# 6 Primo riepilogo

Questa dispensa propone esercizi riepilogativi sui concetti visti finora.

## 6.1 Esercizi

### Esercizio 6.1

(TdE November 2007) Le seguenti dichiarazioni definiscono tipi di dati relativi alla categoria degli impiegati di un'azienda (gli impiegati possono essere di prima, seconda, ..., quinta categoria), agli uffici occupati da tali impiegati, all'edificio che ospita tali uffici (edificio diviso in 20 piani ognuno con 40 uffici).

```
/* definizioni dei tipi */
typedef struct {
  char nome[20], cognome[20];
  int cat; // contiene valori tra 1 e 5
  int stipendio;
} impiegato;

typedef enum {
  nord, nordEst, est, sudEst, sud, sudOvest, ovest, nordOvest
} esposizione;

typedef struct {
  int superficie; /* in m^2 */
  esposizione esp;
  impiegato occupante;
} ufficio;

/* definizioni delle variabili */
ufficio torre[20][40];

/* rappresenta un edificio di 20 piani con 40 uffici per piano */
```

1. Si scriva un frammento di codice (che includa eventualmente anche le dichiarazioni di ulteriori variabili) che stampi il cognome, lo stipendio e la categoria di tutte e sole le persone che occupano un ufficio orientato a sud oppure a sudEst e avente una superficie compresa tra 20 e 30 metri quadri.

2. Visualizzi a schermo i piani che non hanno neanche un ufficio esposto a nord.

3. Stampi lo stipendio medio degli impiegati in questi piani che si chiamano "Giacomo" di nome.

### Esercizio 6.2

(TdE Novembre 2006) Le seguenti dichiarazioni definiscono un tipo di dato che rappresenta una matrice quadrata di dimensione `DIM` (non indicata nel codice proposto, ma

che deve essere precisata per ottenere del codice compilabile) e una variabile `m` di quel tipo.

```
#define DIM ... /* dimensione della matrice, da precisare */

typedef int matriceQuadrata [DIM][DIM];
matriceQuadrata m;
int s,p;
```

Scrivere un frammento di codice che permetta di calcolare nelle variabili:

- `s` la somma dei prodotti degli elementi delle due diagonali della matrice, presi ordinatamente.
- `p` il prodotto delle somme degli elementi delle due diagonali della matrice, presi ordinatamente

Per esempio, se `DIM` avesse valore 5 e la matrice `m` fosse la seguente:

| | | | | |
|---|---|---|---|---|
| 3 | 2 | 1 | 5 | 8 |
| 2 | 5 | 1 | 6 | 4 |
| 12 | 4 | 2 | 6 | 7 |
| 5 | 2 | 13 | 6 | 8 |
| 7 | 3 | 1 | 4 | 1 |

allora:

`s` = $3 \cdot 8 + 5 \cdot 6 + 2 \cdot 2 + 6 \cdot 2 + 1 \cdot 7$

`p` = $(3 + 8) \cdot (5 + 6) \cdot (2 + 2) \cdot (6 + 2) \cdot (1 + 7)$

### Esercizio 6.3

Assumendo che `c1` e `c2` siano due variabili di tipo char, che memorizzano rispettivamente i valori `'e'` ed `'m'` indicare, per ognuna delle espressioni logiche:

1. se l'espressione è vera o falsa (per i valori delle variabili sopra indicati),
2. se è sempre vera per qualsiasi valore che le due variabili possono assumere,
3. se è sempre falsa per qualsiasi valore che le due variabili possono assumere.

Si fornisca una giustificazione per ogni risposta inserita nella tabella (risposte senza giustificazione potranno essere considerate nulle).

1. `((c1 != 'e') && (c2 == 'm')) || ((c1 != 'h') && (c2 == 'm'))`

2. `(c1 < 'g') || !((c1 <= 'g') && (c1 != 'g'))`
3. `(c1 <= 'm') || ((c2 > 'm') || !(c2 > c1))`

### Esercizio 6.4

(TdE Novembre 2007) Trasformare il frammento di codice in un equivalente frammento che contenga istruzioni `while` invece che `for`. Specificare, infine, qual è il valore stampato quando il codice viene eseguito, giustificando adeguatamente la risposta.

```
int i, j, s;
s = 1;

for (i=3; i>=0; i--)
  for (j=3; j>=0; j--)
    if ((i+j) % 2 != 0 )
      s = s * 2;

printf("%d", s);
```

### Esercizio 6.5

Si scriva un programma che controlla se una matrice quadrata di dimensione nota è simmetrica (ovver se l'elemento alla riga `i`, colonna `j` coincide con l'elemento alla riga `j` colonna `i`).

Esempio di matrice simmetrica:

| | | |
|---|---|---|
| 1 | 12 | 1 |
| 12 | 0 | 3 |
| 1 | 3 | 23 |