più vicina deve essere cercata con andamento concentrico a partire da dalla posizione indicata da *r* e *c*, e rispettando i limiti della matrice, come indicato dal seguente esempio.

