

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).

Elezioni

Un sistema per la gestione delle liste elettorali e delle preferenze può essere realizzato attraverso le seguenti costanti e i seguenti tipi:

```
const int MAXN = 100;
const int MAXC = 50;

enum Lista {VIVA_ITALIA, POPOLO_DEMOCRATICO, ITALIA_DEI_COLORI,
            FRATELLANZA_NAZIONALE, RIFONDAZIONE_SINISTRA};

struct Candidato {
    char nome[MAXN];
    char cognome[MAXN];
    Lista lista;
    int voti;
};

struct Elez {
    Candidato cand[MAXC];
    int numCand;
    int votiNulli;
};
```

Il tipo `Lista` rappresenta le liste elettorali a cui appartengono i candidati. La struttura `Candidato` contiene nome, cognome, lista di appartenenza e numero di voti ricevuti del singolo candidato. Il tipo `Elez` viene utilizzato per contenere l'elenco dei candidati e il numero di voti nulli. In particolare, i dati dei candidati vengono memorizzati nell'array `cand` (che ne può contenere al più `MAXC`), mentre il campo `numCand` indica il numero di candidati effettivamente presenti.

Scrivere il corpo delle seguenti funzioni C++.

1. **bool insertCandidato(Elez* pe, const char n[], const char c[], Lista l)** che, se possibile, inserisce un nuovo candidato nella struttura puntata da `pe`. Il nome, il cognome e la lista di appartenenza del candidato sono espressi rispettivamente dai parametri `n`, `c` e `l`. L'inserimento non è possibile se la struttura puntata da `pe` contiene già un candidato che ha lo stesso nome, lo stesso cognome e appartiene alla stessa lista di quello da inserire, oppure se l'array `cand` è pieno. In tali casi viene restituito il valore *false* al chiamante. Nel caso in cui, invece, il candidato venga inserito la funzione restituisce il valore *true*.

bool voto(Elez* pe, const char n[], const char c[], Lista l) che incrementa di uno il numero dei voti del candidato avente nome `n`, cognome `c` e appartenente alla lista `l`. Se un candidato con tali caratteristiche non è presente, il voto è nullo e deve essere incrementato il contatore dei voti nulli. La funzione restituisce *true* se il candidato con le caratteristiche specificate è presente, *false* altrimenti.

2. **bool salvaOrd(const Elez* pe, const char nomef[])** che salva, nel file di nome `nomef`, l'elenco dei candidati ordinato per numero decrescente di voti. Il file deve avere il formato indicato dal seguente esempio (la lista di appartenenza non viene riportata):

```
75 Mario Rossi
34 Ignazio Ladorma
21 Dario Franceschetti
13 Nichi Comprala
11 Antonio Diroccia
2 Silvio Bernasconi
```

La funzione restituisce *true* se l'operazione va a buon fine, *false* altrimenti.

3. **float percentuale(const Elez* pe, Lista l)** che restituisce la percentuale di voti ottenuti dalla lista `l`. La percentuale deve essere calcolata sul totale dei voti validi (escludendo quindi i voti nulli).
4. **int migliori(const Elez* pe, Lista ll[])** che riempie l'array `ll` con le tre liste che hanno ottenuto più voti. La funzione restituisce il numero di elementi di `ll` effettivamente riempiti (potrebbero essere meno di tre nel caso in cui nel sistema ci siano meno di tre liste).