Esercizi fatti a lezione

Vittorio Zaccaria

20 dicembre 2018

Indice

Indice ***1***

1 Esercizi Linguaggio C ***5***
1.1 Conversione tempo . . . 5
1.2 Tracing di programma . . . 6
1.3 Terna pitagorica . . . 7
1.4 Tracing di programma . . . 8
1.5 Tracing di programma . . . 9

2 Esercizi Linguaggio C - Array semplici e stringhe ***11***
2.1 Calcolo del valore massimo all'interno di un vettore . . . 11
2.2 Tabella caratteri ASCII . . . 13
2.3 Conversione stringa da caratteri minuscoli a maiuscoli . . . 14
2.4 Stringhe palindrome . . . 15
2.5 Conta i caratteri . . . 17

3 Esercizi Linguaggio C - Numerica e ordinamento ***19***
3.1 Stampa divisori di un numero . . . 19
3.2 Numeri di Fibonacci . . . 21
3.3 Somma elementi matrice quadrata . . . 23
3.4 Bubble sort . . . 24

4 Esercizi Linguaggio C - Strutture dati e array ***25***
4.1 Memorizzazione di date . . . 25
4.2 Calcolo distanza tra due punti su piano cartesiano . . . 27
4.3 Interpretazione programma . . . 29
4.4 The Matrix . . . 30
4.5 Base dati arresti . . . 32
4.6 Twitters . . . 34
4.7 Spoutify . . . 38

4.8 Audio . . . 40
4.9 Compagnia telefonica . . . 42
4.10 Carte di credito frodate . . . 44
4.11 Operatori telefonici . . . 46
4.12 Feisbuk . . . 48
4.13 Discounted computers Inc. . . . 52
4.14 Taxi . . . 54
4.15 Farmaci . . . 56
4.16 Azienda . . . 58
4.17 Autofficina . . . 60
4.18 War– . . . 62
4.19 Parcheggio . . . 64

5 Esercizi su linguaggio C consigliati ***67***

6 Esercizi di introduzione a Matlab ***71***
6.1 Introduzione a find . . . 71
6.2 Radice quadrata iterativa . . . 73
6.3 Stringhe palindrome . . . 74
6.4 Troviamo l'errore . . . 75
6.5 Comprensione programma . . . 76
6.6 Script . . . 77

7 Esercizi Matlab - Funzioni non ricorsive ***79***
7.1 Matrici . . . 79
7.2 Codice ISBN . . . 81
7.3 Caldaia . . . 83
7.4 Andamento capitale . . . 86
7.5 Cinematica . . . 88

8 Esercizi Matlab - Funzioni ricorsive ***91***
8.1 Mele al mercato . . . 91
8.2 Quadrati concentrici . . . 93
8.3 Grigliopoli . . . 94
8.4 Rolling text . . . 95
8.5 Matrice con frazioni . . . 96
8.6 Funzione sconosciuta . . . 98
8.7 Estrazione cifra . . . 99
8.8 Numeri di Catalan . . . 100
8.9 Comprensione funzione . . . 101
8.10 Comprensione funzione . . . 102
8.11 Traiettoria aereo . . . 104
8.12 Coefficiente binomiale . . . 105
8.13 Massimo comun divisore . . . 106

9 Esercizi su tabelle della verità ***107***
9.1 Esercizio 1 . . . 107
9.2 Esercizio 2 . . . 108
9.3 Esercizio 3 . . . 109

10 Codifica binaria dei numeri interi ***111***
10.1 Codifica numeri . . . 111
10.2 Codifica numeri . . . 112

10.3 Numero minimo di bit 113

11 Informazioni utili ***115***

Capitolo 1

Esercizi Linguaggio C

1.1 Conversione tempo

Si scriva un programma in linguaggio C con la seguente firma:

$$(\texttt{tempoSec}) \rightarrow (\texttt{h,m,s})$$

Si assuma che i dati in ingresso, se ve ne sono, siano introdotti da tastiera e che i valori elaborati siano stampati a video.

DATI IN INGRESSO

- `tempoSec`: intervallo di tempo espresso in secondi

DATI DA ELABORARE

- `h`: il numero intero di ore corrispondente all'intervallo introdotto
- `m`: il rimanente numero intero di minuti
- `s`: il rimanente numero di secondi

ESEMPIO

```
Inserisci il numero di secondi: 3661
Equivalgono a 1 ore, 1 minuti e 1 secondi
```

Soluzione

```c
#include <stdio.h>
int main()
{
    int sec, min, h;
    printf("Inserisci il numero di secondi:\n");
    scanf("%d", &sec);

    h = sec/3600;
    min = (sec - h*3600)/60;
    sec = sec - h*3600 - min*60;

    printf("Equivalgono a %d ore, %d minuti e %d secondi\n", h, min, sec);
    return 0;
}
```

1.2 Tracing di programma

In questo esercizio, cosidetto *di tracing*, viene richiesto di simulare ”mentalmente” il programma seguente e predire che cosa stamperà a terminale durante la sua esecuzione. In pratica, fate finta di essere voi il calcolatore ed eseguite le istruzioni partendo dalla prima fino a che non raggiungete l'ultima. Utilizzate un foglio di carta per annotare il valore corrente delle variabili e abbiate cura di tenerlo aggiornato ogni volta che eseguite ”mentalmente” una istruzione.

```c
main()
{
    int i1 = 3, i2 = 4;
    float f1 = 15.45, f2 = 3.1415;
    char c1 = 'a', c2 = 'b';

    /*quanto valgono le seguenti operazioni eseguite in sequenza?*/

    i2 = i1 + 5;
    printf("i2 = %d\n", i2);
    f1 = i1 + 1.1;      /* i1 convertito in float */
    printf("f1 = %f\n", f1);
    f2 = f2 * f2;
    printf("f2 = %f\n", f2);
    i1=f2+8;                /* f2 convertito in intero */
    printf("i1 = %d\n", i1);
    i2 = i2 + c1;       /* c1 = 97, i2 = 105 */
    printf("i2 = %d\n", i2);
    c2 = c2 + 3;        /* 98 + 3 */
    printf("c2 = %c (corrisponde al codice ASCII %d)\n", c2, c2);
    system("pause");
}
```

Soluzione

```
i2 = 8
f1 = 4.100000
f2 = 9.869022
i1 = 17
i2 = 105
c2 = e (corrisponde al codice ASCII 101)
```

1.3 Terna pitagorica

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio C. A meno che non sia richiesto, non è necessario includere file headers di altre librerie o dichiarare un main. Inoltre, il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Scrivere un programma che verifica se una terna di numeri introdotti dall'utente rispetta il teorema di Pitagora:

$$x^2 + y^2 = z^2$$

Soluzione

```c
#include <stdio.h>
int main() {
        int cat1, cat2, ip;
        printf("Scrivi il valore del primo cateto:\n");
        scanf("%d", & cat1);
        printf("Scrivi il valore del secondo cateto:\n");
        scanf("%d", & cat2);
        printf("Scrivi il valore dell'ipotenusa:\n");
        scanf("%d", & ip);
        if(cat1*cat1 + cat2*cat2 == ip*ip) {
                printf("La terna e' pitagorica\n");
        }
        else {
                printf("La terna non e' pitagorica\n");
        }
}
```

1.4 Tracing di programma

In questo esercizio, cosidetto *di tracing*, viene richiesto di simulare ”mentalmente” il programma seguente e predire che cosa stamperà a terminale durante la sua esecuzione. In pratica, fate finta di essere voi il calcolatore ed eseguite le istruzioni partendo dalla prima fino a che non raggiungete l'ultima. Utilizzate un foglio di carta per annotare il valore corrente delle variabili e abbiate cura di tenerlo aggiornato ogni volta che eseguite ”mentalmente” una istruzione.

```c
int main() {
  int a = 0;
  if (a = 1) {
    printf("A è uguale a 1");
  } else {
    printf("A è uguale a 0");
  }
}
```

Soluzione

L'uso dell'operatore di assegnamento `=` porta ad eseguire il ramo *then* dell'istruzione di controllo `if` (il valore di un assegnamento è il valore assegnato) quindi viene stampato `A è uguale a 1`.

1.5 Tracing di programma

In questo esercizio, cosidetto *di tracing*, viene richiesto di simulare ”mentalmente” il programma seguente e predire che cosa stamperà a terminale durante la sua esecuzione. In pratica, fate finta di essere voi il calcolatore ed eseguite le istruzioni partendo dalla prima fino a che non raggiungete l'ultima. Utilizzate un foglio di carta per annotare il valore corrente delle variabili e abbiate cura di tenerlo aggiornato ogni volta che eseguite ”mentalmente” una istruzione.

```
1  int main() {
2    int s, i, j;
3    s = 0;
4    for (i = 1; i <= 10; i++) {
5      j = i * 2;
6      /* Misura I e J */
7      while (j > 0) {
8        s = s + 1;
9        j = j - 1;
10     }
11     /* Misura S */
12     if (s % 2 == 0)
13       printf("%d", s);
14   }
15 }
```

Soluzione

Per stabilire cosa (e quando) viene stampato alla linea 13, dobbiamo seguire l'andamento di `s` (la condizione alla riga 12 infatti dipende da `s`). A sua volta, `s` dipende da `i` e `j`; bisogna quindi ricavare l'andamento di `i` e `j` come prima cosa.
Per comodità, fissiamo alla riga 6 ed alla riga 11 i punti in cui ”virtualmente” misuriamo il valore di `i, j` ed `s.` Ogni qualvolta che, eseguendo una istruzione alla volta, passeremo attraverso quelle righe, aggiungeremo i valori correnti delle variabili alla seguente tabella:

```
1  i alla riga 6  = 1 2 3  4  5  6  7  8  9  10
2  j alla riga 6  = 2 4 6  8  10 12 14 16 18 20
3  s alla riga 11 = 2 6 12 20 30 42 56 72 90 110
```

Dato l'andamento di `s`, l'istruzione `printf` verrà sempre eseguita (poiché `s` è sempre pari). Ciò significa che al terminale verrà stampato semplicemente il suo valore:

```
1  2 6 12 20 30 42 56 72 90 110
```

Capitolo 2

Esercizi Linguaggio C - Array semplici e stringhe

2.1 Calcolo del valore massimo all'interno di un vettore

Si scriva un programma in linguaggio C con la seguente firma:

$$(N, \texttt{n}_1, \ldots, \texttt{n}_N) \rightarrow (R)$$

Si assuma che i dati in ingresso, se ve ne sono, siano introdotti da tastiera e che i valori elaborati siano stampati a video.

DATI IN INGRESSO

- N: Rappresenta il numero di valori della sequenza inserita successivamente dall'utente
- n_i: Un valore intero inserito dall'utente come i -simo elemento

DATI DA ELABORARE

- `R`: Il massimo dei valori n_i

ESEMPIO

```
Di quanti valori vuoi calcolare il massimo? 3
Inserisci il valore 1: 7
Inserisci il valore 2: 3
Inserisci il valore 3: -1

Il valore massimo e': 7
```

ULTERIORI VINCOLI E SPIEGAZIONI: Si crei un array che riesca a contenere 50 elementi e si memorizzino i valori inseriti in tale array. Si controlli che il valore di N sia maggiore di zero e inferiore a 50 prima di richiedere i numeri. Nel caso il valore di N sia maggiore di 50, richiederne il valore un'altra volta.

Soluzione

```c
#include <stdio.h>

#define MAX 50

int main() {
  int N;
  int numeri[N];
  int i;
```

```c
  int R;
  do {
    printf("Di quanti valori vuoi calcolare il massimo?");
    scanf("%d", &N);
  } while (N > 50 || N <= 0);
  for (i = 0; i < N; i++) {
    printf("Inserisci il valore %d:", i + 1);
    scanf("%d", &numeri[i]);
  }
  R = numeri[0];
  for (i = 1; i < N; i++) {
    if (numeri[i] > R) {
      R = numeri[i];
    }
  }
  printf("Il valore massimo e': %d", R);
  return 0;
}
```

2.2 Tabella caratteri ASCII

Si scriva un programma in linguaggio C con la seguente firma:

$$() \rightarrow (\texttt{l}_1, \texttt{l}_2, \ldots)$$

Si assuma che i dati in ingresso, se ve ne sono, siano introdotti da tastiera e che i valori elaborati siano stampati a video.

DATI DA ELABORARE

- $\texttt{l}_i$: rappresente il carattere i-esimo dell'alfabeto

ESEMPIO

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

ULTERIORI VINCOLI E SPIEGAZIONI: Non è possibile usare più di due `printf` nel codice. Si suggerisce di usare un ciclo `for` e di non introdurre arrays.

Soluzione

```c
#include <stdio.h>

int main() {
  char c;
  for (c = 'A'; c <= 'Z'; c++) {
    printf("%c", c);
  }
  return 0;
}
```

2.3 Conversione stringa da caratteri minuscoli a maiuscoli

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio C. A meno che non sia richiesto, non è necessario includere file headers di altre librerie o dichiarare un main. Inoltre, il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Scrivere un programma che converte una stringa di caratteri inseriti dall'utente in maiuscolo.

Soluzione

```c
#include <stdio.h>
#define MAX_LEN 100

int main() {
  int offset = 'A' - 'a';
  char s[MAX_LEN];
  int i = 0;
  printf(
      "Inserisci una stringa di caratteri minuscoli (no spazi, no numeri):\n");
  scanf("%s", s);
  while (s[i] != '\0') {
    if (s[i] >= 'a' && s[i] <= 'z') {
      s[i] += offset;
    }
    i++;
  }
  printf("La stringa ora e' %s\n", s);
  return 0;
}
```

2.4 Stringhe palindrome

Si scriva un programma in linguaggio C con la seguente firma:

$$(\texttt{parola}) \rightarrow (\texttt{messaggio})$$

Si assuma che i dati in ingresso, se ve ne sono, siano introdotti da tastiera e che i valori elaborati siano stampati a video.

DATI IN INGRESSO

- `parola`: una stringa di massimo 50 caratteri

DATI DA ELABORARE

- `messaggio`: Si veda gli esempi sotto

ESEMPIO

```
Inserisci una parola: pippo
'pippo' NON è una parola palindroma
```

ESEMPIO

```
Inserisci una parola: anilina
'anilina' è una parola palindroma
```

ULTERIORI VINCOLI E SPIEGAZIONI: Una stringa è ***palindroma*** se, letta da destra a sinistra, equivale alla stessa letta da sinistra a destra.

Soluzione

```c
#include <stdio.h>
#include <string.h>

#define MAX 50

int main() {
  char parola[MAX];
  int i, palindroma, len;

  printf("Inserisci una parola: ");
  scanf("%s", parola);

  len = strlen(parola);
  palindroma = 1;

  for (i = 0; i < len / 2 && palindroma != 0; i++) {
    if (parola[i] != parola[len - 1 - i])
      palindroma = 0;
  }

  printf("'%s' ", parola);

  if (palindroma == 0)
    printf("NON ");

```

```
26    printf("è una parola palindroma\n");
27
28    return 0;
29  }
```

2.5 Conta i caratteri

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio C. A meno che non sia richiesto, non è necessario includere file headers di altre librerie o dichiarare un main. Inoltre, il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Si supponga di avere una stringa `str` contenente al massimo 100 caratteri alfabetici, senza spazi, ad esempio:

```c
char str[100] = "aadddfffzzzzdd";
```

Scrivere una porzione di codice che, per ogni carattere `c` ***a partire dall'ultimo fino ad arrivare al primo***, stampi senza lasciare spazi il carattere `c`, seguito dal numero di volte che questo compare consecutivamente in `str`.
Ad esempio, per la stringa di cui sopra, il programma deve stampare:

```
d2z4f3d3a2
```

RISPOSTA/SOLUZIONE:

```c
char c;
int freq = 1, i, n = strlen(str) - 1;
c = str[n];
for (i = n - 1; i >= 0; i--) {
  if (str[i] == c) {
    freq++;
  } else {
    printf("%c%d", c, freq);
    freq = 1;
    c = str[i];
  }
}
printf("%c%d\n", c, freq);
```

Capitolo 3

Esercizi Linguaggio C - Numerica e ordinamento

3.1 Stampa divisori di un numero

Si scriva un programma in linguaggio C con la seguente firma:

$$(\texttt{numero}) \rightarrow (\texttt{d}_1\texttt{, d}_2\texttt{, ..., d}_k)$$

Si assuma che i dati in ingresso, se ve ne sono, siano introdotti da tastiera e che i valori elaborati siano stampati a video.

DATI IN INGRESSO

- `numero`: numero intero, maggiore di 0, di cui bisogna trovare i k divisori

DATI DA ELABORARE

- $\texttt{d}_i$: i-esimo divisore di numero

ESEMPIO

```
Inserisci un numero: 10
I divisori sono:
2
5
```

ESEMPIO

```
Inserisci un numero: 20
I divisori sono:
2
4
5
10
```

ULTERIORI VINCOLI E SPIEGAZIONI: Si ricordi che uno dei modi in cui e' possibile verificare se un numero (positivo) e' divisibile per un altro e' verificare se il resto della divisione tra il primo e il secondo numero sia nullo; ovvero, `n` e' divisibile per `i` se `(n % i)` e' uguale a 0.

Soluzione

```c
#include <stdio.h>
int main() {
  int n, i;
  printf("Inserisci un numero: ");
  scanf("%d", &n);
  printf("I divisori sono: ");
  i = 1;
  while (i <= n) {
    if (n % i == 0)
      printf("%d ", i);
    i++;
  }
  printf("\n");
  return 0;
}
```

3.2 Numeri di Fibonacci

Si scriva un programma in linguaggio C con la seguente firma:

$$(\texttt{n}) \rightarrow (\texttt{f}_0\texttt{, ... f}_n)$$

Si assuma che i dati in ingresso, se ve ne sono, siano introdotti da tastiera e che i valori elaborati siano stampati a video.

DATI DA ELABORARE

- $\texttt{f}_j$: il numero della serie di fibonacci in posizione j

DATI IN INGRESSO

- `n`: l'indice dell'ultimo numero della serie da stampare. Si assuma che l'utente inserisca un numero maggiore o uguale a 1

ESEMPIO

```
Indice dell'ultimo numero della serie da stampare: 5
0
1
1
2
3
5
```

ULTERIORI VINCOLI E SPIEGAZIONI:
Ricordiamo che la successione dei numeri di Fibonacci

$$[f_0, f_1, \ldots f_j \ldots]$$

è definita come segue:

$$f_j = \begin{cases} 0 & \text{se la posizione } j = 0 \\ 1 & \text{se la posizione } j = 1 \\ f_{j-1} + f_{j-2} & \text{se la posizione } j > 1 \end{cases} \tag{3.1}$$

Ulteriori vincoli sono i seguenti:

- La sequenza di valori f_j deve essere prodotta con un ciclo
- Non è possibile utilizzare array.
- Il massimo numero di variabili `int` utilizzabili è 5.

Suggerimento: si consiglia, per ciascuna iterazione j di utilizzare due variabili `f1` ed `f2`, che contengano, rispettivamente, i valori di f_{j-1} ed f_{j-2}, da aggiornare ad ogni iterazione.

Soluzione

```c
#include <stdio.h>
main() {
  int f0, f1, f2, j, n;
  printf("Indice dell'ultimo numero della serie da stampare (>1): ");
  scanf("%d", &n);

  printf("0\n");
  printf("1\n");

  f2 = 0;
  f1 = 1;
  j = 2;
  while (j <= n) {
    f0 = f1 + f2;
    printf("%d\n", f0);
    f2 = f1;
    f1 = f0;
    j = j + 1;
  }
}
```

3.3 Somma elementi matrice quadrata

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio C. A meno che non sia richiesto, non è necessario includere file headers di altre librerie o dichiarare un main. Inoltre, il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Richiesta

- Scrivere un programma che, data una matrice quadrata di dimensione N (costante), calcoli la somma degli elementi sulle due diagonali.
- La matrice deve essere richiesta all'utente

Soluzione

```c
#include <stdio.h>
#define N 4
int main() {
  int i = 0, j = 0;
  int matrice[N][N];
  int diag = 0;     /* somma dei valori della diagonale principale */
  int antidiag = 0; /* somma dei valori dell'antidiagonale */
  for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
      printf("numero in riga %d - colonna %d: ", i + 1, j + 1);
      scanf("%d", &matrice[i][j]);
    }
  }
  printf("\n");
  for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++)
      printf("%2d ", matrice[i][j]);
    printf("\n");
  }
  printf("\n");
  for (i = 0; i < N; i++) {
    diag = diag + matrice[i][i];
    antidiag = antidiag + matrice[N - 1 - i][i];
  }
  printf("Somma diag principale: %d\n", diag);
  printf("Somma diag secondaria: %d\n", antidiag);
  return 0;
}
```

3.4 Bubble sort

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio C. A meno che non sia richiesto, non è necessario includere file headers di altre librerie o dichiarare un main. Inoltre, il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Si scriva un programma che richieda una sequenza di numeri interi all'utente e la stampi ordinata in modo crescente. Si usi l'algoritmo del Bubble Sort.

Soluzione

```c
#include <stdio.h>
#define DIMENSIONE_ARRAY 10
int main() {
  int elenco[DIMENSIONE_ARRAY];
  int i, j, temporaneo, n;
  do {
    printf("Inserisci il numero di elementi: ");
    scanf("%d", &n);
  } while (n >= DIMENSIONE_ARRAY);
  for (i = 0; i < n; i++) {
    printf("Inserisci elemento numero %d: ", i);
    scanf("%d", &elenco[i]);
  }
  for (i = 0; i < n; i++) {
    for (j = 0; j < n - 1; j++) {
      if (elenco[j] > elenco[j + 1]) {
        temporaneo = elenco[j + 1];
        elenco[j + 1] = elenco[j];
        elenco[j] = temporaneo;
      }
    }
  }
  printf("Array ordinato: ");
  for (i = 0; i < n; i++) {
    printf("%d ", elenco[i]);
  }
}
```

Capitolo 4

Esercizi Linguaggio C - Strutture dati e array

4.1 Memorizzazione di date

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio C. A meno che non sia richiesto, non è necessario includere file headers di altre librerie o dichiarare un main. Inoltre, il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Si chiede di estendere i tipi del C per rappresentare, in forma organica, i seguenti tipi:

- un tipo di dato `tipo_orario` atto a rappresentare un classico orario di ore, minuti e secondi.
- un tipo di dato `tipo_data` atto a rappresentare una classica data dell'anno.
- (usando i tipi definiti precedentemente) un tipo di dato `tipo_evento` atto a rappresentare un'evento caratterizzato da data e orario.
- un tipo di dato `tipo_programma` atto a rappresentare l'evento della trasmissione di un programma di cui si conosce il nome.
- due variabili, `palinsesto` e `primaserata`, di tipo array di 30 elementi di `tipo_programma`.

Si chiede inoltre di scrivere una parte di programma C che copi in `primaserata` tutti gli elementi di `palinsesto` che sono trasmessi tra le 20 e le 22.

Soluzione

```
typedef struct {
  unsigned short ore;
  unsigned short minuti;
  unsigned short secondi;
} tipo_orario;

typedef unsigned short tipo_giorno; /* giorno del mese */

typedef enum {gennaio, febbario, marzo, aprile, maggio, giugno, luglio, agosto,
  settembre, ottobre, novembre, dicembre
} tipo_mese;

typedef int tipo_anno;
```

```
14
15  typedef struct {
16    tipo_giorno giorno;
17    tipo_mese mese;
18    tipo_anno anno;
19  } tipo_data;
20
21  typedef struct {
22    tipo_data data;
23    tipo_orario orario;
24  } tipo_evento;
25
26  typedef char stringa[100];
27
28  typedef struct {
29    tipo_evento evento_trasmissione_programma;
30    stringa nome_programma;
31  } tipo_programma;
32
33  tipo_programma palinsesto[30], primaserata[30];
34
35  int main() {
36    int i,j=0;
37    for(i=0; i<30; i++) {
38      int dopoleotto = palinsesto[i].evento_trasmissione_programma.orario.ore >= 20;
39      int primadelledieci = palinsesto[i].evento_trasmissione_programma.orario.ore <= 22;
40      if(dopoleotto && primadelledieci) {
41        primaserata[j] = palinsesto[i];
42        j++;
43      }
44    }
45  }
```

4.2 Calcolo distanza tra due punti su piano cartesiano

Si scriva un programma in linguaggio C che implementi un algoritmo con la seguente firma:

$$(\texttt{a}_x\texttt{, a}_y\texttt{, b}_x\texttt{, b}_y) \rightarrow (\texttt{dist(a,b)})$$

Si assuma che i dati in ingresso, se ve ne sono, siano introdotti da tastiera e che i valori elaborati siano stampati a video. Il programma deve ***ripetutamente*** eseguire l'algoritmo finchè una ***particolare condizione*** di uscita non è verificata.

DATI IN INGRESSO

- $\texttt{a}_x$: coordinata x del punto cartesiano a
- $\texttt{a}_y$: coordinata y del punto cartesiano a
- $\texttt{b}_x$: coordinata x del punto cartesiano b
- $\texttt{b}_y$: coordinata y del punto cartesiano b

DATI DA ELABORARE

- `dist(a,b)`: distanza euclidea fra a e b

CONDIZIONE DI USCITA: Il programma deve terminare se tutte e 4 le coordinate in ingresso sono pari a 0.
ULTERIORI VINCOLI E SPIEGAZIONI: Deve essere definito ed utilizzato un nuovo tipo `Punto` atto a rappresentare in maniera organica un punto bidimensionale.

Soluzione

```c
#include <math.h>
#include <stdio.h>

typedef struct {
  float x;
  float y;
} Punto;

int main() {
  Punto a, b;
  int uscita = 0;
  float distanza;

  do {
    printf("punto 1, coord x: ");
    scanf("%f", &a.x);

    printf("punto 1, coord y: ");
    scanf("%f", &a.y);

    printf("punto 2, coord x: ");
    scanf("%f", &b.x);

    printf("punto 2, coord y: ");
    scanf("%f", &b.y);

    if (a.x == 0 && a.y == 0 && b.x == 0 && b.y == 0) {
      uscita = 1;
```

```
29        printf("Esco dal programma...\n");
30      } else {
31        distanza = sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
32        printf("distanza: %f\n", distanza);
33      }
34    } while (uscita == 0);
35
36    return 0;
37  }
```

4.3 Interpretazione programma

Si consideri il seguente programma, che richiede in ingresso l'inserimento di ***una matrice 3x3***:

```c
#include <stdio.h>
#define DIM 3
int main() {
  int i, j, M[DIM][DIM];

  int s[3] = {0, 0, 0};

  for (i = 0; i < DIM; i++)
    for (j = 0; j < DIM; j++)
      scanf("%d", &M[i][j]);

  for (i = 0; i < DIM; i++) {
    s[0] += M[0][i];
    s[1] += M[1][i];
    s[2] += M[2][i];
  }
  printf("%d %d %d\n", s[0], s[1], s[2]);
  return 0;
}
```

Rispondere alle seguenti domande:

1. Indicare la firma del programma, il significato dei dati in ingresso e quelli in uscita.
2. Descrivere cosa stampa a video il programma nel caso l'utente inserisca da tastiera il seguente input: `1 1 1 2 2 2 3 3 3`

Soluzione

1. la firma del programma è
$$(m_{0,0}, \ldots, m_{3,3}) \rightarrow (s_1, s_2, s_3)$$
2. I dati in ingresso sono gli elementi di una matrice m di dimensioni 3×3 inserita per righe.
3. s_1, s_2, s_3 sono le somme delle colonne di m.
4. I dati elaborati sono "3 6 9"

4.4 The Matrix

Si scriva un programma in linguaggio C con la seguente firma:

$$(\texttt{nr, nc, m(1,1), …, m(nr,nc)}) \rightarrow (\texttt{num})$$

Si assuma che i dati in ingresso, se ve ne sono, siano introdotti da tastiera e che i valori elaborati siano stampati a video.

DATI IN INGRESSO

- `nr`: numero di righe della matrice m (inserita di seguito)
- `nc`: numero di colonne della matrice m
- `m(i,j)`: valore della cella i, j della matrice m

DATI DA ELABORARE

- `num`: numero di sottomatrici 2x2 con media maggiore di 10

ULTERIORI VINCOLI E SPIEGAZIONI:

- I due valori interi `nr` e `nc` devono essere pari, maggiori di 0 e minori di `N`, dove `N` è una costante che deve essere opportunamente definita con valore 100. L'acquisizione deve essere ripetuta finché l'utente non inserisce valori corretti.
- Successivamente, il programma acquisisce da tastiera `nr` x `nc` valori interi e li memorizza in una matrice `m`, di dimensioni massime `N` x `N`, organizzandoli su un numero di righe pari a `nr` e un numero di colonne pari a `nc`.

ESEMPIO

Per esempio, si consideri il caso in cui l'utente inserisce la seguente matrice `m`:

$$m = \begin{bmatrix} 3 & 20 & 4 & 5 & 2 & 7 \\ 6 & 7 & 50 & 60 & 9 & 15 \\ 2 & 4 & 1 & 1 & 20 & 10 \\ 80 & 3 & 1 & 1 & 8 & 12 \end{bmatrix} \tag{4.1}$$

Questa matrice ha 4 righe e 6 colonne, ed è di fatto composta da 6 sotto-matrici 2x2

$$\begin{bmatrix} M_1 & M_2 & M_3 \\ M_4 & M_5 & M_6 \end{bmatrix} \tag{4.2}$$

ove, ad esempio, la sotto-matrice M_1 è pari a

$$M_1 = \begin{bmatrix} 3 & 20 \\ 6 & 7 \end{bmatrix} \tag{4.3}$$

ed ha media pari a 9. La seconda matrice (M_2) ha media 29.75, la terza 8.25, la quarta 22.25, la quinta 1 e la sesta 12.5. Per questo particolare esempio, il valore che deve essere stampato a video è quindi 3 poiché solo tre sotto-matrici hanno media maggiore di 10.

Soluzione

```c
int main() {
  int m[N][N], nRighe, nCol, i, j, cont = 0;
  float media;

  /* Acquisizione dei numeri di righe e colonne */
  do {
    printf("Inserire il numero di righe: ");
    scanf("%d", &nRighe);
  } while (nRighe <= 0 || nRighe >= N || (nRighe % 2 != 0));

  do {
    printf("Inserire il numero di colonne: ");
    scanf("%d", &nCol);
  } while (nCol <= 0 || nCol >= N || (nCol % 2 != 0));

  /* Acquisizione della matrice */
  for (i = 0; i < nRighe; i++) {
    for (j = 0; j < nCol; j++) {
      printf("Inserire l'elemento in posizione (%d, %d): ", i, j);
      scanf("%d", &m[i][j]);
    }
  }

  /* Conteggio delle sottomatrici con media maggiore di 10 */
  for (i = 0; i < nRighe; i = i + 2) {
    for (j = 0; j < nCol; j = j + 2) {
      media = (m[i][j] + m[i + 1][j] + m[i][j + 1] + m[i + 1][j + 1]) / 4.0;
      if (media > 10)
        cont++;
    }
  }
  printf("Ci sono %d sottomatrici con media maggiore di 10.", cont);
  return 0;
}
```

4.5 Base dati arresti

Richiesta

Si scriva un programma in linguaggio C con la seguente firma:

$$(\texttt{anno, codicecrimine}) \rightarrow (\texttt{risultato})$$

Si assuma che i dati in ingresso, se ve ne sono, siano introdotti da tastiera e che i valori elaborati siano stampati a video.

DATI IN INGRESSO

- `anno`: numero intero positivo
- `codicecrimine`: numero intero positivo

DATI DA ELABORARE

- `risultato`: numero intero maggiore o uguale a 0, calcolato come descritto di seguito

Spiegazioni e vincoli

Il `risultato` del programma è il numero massimo di arresti giornalieri dei nati per un particolare `anno` ed un particolare `codicecrimine` inserito dall'utente che deve essere calcolato utilizzando i dati contenuti nella variabile `database`, come descritta in seguito.

Strutture dati

```c
typedef struct {
  int giorno;
  int mese;
  int anno;
} data;

typedef struct {
  data data_di_nascita;
  int giorno_dell_anno;       /* Giorno in cui è avvenuto l'arresto, da 0 a 364 */
  int codice_crimine;
  stringa stringa_crimine;
} arresto;

arresto database[80];         /* Numero massimo di elementi arresto è 80 */
int arresti_per_giorno[365];
```

Ulteriori informazioni

- Il tipo `arresto` rappresenta le informazioni relative ad un evento di arresto, avvenuto in `giorno_dell_anno`, di una persona nata in una certa data (`data_di_nascita`).
- `database` contiene le informazioni degli arresti effettuati in un determinato anno (ad esempio 2017). Si assuma che tale array non sia ordinato e inoltre sia ***già stato riempito*** (non è necessario quindi assegnargli nessun valore, per questo esercizio)
- `arresti_per_giorno` è una variabile array ausiliaria (i cui elementi sono inizializzati tutti a `0`) utilizzabile per i calcoli intermedi.

Soluzione

```c
/* Si assume che database sia già stato inizializzato */

int main() {
    int anno, codicecrimine, risultato;
    int i;

    printf("Inserire anno: ");
    scanf("%d", &anno);

    printf("Inserire codicecrimine: ");
    scanf("%d", &codicecrimine);

    for(i=0; i<80; i++)
      if(database[i].data_di_nascita.anno == anno &&
        database[i].codice_crimine == codicecrimine)
        arresti_per_giorno[database[i].giorno_dell_anno] += 1;

    risultato = 0;
    for(i=0; i<365; i++)
      if(arresti_per_giorno[i] > risultato)
        risultato = arresti_per_giorno[i]

    printf("Massimo numero di arresti giornalieri: %d", risultato);
}
```

4.6 Twitters

Si scriva un programma in linguaggio C con la seguente firma:

$$(\texttt{anno, hashtag}) \rightarrow (\texttt{nome}_1\texttt{, …, nome}_n)$$

Si assuma che i dati in ingresso, se ve ne sono, siano introdotti da tastiera e che i valori elaborati siano stampati a video.

DATI IN INGRESSO

- `anno`: numero intero positivo
- `hashtag`: stringa

DATI DA ELABORARE

- $\texttt{nome}_j$: stringa corrispondente al nome del j-simo utente che rispetta la condizione descritta di seguito

ULTERIORI VINCOLI E SPIEGAZIONI:
La condizione che ciascun utente deve rispettare è di essere iscritto da `anno` e che abbia usato l'hashtag contenuto in `hashtag` in almeno uno dei propri messaggi. Si ricorda che per confrontare stringhe non è possibile usare l'operatore `==`; è invece necessario usare la funzione `strcmp` (libreria `string.h`). I risultati devono essere elaborati utilizzando i dati contenuti nella variabile `twitterdb`, definita da queste righe di codice che si assumono già date:

```c
typedef char stringa[30];

typedef struct {
    char contenuto[140];
    int  numero_hashtags;
    stringa hashtags[4];
} tweet;

typedef struct {
    stringa nome;
    int     data_iscrizione; /* anno di iscrizione a twitter */
    int     numero_messaggi; /* numero effettivo messaggi */
    tweet   messaggi[100];
} utente;

typedef struct {
    int numero_utenti;
    utente dati_utente[100];
} utenti;

utenti twitterdb;
```

Si assuma che `twitterdb` sia già stata inizializzata (non c'è quindi bisogno di assegnarle nessun valore)
ESEMPIO
Supponiamo che `twitterdb` sia già stato inizializzato in questo modo:

```c
utenti twitterdb = { 3, {
    { "vittorio", 2011 , 2, {
        { "ciao", 1, { "#poli" } },
        { "state sereni", 1, { "#poli"} }
        }  } ,
```

```
6       { "marco",    2017 , 1, { { "ah?" , 1, { "#poli" } }  } },
7       { "daniele",  2011 , 1, { { "eh?" , 1, { "#poli" } }  } }
8     }
9   };
```

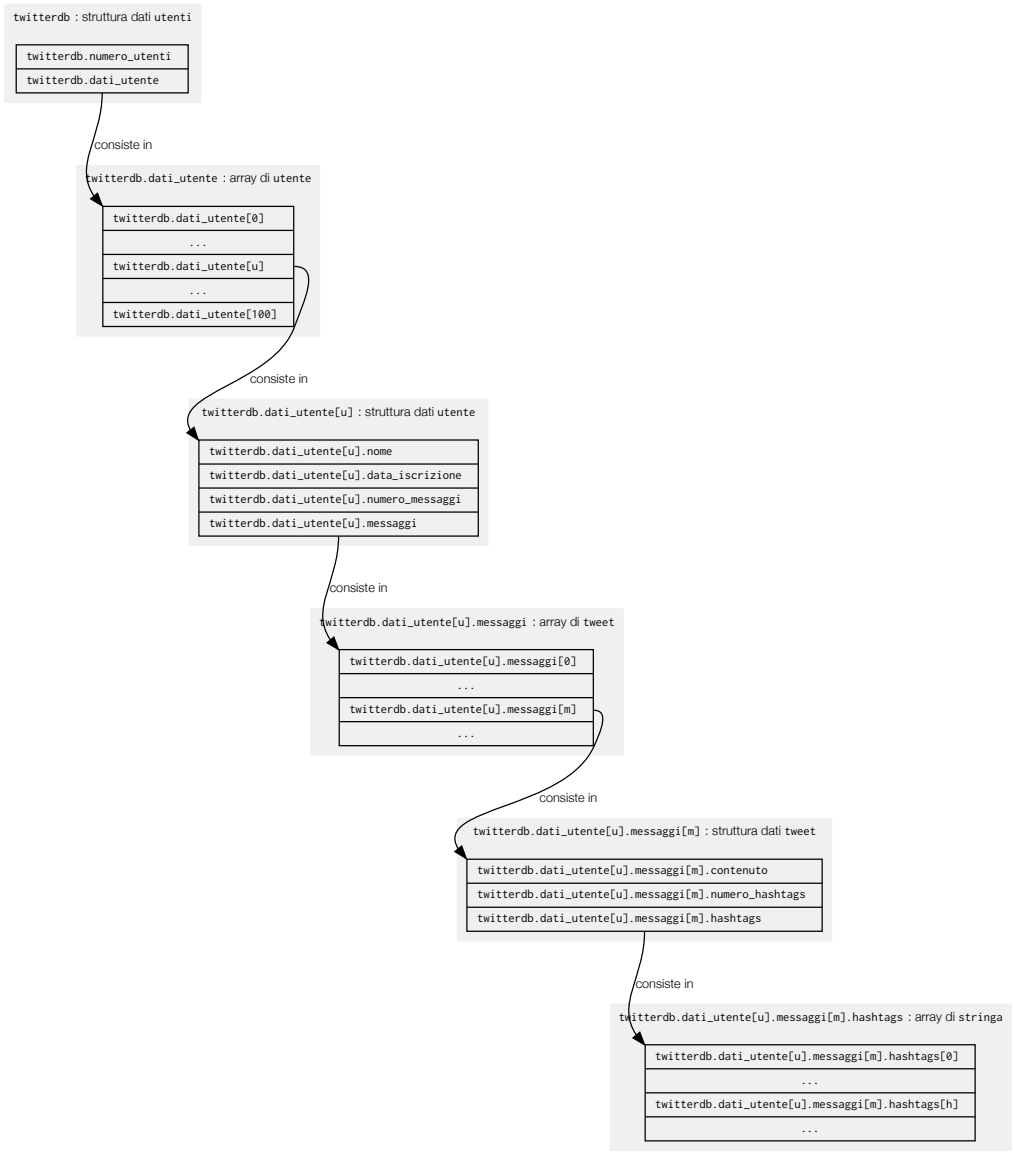
Una tipica sessione di utilizzo del programma è allora la seguente:

```
Inserisci l'anno di iscrizione: 2012
Inserisci l'hashtag: #poli

Dati trovati:
L'utente vittorio ha usato #poli
L'utente daniele ha usato #poli
```

Soluzione

Visualmente, i dati sono organizzati come nella seguente figura:



La struttura indica che vi sono 3 array (di utente, di tweet e di stringa) innestati. La soluzione quindi conterrà tre cicli innestati.

```
1  int main() {
2    int u, m, h;
3
4    int anno;
5    stringa hashtag;
6
7    printf("Inserisci l'anno di iscrizione: ");
8    scanf("%d", &anno);
9
```

```
10    printf("Inserisci l'hashtag: : ");
11    scanf("%s", hashtag);
12
13    for (u = 0; u < twitterdb.numero_utenti; u++) {
14      int trovato = 0;
15      utente ut = twitterdb.dati_utente[u];
16      if (ut.data_iscrizione < anno) {
17        for (m = 0; m < ut.numero_messaggi; m++) {
18          tweet ms = ut.messaggi[m];
19          for (h = 0; h < ms.numero_hashtags; h++) {
20            if (strcmp(ms.hashtags[h], hashtag) == 0) {
21              trovato = 1;
22            }
23          }
24        }
25      }
26      if (trovato == 1) {
27        printf("L'utente %s ha usato #poli \n", ut.nome);
28      }
29    }
30  }
```

4.7 Spoutify

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio C. A meno che non sia richiesto, non è necessario includere file headers di altre librerie o dichiarare un main. Inoltre, il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Il servizio online Spoutify fornisce musica in streaming sotto-forma di alcuni canali a tema. Ad ogni canale è associato un codice identificativo, un titolo ed una lista di canzoni. L'insieme di tutti i canali è contenuto in una variabile `canali` già inizializzata. Si assuma di avere poi un array `utenti` che contiene i dati degli utenti attualmente collegati, ovvero nome dell'utente ed il canale che sta ascoltando in quel particolare momento.

```c
#include <stdio.h>
#define NUM_CANALI 2
#define NUM_UTENTI 4
#define MAX_NUM_CANZONI 20

typedef char stringa[40];

typedef struct {
  int identificativo;
  stringa titolo;
  stringa canzoni[MAX_NUM_CANZONI];
  int numCanzoni;
} canale;

canale canali[NUM_CANALI] = {
    {111, "Il buongiorno si vede dal mattino",
     {"Symphony of destruction - Megadeth", "Paradise city - Guns n' roses"}, 2},
    {222, "Pizza e mandolino", {"Napoli - Nino d'angelo", "Napul'è - Pino Daniele"}, 2}};

typedef struct {
  stringa nomeUtente;
  int canale;
} utente;

utente utenti[NUM_UTENTI] = {{"Vittorio Zaccaria", 222},
                             {"Pippo Pippo", 222},
                             {"Peppone 'O Pizzaiolo", 222},
                             {"Mister metal", 111}};
```

Scrivere un programma che stampi a video il nome del canale ed il numero di utenti attualmente collegati. Ad esempio, per come sono state inizializzate le variabili `canali` e `utenti`, il programma dovrebbe stampare:

```
Il canale 'Il buongiorno si vede dal mattino' ha 1 utenti al momento.
Il canale 'Pizza e mandolino' ha 3 utenti al momento.
```

Soluzione

```c
int main() {
  int i, j;
```

```
3     for (i = 0; i < NUM_CANALI; i++) {
4       int sumUtenti = 0;
5       for (j = 0; j < NUM_UTENTI; j++) {
6         if (utenti[j].canale == canali[i].identificativo) {
7           sumUtenti++;
8         }
9       }
10      printf("Il canale '%s' ha %d utenti al momento.\n", canali[i].titolo,
11             sumUtenti);
12    }
13    return 0;
14  }
```

4.8 Audio

Si scriva un programma in linguaggio C con la seguente firma:

$$(\texttt{n, s(0), …, s(n-1),w}) \rightarrow (\texttt{t(0), …, t(n-1)})$$

Si assuma che i dati in ingresso, se ve ne sono, siano introdotti da tastiera e che i valori elaborati siano stampati a video.

DATI IN INGRESSO

- `n`: dimensione del vettore s
- $\texttt{s}_j$: j-simo elemento del vettore s; è un cosidetto *campione*, ovvero un numero con la virgola, che descrive l'intensità di un segnale audio al tempo j
- `w`: numero intero positivo, rappresenta la dimensione della "finestra" di segnale di cui calcolare la media

DATI DA ELABORARE

- $\texttt{t}_j$: j-simo elemento del vettore t (calcolato come descritto in seguito)

ULTERIORI VINCOLI E SPIEGAZIONI:
Il vettore s è una sequenza finita di n campioni (numeri) reali; il vettore t deve essere calcolato in modo tale che il valore del segnale t alla posizione i sia la media aritmetica dei valori di $s_{i-w} \ldots s_{i-1}$. Se $i - w < 0$ allora si consideri $t_i = s_i$.
ESEMPIO

```
Quanti valori del segnale s vuoi inserire? 7
Inserisci campione n. 0 = 0.42
Inserisci campione n. 1 = 0.91
Inserisci campione n. 2 = 5.24
Inserisci campione n. 3 = 3.07
Inserisci campione n. 4 = 1.11
Inserisci campione n. 5 = 9.50
Inserisci campione n. 6 = 1.50
Valore di w (finestra)? 3

Ecco la media del segnale
Campione n. 0: 0.42
Campione n. 1: 0.91
Campione n. 2: 5.24
Campione n. 3: 2.19
Campione n. 4: 3.073
Campione n. 5: 3.14
Campione n. 6: 4.56
```

dove, ad esempio, 2.19 = (0.42 + 0.91 + 5.24) / 3 e così via.

Soluzione

```c
#include <stdio.h>
#define MAX_LEN 20
int main() {
  int len;
  float s[MAX_LEN];
  float t[MAX_LEN];
```

```
7     float sum;
8     int w;
9     int i, j;
10    printf("Quanti valori del segnale s vuoi inserire? ");
11    scanf("%d", &len);
12    for (i = 0; i < len; i++) {
13      printf("Inserisci campione n. %d = ", i);
14      scanf("%f", &s[i]);
15    }
16    printf("Valore di w (finestra)");
17    scanf("%d", &w);
18    for (i = 0; i < w; i++) {
19      t[i] = s[i];
20    }
21    for (i = w; i < len; i++) {
22      sum = 0;
23      for (j = i - w; j < i; j++) {
24        sum += s[j];
25      }
26      t[i] = sum / w;
27    }
28    printf("Ecco la media del segnale\n");
29    for (i = 0; i < len; i++) {
30      printf("Campione n. %d: %f\n", i, t[i]);
31    }
32  }
```

4.9 Compagnia telefonica

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio C. A meno che non sia richiesto, non è necessario includere file headers di altre librerie o dichiarare un main. Inoltre, il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Il programma di gestione dei clienti della compagnia telefonica TT, utilizza le seguenti strutture dati per memorizzare i dati delle chiamate effettuate dai propri clienti ***nell'ultimo mese***:

```c
#define MAXCHIAMATE 1000
#define MAXCLIENTI 100

typedef char stringa[50];
typedef struct { int ora, minuti, durata; } chiamata;

typedef struct {
  stringa codicefiscale;
  chiamata chiamate[MAXCHIAMATE];
  int n_chiamate; /* numero chiamate effettuate */
} cliente;

typedef struct {
  cliente lista_clienti[MAXCLIENTI];
  int numero_clienti;
} archivio_clienti;

archivio_clienti db; /* si suppone già popolata */
```

Si ipotizzi che le telefonate abbiano il seguente costo:

- le telefonate iniziate dalle 22:00 alle 07:59 (estremi inclusi) hanno un costo di ***0.005 euro al secondo***
- tutte le altre costano ***0.01 euro al secondo***

Domande

1. Si dichiari un array di `float` di dimensione `MAXCLIENTI` e di nome `bolletta` e si scriva un frammento di codice C che riempia `bolletta`, in modo tale che la posizione `i`-esima di bolletta contenga il costo complessivo delle chiamate effettuate dal cliente `i`-esimo nell'ultimo mese:
2. Si dichiarari un array di nome `premium` contenente elementi di tipo `cliente` e riempirlo (senza lasciare spazi vuoti) con i dati dei clienti che hanno speso ***più di 100 euro*** nel corso dell'ultimo mese.

Soluzione

```c
int main() {
  float bolletta[MAXCLIENTI];
  for (int i = 0; i < db.numero_clienti; i++) {
    bolletta[i] = 0;
    for (int j = 0; j < db.lista_clienti[i].n_chiamate; j++) {
      int ora = db.lista_clienti[i].chiamate[j].ora;
```

```
7         int durata = db.lista_clienti[i].chiamate[j].durata;
8         if (ora >= 22 || ora < 8) {
9           bolletta[i] += durata * 0.005;
10        } else {
11          bolletta[i] += durata * 0.01;
12        }
13      }
14    }
15    cliente premium[MAXCLIENTI];
16    int i, k = 0;
17
18    for (i = 0; i < db.numero_clienti; i++) {
19      if (bolletta[i] > 100) {
20        premium[k++] = db.lista_clienti[i];
21      }
22    }
23  }
```

4.10 Carte di credito frodate

Si considerino i seguenti tipi di dato in C. Essi servono a contenere le informazioni relative ad un insieme di carte di credito `cards`, la cui dimensione massima è `MAXCARDS`.

```
#define MAXSTRINGLEN 100
#define MAXACQUISTI 100
#define MAXCARDS 100

typedef char stringa[MAXSTRINGLEN];
typedef enum { falso, vero } bool;

typedef struct {
  float importo;
  stringa nazione;

  int timestamp;
  /* tempo dell'acquisto espresso in secondi dal 1/1/1970.

     - le ore 9:00:00 del il 2015.06.29 corrispondono al 1435561200 secondo
     - le ore 9:01:00 del il 2015.06.29 corrispondono al 1435561260 secondo
  */

  bool usato_pin; /* determina se la transazione è avvenuta richiedendo il pin
                     all'utente */
} acquisto;

typedef struct {
  int card_number;             /* numero della carta */
  acquisto trans[MAXACQUISTI]; /* acquisti effettuati con la carta */
  int n_trans; /* numero delle transazioni eseguite */
} carta;

/* Si suppone che le seguenti variabili siano già state popolate */
carta cards[MAXCARDS];
int n_cards; /* Numeri effettivo di carte in cards */
```

1. Si definisca un nuovo tipo di struttura `Persona` atta a contenere le informazioni relative al proprietario della carta e si modifichi la definizione del tipo `Carta` in modo che contenga un campo `proprietario` di tipo `Persona`.

2. Si scriva un frammento di codice in linguaggio C per rilevare le carte che possono aver subito una frode. Una carta può aver subito una frode se *i)* riporta due transazioni consecutive in meno di 1 minuto, oppure *ii)* riporta due transazioni consecutive con PIN in nazioni diverse in meno di un'ora.

   In particolare: *1)* si elaborino e stampino i numeri delle carte che possono aver subito una frode considerando che le transazioni registrate nel campo `trans` di ciascuna carta siano ordinate cronologicamente e *2)* si copino le informazioni di tutte le persone frodate in un array `persone_frodate` (senza lasciare buchi).

Soluzione

```
typedef struct {
  stringa nome;
```

```
  stringa cognome;
} persona;

typedef struct {
  int card_number;             /* numero della carta */
  acquisto trans[MAXACQUISTI]; /* acquisti effettuati con la carta */
  persona proprietario;
  int n_trans; /* numero delle transazioni eseguite */
} carta;
```

```
int main() {
  int i, j, k, carta_frodata;
  persona persone_frodate[MAXCARDS];
  for (i = 0; i < n_cards; i++) {
    carta_frodata = falso;
    k = 0;
    acquisto t0 = cards[i].trans[j];
    acquisto t1 = cards[i].trans[j + 1];
    for (j = 0; (j < cards[i].n_trans - 1) && !carta_frodata; j++) {
      if (t1.timestamp - t0.timestamp < 60) {
        carta_frodata = vero;
      } else {
        if (t1.usato_pin && t0.usato_pin &&
            (t1.timestamp - t0.timestamp < 60 * 60) &&
            strcmp(t1.nazione, t0.nazione)) {
          carta_frodata = vero;
        }
      }
    }
    if (carta_frodata) {
      printf("\nla carta %d è stata carta_frodata ", cards[i].card_number);
      persone_frodate[k] = cards[i].proprietario;
      k++;
    }
  }
}
```

4.11 Operatori telefonici

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio C. A meno che non sia richiesto, non è necessario includere file headers di altre librerie o dichiarare un main. Inoltre, il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Un operatore telefonico è caratterizzato da un ***nome***, il ***paese in cui opera*** e le informazioni riguardanti ***i piani tariffari che offre*** (un operatore offre al massimo 20 piani differenti). Ogni piano tariffario, a sua volta, è caratterizzato da un ***nome***, il ***costo mensile del piano***, il numero di ***minuti mensili inclusi*** nel piano e il numero di ***GB di traffico dati mensili*** inclusi nel piano.
Si ipotizzi quindi di avere i tipi di strutture dati mostrati accanto e si risponda alle seguenti richieste:

1. Scrivere un frammento di codice in C che, ipotizzando che la variabile `db` sia stata precedentemente riempita con i dati di 100 operatori, copi nella variabile `sel` (senza lasciare spazi) i dati degli operatori che hanno almeno 3 piani tariffari con più di 2 GB di traffico dati inclusi;

2. Scrivere un frammento di codice C che stampi a video la percentuale degli operatori contenuti in `sel` che operano in Italia.

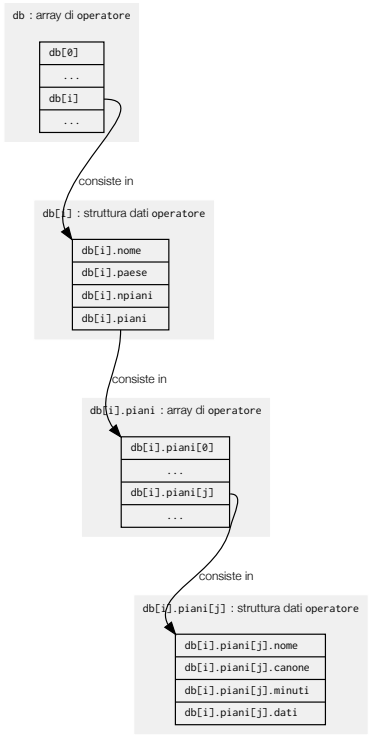
```c
typedef struct {
    char nome[50];
    float canone;
    int minuti;
    int dati;
} piano;

typedef struct {
    char nome[50];
    char paese[50];
    int npiani;
    piano piani[20];
} operatore;

operatore db[100], sel[100];
```

Soluzione

Visualmente, i dati sono organizzati come nella seguente figura:



L'organizzazione delle strutture dati suggerisce le seguenti soluzioni:

```
int i,j,k,n;

k = 0;
for (i=0; i<100; i++) {
    n=0;
    for (j=0; j<db[i].npiani; j++) {
        if (db[i].piani[j].dati > 2) {
            n++;
        }
    }
    if (n>=3) {
        sel[k] = db[i];
        k++;
    }
}
```

```
n=0;
for (i=0; i<k; i++)
    if (strcmp(sel[i].paese,"Italia")==0)
        n++;

printf("Pecent. Italiani: %f\n",(n*100.0)/k);
```

4.12 Feisbuk

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio C. A meno che non sia richiesto, non è necessario includere file headers di altre librerie o dichiarare un main. Inoltre, il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Siano date in linguaggio C le seguenti strutture dati, che rappresentano un database di utenti e relative foto, simile a quello di Facebook:

```c
#define MAX_PHOTOS 100
#define MAX_USERS 100
#define MAX_PHOTO_DIM 100
#define SOGLIA 180

typedef char stringa[100];

typedef struct {
  stringa link;
} photo_t;

typedef struct {
  stringa user_name;
  int anno_iscrizione;
  int num_photos;             /* numero di foto dell'utente */
  photo_t photos[MAX_PHOTOS]; /* foto fatte dall'utente */
} user_t;

user_t db_utenti[MAX_USERS];
int num_utenti;
```

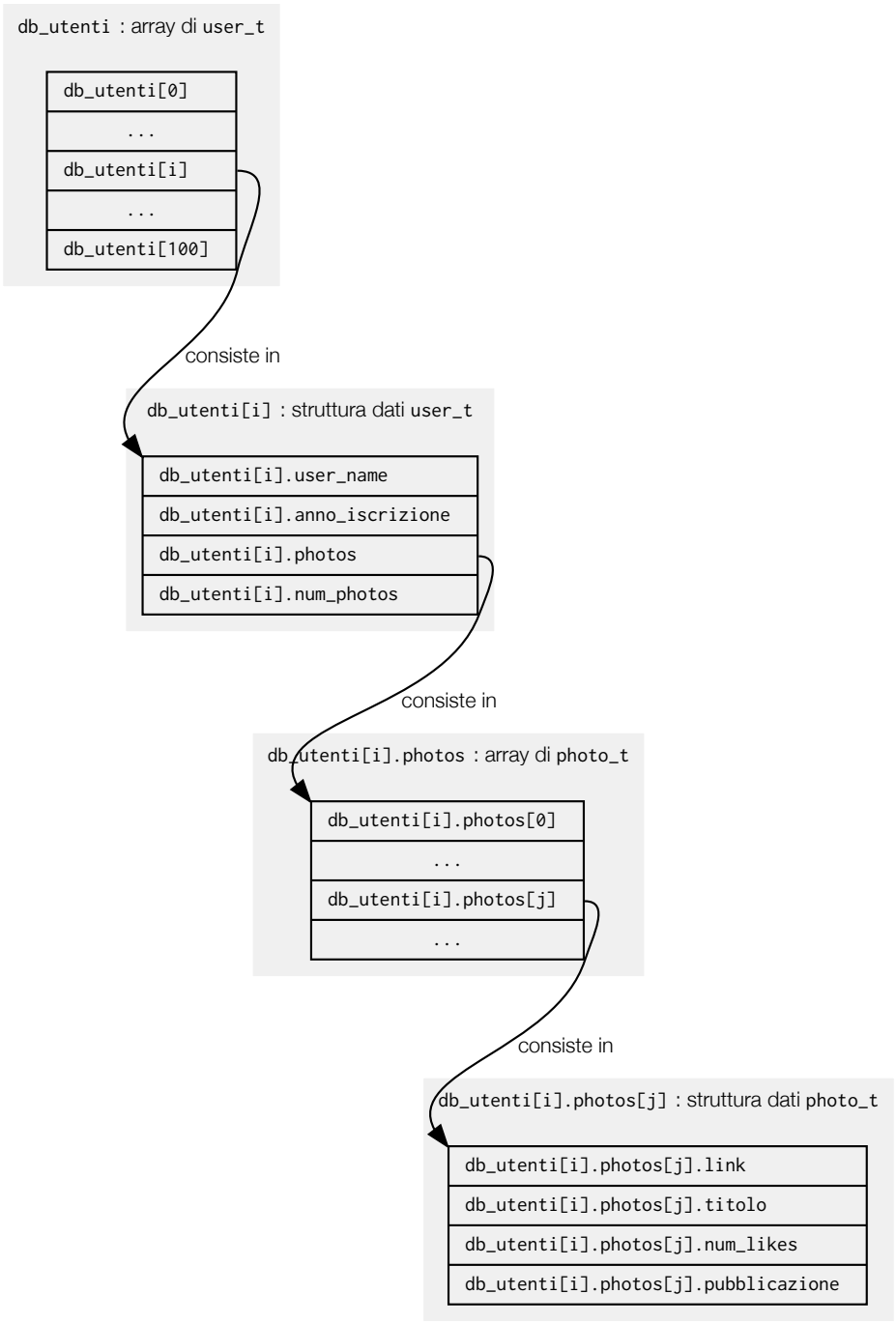
1. Si riscriva la definizione del tipo `photo_t` estendendola in modo tale che includa anche le informazioni relative al numero di gradimenti (anche detti *like*) che ciascuna foto ha ricevuto, il titolo e la data (giorno, mese, anno) di pubblicazione della foto.
2. Si supponga che `database` e `num_users` siano stati inizializzati. Definendo le aggiuntive variabili necessarie, scrivere una porzione di codice in linguaggio C che per ciascun utente stampi a video il titolo della sua foto che ha ricevuto il massimo numero di like, riportando il nome dell'utente, il titolo della foto e il numero di like che questa ha ricevuto.
3. Si estenda il tipo di foto `photo_t` in modo tale che contenga un'immagine quadrata ed un tag.
4. Si scriva una porzione di codice che, tra tutte le foto di tutti degli utenti stampi a schermo il titolo della foto che ha il maggior numero di elementi che superano il valore di `SOGLIA` ed il corrispondente `tag` sia "Montagna" oppure "Neve".

Soluzione

```c
typedef struct {
  int giorno;
  int mese;
  int anno;
} data;
```

```
6
7  typedef struct {
8    stringa link;
9    stringa titolo;
10   int num_likes;
11   data pubblicazione;
12 } photo_t;
```

Una volta introdotti i cambiamenti, la struttura risultante è la seguente:



L'organizzazione delle strutture dati suggerisce le seguenti soluzioni:

```
/* Utente con il massimo di like */
for (i = 0; i < num_utenti; i++) {
  max = 0;
  for (j = 0; j < db_utenti[i].num_photos; j++) {
    if (db_utenti[i].photos[j].num_likes > max) {
      max = db_utenti[i].photos[j].num_likes;
      posizione = j;
    }
  }
  printf("Utente %s: \n Foto: %s\n Numero di likes ricevuti %d \n\n",
         db_utenti[i].user_name, db_utenti[i].photos[posizione].titolo, max);
}
```

```
typedef struct {
  stringa link;
  stringa titolo;
  stringa tag;
  int num_likes;
  data pubblicazione;
  int immagine[MAX_PHOTO_DIM][MAX_PHOTO_DIM];
} photo_t;
```

```
stringa foto;

  int max_sum_soglia = 0;
  int sum_soglia;

  for (i = 0; i < num_utenti; i++) {
    for (j = 0; j < db_utenti[i].num_photos; j++) {
      stringa tag;
      strcpy(tag, db_utenti[i].photos[j].tag);
      if (strcmp(tag, "Montagna") == 0 || strcmp(tag, "Neve") == 0) {
        sum_soglia = 0;
        {
          for (int k = 0; k < MAX_PHOTO_DIM; k++)
            for (int h = 0; h < MAX_PHOTO_DIM; h++) {
              if (db_utenti[i].photos[j].immagine[k][h] > SOGLIA)
                sum_soglia++;
            }
        }
        if (sum_soglia >= max_sum_soglia) {
          strcpy(foto, db_utenti[i].photos[j].titolo);
          max_sum_soglia = sum_soglia;
        }
      }
    }
  }
  printf("%s", foto);
```

4.13 Discounted computers Inc.

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio C. A meno che non sia richiesto, non è necessario includere file headers di altre librerie o dichiarare un main. Inoltre, il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Si considerino le seguenti dichiarazioni di tipi e variabili che definiscono le informazioni relative ai laptop presenti nel catalogo di un distributore di materiale informatico:

```c
#define N 30

typedef char stringa[100];

typedef struct{
    int gb;
    int rpm; // round per minutes del disco fisso
} Hdd;

typedef struct{
    int mHz;
    int gb;
    stringa tipologia;
} Ram;

typedef struct{
    int ghz;
    Ram ram[4];
    int nBanchiRam; // numero effettivo di elementi dell'array `ram`
    Hdd hardDisk;
    stringa produttore;
} Laptop;

Laptop listino[N];
int numeroLaptop;
```

ove `listino` è un array di (massimo) N Laptops e `numeroLaptop` è il numero di laptop effettivi presenti in `listino`.

Domande

1. Si copi all'interno della variabile `selezione`:

   ```
   Laptop selezione[N]
   ```

   tutti i laptop di marca uguale al migliore (trovato al punto precedente), senza lasciare buchi.

2. Si definisca un tipo `cache` con le seguenti caratteristiche:

   - può essere di due tipologie differenti: `L1` oppure `L2`
   - è caratterizzata dai parametri visti a lezione per quanto riguarda il tempo di accesso

3. Si estenda il tipo `Laptop` in modo tale che possa contenere anche le informazioni sulle cache in esso contenuta. Si tenga presente che le cache possono essere più di una.

Soluzione

```
int j = 0;
for(i=0; i < numeroLaptop; i++) {
    if(strcmp(listino[i].produttore, listino[n].produttore) == 0) {
        selezione[j] = listino[i];
        j++;
    }
}
```

```
typedef enum { L1, L2 } cache_type;

typedef struct {
    cache_type tipo;
    float hit_time;
    float miss_penalty;
    float miss_rate;
} cache;
```

```
typedef struct{
    int ghz;
    Ram ram[4];
    int nBanchiRam; // numero effettivo di elementi dell'array `ram`
    Hdd hardDisk;
    stringa produttore;

    cache memorieCaches[4];
    int nMemorieCaches;

} Laptop;
```

4.14 Taxi

Richiesta

Umber vi ha incaricato di sviluppare in C il loro sistema di gestione dei taxi all'interno di un'area urbana. In particolare, vi ha chiesto di progettare le strutture dati necessarie per immagazzinare le seguenti informazioni:

- *Dati relativi a tutte le le città servite dal servizio taxi*. Ogni città è caratterizzata da un ***nome*** e ***una lista zone*** (massimo 10). Umber gestisce fino a un massimo di 100 città.
- *Dati relativi alle zone all'interno di una città*. Ogni zona è identificata da un ***numero univoco*** e contiene la ***lista dei taxi*** che si trovano in quella zona, fino a un massimo di 50 taxi.
- *Dati relativi ai taxi*. Ogni taxi è identificato da un ***numero univoco*** ed è caratterizzato dal ***numero di passeggeri*** che può portare.

Richiesta 2

Assumendo che `listaCitta` sia una variabile già popolata contenente la lista delle città servite da Umber

```
Citta listaCitta[MAX_CITTA];
```

e che le seguenti variabili siano già riempite:

```
Stringa citta;
int nZona;
int nPasseggeri;
```

Sviluppate un frammento di programma che stampa a schermo il codice numerico di un taxi che si trova nella zona indicata e che ha a disposizione un numero di posti maggiore o uguale al numero di passeggeri da trasportare.

Soluzione

```
typedef struct {
   int numeroTaxi;
   int nPosti;
} Taxi;

typedef struct {
  int numeroZona;
  Taxi listaTaxi[MAX_TAXI];
  int numeroTaxi;
} Zona;

typedef struct {
   Stringa nome;
   Zona listaZone[MAX_ZONE];
   int numeroZone;
} Citta;

int i, j, k;
```

```
19    for(i = 0; i< nCitta; i++ ) {
20      for(j = 0; j < listaCitta[i].numeroZone; j++) {
21        for(k = 0; k < listaCitta[i].listaZone[j].numeroTaxi; k++) {
22          if(strcmp(listaCitta[i].nome, citta) == 0 &&
23             listaCitta[i].listaZone[j].numeroZona == 2 &&
24             listaCitta[i].listaZone[j].listaTaxi[k].nPosti >= nPasseggeri)  {
25            printf("Trovato taxi numero %d - Citta %s, Zona %d",
26                   listaCitta[i].listaZone[j].listaTaxi[k].numeroTaxi, citta, nZona);
27            return 0;
28          }
29        }
30      }
31    }
32    printf("Mi dispiace, non ho trovato taxi");
33    return 0;
```

4.15 Farmaci

Richiesta n. 1

Si scriva un programma in linguaggio C per la gestione del magazzino di una farmacia mantiene i dati sui farmaci disponibili in una variabile `db`. La variabile `db` consiste in un array di elementi di tipo `Farmaco`. Il numero effettivo di valori contenuti in tale array è memorizzato nella variabile `numeroFarmaci`.
Il tipo `Farmaco`, a sua volta, mantiene informazioni sul nome del farmaco (`nomeFarmaco`), numero di scatole disponibili (`nscatole`) e sulle informazioni di ciascuna scatola (`scatole`).
Le informazioni relative a ciascuna scatola sono rappresentate con il tipo `Scatola`. Questo tipo mantiene informazioni sulla quantità (`quantita`) del farmaco presente nella scatola (uno stesso farmaco può essere disponibile in scatole che contengono quantità diverse di medicinale, per esempio, farmaco Marmellad disponibile in scatole da 15 o 30 pillole) e il mese di scadenza della scatola, definito utilizzando un seguente tipo enumerativo.

Soluzione

```c
#define N 100
#define M 1000

typedef char stringa[30];

typedef enum { gen, feb, mar, apr, mag, giu, lug, ago, set, ott, nov, dic } meseScadenza;

typedef struct {
    int quantita;
    meseScadenza scadenza;
} Scatola;

typedef struct {
    stringa nomeFarmaco;
    int nscatole;
    Scatola scatole[N];
} Farmaco;

int numeroFarmaci;
Farmaco db[M];
```

Richiesta n. 2

Si scriva una porzione di codice in linguaggio C che, ipotizzando di avere il nome di un farmaco nella variabile stringa `nomeF` e un mese specificato in una variabile `mese`, stampi a video la somma totale delle quantità del farmaco la cui scadenza e' posteriore al valore della variabile mese. Si supponga che tutte le variabili necessarie siano state opportunamente inizializzate. In particolare che la variabile `db` contenga un numero di dati validi di tipo `Farmaco` pari al valore contenuto in `numeroFarmaci`, e che `nomeF` e `mese` siano state inizializzate con il nome del farmaco di cui verificare la disponibilità e il mese di scadenza.

Esempio

Se `nomeF` = “Marmellad” e `mese` = lug e se supponiamo di avere fra i farmaci nella variabile `db` il farmaco:

```
nomeFarmaco = "Marmellad"
nscatole = 3
scatole[0] = { quantita: 20, scadenza: apr }
scatole[1] = { quantita: 15, scadenza: ott }
scatole[2] = { quantita: 30, scadenza: ago }
```

il programma dovrà stampare:

```
Quantità di Marmellad disponibile dopo il mese 7: 45
```

Poiché solo una scatola da 30 e una da 15 unità non saranno ancora scadute dopo luglio.

Soluzione

```
int quantita_tot = 0;
Farmaco f;
for(int i = 0; i < numeroFarmaci; i++)
  if(strcmp(db[i].nomeFarmaco, nomeF) == 0) {
    f = db[i];
    for(int j = 0; j < f.nscatole; j++)
      if(f.scatole[j].scadenza > mese)
        quantita_tot += f.scatole[j].quantita;
  }
printf("Quantità di %s disponibile dopo il mese %d: %d\n", nomeF, mese, quantita_tot);
```

4.16 Azienda

Introduzione

Le seguenti dichiarazioni definiscono tipi di dati che descrivono i candidati per l'assunzione in un'azienda.

```
typedef enum {italiano, inglese, spagnolo, francese,
  tedesco, cinese, portoghese } lingua;

typedef struct {
  lingua l;
  int livello; /* numero da 1 a 5 con 1=basso 5=alto */
} linguaParlata;

typedef struct {
  stringa nome, cognome, diploma, laurea;
  linguaParlata lingue[5];
  int nLauree;
  int anniEsperienza;
} persona;

persona persone[40];
persona personeScelte[40];
```

Richiesta

Si scriva un frammento di codice, che includa eventualmente anche le dichiarazioni di ulteriori variabili e tipi, che copi nella parte iniziale del vettore `personeScelte` (senza lasciare buchi) le persone che parlano inglese con un livello superiore a 3 oppure soddisfano entrambi i seguenti requisiti: ***hanno non meno di una laurea e un numero di anni di esperienza non inferiore a tre***.

Soluzione

```
typedef enum {
  italiano,
  inglese,
  spagnolo,
  francese,
  tedesco,
  cinese,
  portoghese
} lingua;

typedef char stringa[30];

typedef struct {
  lingua l;
  int livello; /* numero da 1 a 5 con 1=basso 5=alto */
} linguaParlata;

typedef struct {
  stringa nome, cognome, diploma, laurea;
```

```
20    linguaParlata lingue[5];
21    int nLauree;
22    int anniEsperienza;
23  } persona;
24
25  void main() {
26    persona persone[40];
27    persona personeScelte[40];
28    int i, s, l;
29    s = 0;
30    for (i = 0; i < 40; i++) {
31      if (persone[i].nLauree >= 1 && persone[i].anniEsperienza >= 3) {
32        personeScelte[s] = persone[i];
33        s++;
34      } else {
35        for (l = 0; l < 5; l++) {
36          if (persone[i].lingue[l].l == inglese &&
37              persone[i].lingue[l].livello > 3) {
38            personeScelte[s] = persone[i];
39            s++;
40          }
41        }
42      }
43    }
44  }
```

4.17 Autofficina

Introduzione

In un'officina informatizzata ogni auto viene registrata all'ingresso ed inserita in una coda da 0 a massimo 20 macchine (altrimenti non viene presa in carico) poi passa ad uno dei seguenti reparti:

- diagnosi
- manutenzione ordinaria
- manutenzione straordinaria
- uscita

Dove viene eseguito l'intervento corrente (tagliando, marmitta, cinghia, cilindri, freni). Le informazioni relative all'evento corrente sono corredate da quelle degli interventi precedenti (massimo 100).

Richiesta

- Si definisca in C un tipo di struttura dati `auto` adeguato per rappresentare tutte le informazioni sull'auto.
- Si supponga che la coda delle auto in attesa sia definita in questo modo:

```
typedef struct {
  auto elenco[20];
  int primaInCoda;  /* indica la posizione della prima auto in coda */
  int ultimaInCoda; /* indica la posizione dell'ultima auto in coda */
} codaAuto;
codaAuto lista;
```

  Si scriva il frammento di codice C che aggiorna opportunamente il campo `primaInCoda` in modo tale la coda sia ridotta di un elemento ogni qualvolta una macchina è presa in carico.

Soluzione

```
typedef char stringa[8];
typedef enum { inDiagnosi, inManutOrd, inManutStraord, inUscita } stato;
typedef enum { tagliando, marmitta, cinghia, cilindri, freni } tipoIntervento;
typedef struct {
  stringa targa;
  stato s;
  tipoIntervento interventoCorrente;
  tipoIntervento interventiPrec[100];
} auto;
typedef struct {
  auto elenco[20];
  int primaInCoda;  /* indica la posizione della prima auto in coda */
  int ultimaInCoda; /* indica la posizione dell'ultima auto in coda */
} listaAuto;
listaAuto lista;

void main() {
```

```
18    if (lista.primaInCoda != lista.ultimaInCoda) {
19      lista.primaInCoda = lista.primaInCoda - 1;
20      if (lista.primaInCoda < 0)
21        lista.primaInCoda = 19;
22    }
23  }
```

4.18 War–

Introduzione

Il gioco ”War–” è un gioco di fortuna a carte fra a due giocatori; esso funziona come segue. A ogni giocatore vengono date un numero `N` costante di carte estratte a caso da un mazzo. Le carte di ciascun giocatore sono coperte messe una sopra l'altra.
Durante una mano del gioco, entrambi i giocatori estraggono la prima carta del mazzo e la girano. Se le carte sono uguali (in termini di valore da 1 a 10) vengono reinserite nel mazzo di ciascun giocatore come ultime.
Se invece una è maggiore dell'altra, il giocatore con la carta più alta prende quella più bassa e inserisce entrambe le carte in fondo al proprio mazzo.

Richieste

1. Domanda 1

   Si implementi la struttura dati associata allo stato del gioco

2. Domanda 2

   Si scriva una porzione di programma C che inizializzi le carte in maniera casuale e simuli il gioco fra due giocatori con le stesse.

Soluzione

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#define CARDS 5

typedef struct {
  char name[100];
  int cards[CARDS * 2];
  int nocards;
} player;

player first, second;


int main() {
  time_t t;
  srand((unsigned)time(&t));
  for (int i = 0; i < CARDS; i++) {
    first.cards[i] = rand() % 10;
    second.cards[i] = rand() % 10;
  }
  strcpy(first.name, "Vittorio");
  strcpy(second.name, "Chiara");
  first.nocards = CARDS;
  second.nocards = CARDS;
  while (1) {
    if (first.nocards == 0) {
      printf("First player %s loses match!\n", first.name);
      break;
    }
```

```
32      if (second.nocards == 0) {
33        printf("Second player %s loses match\n", second.name);
34        break;
35      }
36      int fsttop = first.cards[0];
37      int sndtop = second.cards[0];
38
39      if (fsttop == sndtop) {
40        for (int i = 1; i < first.nocards; i++)
41          first.cards[i - 1] = first.cards[i];
42        first.cards[first.nocards] = fsttop;
43
44        for (int i = 1; i < second.nocards; i++)
45          second.cards[i - 1] = second.cards[i];
46        second.cards[second.nocards] = sndtop;
47
48      } else {
49        player winner, loser;
50        if (fsttop > sndtop) {
51          winner = first;
52          loser = second;
53        } else {
54          winner = second;
55          loser = first;
56        }
57        for (int i = 1; i < loser.nocards; i++) {
58          loser.cards[i - 1] = loser.cards[i];
59        }
60        for (int i = 1; i < winner.nocards; i++) {
61          winner.cards[i - 1] = winner.cards[i];
62        }
63        winner.cards[winner.nocards - 1] = fsttop;
64        winner.cards[winner.nocards] = sndtop;
65        loser.nocards -=1;
66        winner.nocards += 1;
67        if (fsttop > sndtop) {
68          first = winner;
69          second = loser;
70        } else {
71          second = winner;
72          first = loser;
73        }
74      }
75    }
76  }
```

4.19 Parcheggio

Introduzione

Si vuole progettare il sistema informatico di un parcheggio per auto e ciclomotori che è collegato al semaforo di ingresso del parcheggio stesso. Il parcheggio è automatico ovvero, al verde del semaforo, il conducente lascia la macchina ad una piattaforma meccanica che posiziona la sua macchina in un posto assegnato dal sistema. Purtroppo, solo un veicolo alla volta può essere gestito dal sistema. Inoltre, un posto può essere occupato o da un'auto o da due ciclomotori.

Richiesta 1

Si definiscano i seguenti dati:

- il tipo di dato `veicolo`, che contiene la targa, il tipo di veicolo (`auto`, `moto`) e lo `stato` del veicolo (se `parcheggiata` oppure in `coda`).
- il tipo di dato `posto`, che contiene il numero del posto, il numero di veicoli occupanti, ed un array di `veicolo`
- il tipo di dato `parcheggio`, che contiene un array di `posto`, un array di `veicolo` in coda in ingresso e una variabile booleana che indichi se il parcheggio è correntemente impegnato nel posizionamento di una macchina. Si dimensioni gli array con un valore massimo ma si considerino le opportune variabili aggiuntive per considerare il numero di elementi effettivi

Richiesta 2

Si assuma di avere una variabile `park` di tipo parcheggio. Si scriva un programma che stampi lo stato che deve avere il semaforo dato lo stato del parcheggio, ovvero `rosso` se non vi sono posti per il primo veicolo in coda, giallo se ci sono posti ma il parcheggio è impegnato in un posizionamento, verde altrimenti.

Richiesta 3

Si assuma che sia stato trovato un posto in posizione `np`. Si scriva una porzione di codice che rimuova il veicolo in testa alla coda e lo posizioni nel posto assegnato all'interno del parcheggio. Si aggiorni lo stato del veicolo di conseguenza.

Soluzione

```
typedef char stringa[100];
typedef enum { automobile, moto } tipo_veicolo;
typedef enum { coda, manovra, parcheggiata } stato_veicolo;
typedef enum { false, true } bool;

typedef struct {
  stringa targa;
  tipo_veicolo tipo;
  stato_veicolo stato;
} veicolo;

typedef struct {
  int nposto;      /* numero del posto */
  int nvo;         /* quanti veicoli ci sono */
  veicolo vo[2];   /* veicoli parcheggiati */
} posto;

typedef struct {
  posto pd[MAX];   /* posti del parcheggio */
```

```
  int npd;         /* numero posti effettivi */

  veicolo vc[MAX]; /* veicoli in coda */
  int nvc;         /* numero veicoli in coda effettivi */

  bool pi;         /* parcheggio impegnato? */
} parcheggio;
```

```
int np;
if (park.nvc != 0) {
  int trovato = false;
  if (park.vc[0].tipo == automobile) {
    for (int i = 0; i < park.npd; i++) {
      if (park.pd[i].nvo == 0) {
        trovato = true;
        np = i;
      }
    }
  } else {
    for (int i = 0; i < park.npd; i++) {
      if (park.pd[i].nvo == 1 && park.pd[i].vo[0].tipo == moto) {
        trovato = true;
        np = i;
      }
    }
  }
  if (trovato && !park.pi) {
    printf("semaforo: verde");
  } else {
    if (park.pi) {
      printf("semaforo: giallo");
    } else {
      printf("semaforo: rosso");
    }
  }
} else {
  printf("semaforo: rosso");
}
```

```
park.vc[0].stato = parcheggiata;
park.pd[np].vo[park.pd[np].nvo++] = park.vc[0];
for(int i=0; i<(park.nvc - 1); i++) {
  park.vc[i] = park.vc[i+1];
}
park.nvc--;
```

Capitolo 5

Esercizi su linguaggio C consigliati

I seguenti sono esercizi semplici che possono essere verificati direttamente al calcolatore. Fallire i primi esercizi indica problemi gravi con la preparazione per l'esame.

1. Qual'è il valore della variabile `b` al termine di questo frammento di programma C?

```
int a;
int b;
a = 3;
b = 2*a;
a = 6;
```

2. Qual'è il valore della variabile `b` al termine di questo frammento di programma C?

```
int a;
int b;
a = 3;
b = 2*a;
a = 6;
b = b*a;
```

3. Qual'è il valore della variabile `b` al termine di questo frammento di programma C?

```
int a;
int b=0;
a = 3;
while(a--) b++;
```

4. Qual'è il valore della variabile `b` al termine di questo frammento di programma C?

```
int a;
int b=0;
a = 3;
while(--a) b++;
```

5. Qual'è il valore della variabile `b` al termine di questo frammento di programma C?

```
int a;
int b=1;
a = 3;
while(--a && b) b--;
```

6. Qual'è il problema con questo programma C?

```
int a;
int b;
a = 3;
b = 2*b*a;
a = 6;
b = b*a;
```

7. Qual'è il valore della variabile `b` al termine di questo frammento di programma C?

```
int a;
int b=1;
for(a=4; a>0; a--) b++;
```

8. Qual'è il valore della variabile `b` al termine di questo frammento di programma C?

```
int a;
int b=1;
for(a=4; a>=0; a--) b++;
```

9. Scrivere un programma che data una stringa `s` da tastiera, la copia in un'altra stringa `d` senza usare la funzione `strcpy`.
10. Scrivere un programma che data una stringa `s` da tastiera, ne calcola e stampa la lunghezza senza usare la funzione `strlen`.
11. Scrivere un programma che date due stringhe `nome` e `cognome` lette da tastiera, le concatena in un unica stringa `nomeECognome` la stampa a video senza usare la funzione `strcat`.
12. Scrivere un programma che dato un vettore di 5 stringhe lette da tastiera, lo ordina usando bubble sort, `strcmp` e `strcpy` (si ricordi che non è possibile assegnare fra di loro stringhe con l'operatore di assegnamento).
13. Modificare il programma che determina se una stringa è palindroma in modo tale che ignori spazi eventualmente presenti nella stringa stessa. Ad esempio la stringa `occorre portar aratro per rocco` è palindroma se gli spazi vengono
14. Scrivere un programma che data una stringa `s` calcoli la frequenza di tutti i caratteri e la stampi a video.
15. Scrivere un programma che legga una serie di caratteri da tastiera e, solo se questi sono tutti numeri, converta il numero corrispondente in una variabile di tipo `int` e la stampi (non si considerino numeri negativi).
16. Scrivere un programma come quello precedente ma considerando che può essere presente un meno (`-`) prima della prima cifra e che quindi il numero possa essere negativo.

17. Scrivere un programma come quello precedente ma considerando che può essere un separatore di decimali e che quindi scriva il numero corrispondente in una variabile di tipo `float`.

18. Scrivere un programma che lette le dimensioni di una matrice ed il valore di ciascun suo elemento determini se questa sia ”diagonale”.

19. Scrivere un programma che lette le dimensioni di una matrice ed il valore di ciascun suo elemento determini se questa sia ”simmetrica”.

Capitolo 6

Esercizi di introduzione a Matlab

6.1 Introduzione a find

Questo esercizio è, pratica, un mini tutorial matlab sulle potenzialità della funzione find. Utilizzeremo un file dati disponibile nelle installazioni Matlab che corrisponde ad una lista di pazienti:

```
load('patients')
```

Questo comando trova pazienti che fumano con eta' maggiore dei 40:

```
>> Smoker(find(Age >= 40))

ans =

     0
     0
     0
     0
     0
     0
     0
     1
     ...
```

Questo comando trova la percentuale dei pazienti piu' vecchi di 40 anni che fumano:

```
>> fum40 = Smoker(find(Age >= 40))
>> sum(fum40)/size(fum40,1)

ans =

    0.3636
```

Questo comando trova i cognomi dei pazienti che fumano e hanno piu' di 40 anni:

```
>> LastName(find(Age>=40 & Smoker))

ans =

    'Martin'
    'Lee'
    'Wright'
    'Baker'
    'Roberts'
    'Collins'
    'Stewart'
    'Sanchez'
    'Morris'
    'Reed'
    'Bell'
    'Ramirez'
    'Watson'
    'Hughes'
    'Russell'
    'Diaz'
```

Ulteriori esercizi

1. Creare una funzione `fuma(age)` che, dato un valore di eta, ritorna la percentuale calcolata all'inizio dell'esercizio. Si consiglia di usare la funzione `evalin('base', nomevar)` per accedere alle variabili del workspace base di Matlab.

2. Applicare la funzione di cui sopra ad un range di età e plottarla. Si consiglia di usare la funzione `arrayfun(@nomefunzione, array)` per applicare la funzione `nomefunzione` a ciascun valore di un array (ricordarsi la chiocciola `@` di fronte al nome).

6.2 Radice quadrata iterativa

In questo esercizio viene richiesto di ri-scrivere in linguaggio Matlab la soluzione in linguaggio C di un esercizio già dato precedentemente. Si cerchi di sfruttare il più possibile i costrutti offerti da Matlab per rendere sintetico il programma. Infine, si ricorda che il testo potrebbe non dichiarare esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; in quel caso tali informazioni sono infatti da dedurre dal testo stesso.

Calcolare e stampare l'intero più vicino alla radice quadrata di un numero `n` dato dall'utente

Soluzione

Vedremo due versioni, la prima semplice e la seconda più complessa. La versione semplice permette di calcolare l'intero r tale che $r^2 \leq n$:

```
n = input('numero di cui calcolare la radice quadrata approssimata: ');
v = 1:n;
r = size(find(v .* v <= n),2);
fprintf("la radice quadrata approssimata è %g\n", r);
```

Sappiamo però che non sempre l'intero più vicino alla radice quadrata di un numero è minore della stessa (si consideri il caso in cui $n = 8$ oppure $n = 48$). Si rende quindi necessario trovare il numero che minimizza la distanza assoluta tra il proprio quadrato e il numero n dato, ovvero:

```
n = input('numero di cui calcolare la radice quadrata approssimata: ');
v = 1:n;
dd = abs(n - v .* v);
r = find(dd == min(dd));
fprintf("la radice quadrata approssimata è %g\n", r);
```

6.3 Stringhe palindrome

In questo esercizio viene richiesto di ri-scrivere in linguaggio Matlab la soluzione in linguaggio C di un esercizio già dato precedentemente. Si cerchi di sfruttare il più possibile i costrutti offerti da Matlab per rendere sintetico il programma. Infine, si ricorda che il testo potrebbe non dichiarare esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; in quel caso tali informazioni sono infatti da dedurre dal testo stesso.

Si scriva un programma in Matlab per determinare se una stringa introdotta dall'utente sia palindroma oppure no. ***Non è possibile usare cicli***.

Soluzione

```
parola = input('inserisci parola: ')
if all(s == flip(s))
    disp('palindroma')
else
    disp('non palindroma')
end
```

6.4 Troviamo l'errore

Si consideri la seguente richiesta:

> Un programma deve controllare che un numero N inserito dall'utente sia: primo, dispari, compreso tra 3 e 100, estremi inclusi. Quando il numero N inserito non soddisfa tutte le condizioni sopra, l'inserimento viene ripetuto.

Vittorio, un programmatore alle prime armi, ha scritto il seguente frammento di codice Matlab, sfruttando la funzione `primo` che restituisce 1 se il suo argomento è un numero primo, 0 altrimenti:

```
N = 0
while(primo(N) && (3<=N) && (N<=100) && mod(N,2))
 N = input('Inserire un numero: ');
end
```

Domanda 1

La condizione del ciclo `while` è chiaramente sbagliata rispetto alla richiesta iniziale; tuttavia è possibile modificarla leggermente in modo tale da renderla coerente con la richiesta iniziale. Si riscriva la condizione di cui sopra ***aggiungendo solo quello che manca e senza cancellare nulla di esistente***:

Domanda 2

L'espressione risultante al passo precedente potrebbe essere *ridondante* ovvero vi potrebbero essere alcune operazioni che, anche se eliminate, non cambierebbero la funzionalità del ciclo `while` corrispondente. Quale eliminereste?

```
 Eliminerei 3<=N
 Eliminerei N<=100
 Eliminerei primo(N)
 Eliminerei mod(N,2)
 Non eliminerei nulla, non c'è nulla di ridondante
```

Soluzione

La condizione corretta è la seguente:

```
(primo(N) && (3<=N) && (N<=100) && mod(N,2)) == 0
```

dalla quale va eliminato `mod(N,2)` poiché tutti i `primo(N)` sono sicuramente dispari, per N>=3.

6.5 Comprensione programma

Si consideri il seguente programma Matlab

```
a = 0;
for b = 1:5
  a = a + b;
  if (mod(a, 2) == 0 )
     c = 0;
     while(c < b)
        a = a + c;
        c = c + 1;
     end
  end
  fprintf('a = %d, b = %d\n', a, b);
end
```

Si determini l'output a video del programma

Soluzione

```
a = 1, b = 1
a = 3, b = 2
a = 9, b = 3
a = 13, b = 4
a = 28, b = 5
```

6.6 Script

Introduzione

Si supponga di avere nel `main workspace` di Matlab la seguente matrice `temperature` che rappresenta le temperature giornaliere rilevate da una stazione meteo:

$$\text{temperature} = \begin{bmatrix} 1 & 3 & 2013 & 18.1 \\ \dots & & & \\ 10 & 4 & 2016 & 23.1 \\ 11 & 4 & 2016 & 24.1 \\ \dots & & & \end{bmatrix} \tag{6.1}$$

ove le colonne indicano, rispettivamente, il giorno, il mese, l'anno e la temperatura misurata.

Richiesta

Si implementi uno script MATLAB che costruisce un vettore `sintesi` che contenerrà 12 elementi, uno per ciascun mese dell'anno, contenenti il valore medio di tutte le temperature rilevate in quel mese per tutti gli anni considerati nella matrice. Per calcolare il valore medio di un array è possibile utilizzare la funzione di libreria `mean`.

Soluzione

```
1 for i=1:12
2         selez = temperature(:,2)==i;
3         sintesi(i) = mean(temperature(selez,4));
4 end
```

Capitolo 7

Esercizi Matlab - Funzioni non ricorsive

7.1 Matrici

In questo esercizio viene richiesta la scrittura di una funzione che, dati alcuni parametri in ingresso, calcola un valore risultante ritornato al chiamante (sia esso l'utente oppure un altro programma). Per controllare di aver eseguito l'esercizio correttamente, scrivete la funzione in un file Matlab e provate ad invocarla direttamente da linea comando passando i valori di esempio e controllando che ritorni effettivamente il valore aspettato.

Richiesta

Si implementi in MATLAB una funzione che svolga le seguenti operazioni:

- Riceve in ingresso due matrici `A` e `B` di `M=x=N` elementi.
- Produce una terza matrice `C` ottenuta da `A` e `B` secondo la seguente regola:

$$C(r,c) = \begin{cases} A(r,c), & \text{se } B(r,c) < \text{minimo valore di tutta la matrice A} \\ B(r,c), & \text{altrimenti} \end{cases} \tag{7.1}$$

Ove (r, c) sono, rispettivamente, la riga e la colonna dell'elemento considerato.

Esempio

$$A = \begin{bmatrix} 9 & 2 \\ 3 & 4 \end{bmatrix}, B = \begin{bmatrix} 10 & 8 \\ 1 & 7 \end{bmatrix} \tag{7.2}$$

allora:

$$C = \begin{bmatrix} 10 & 8 \\ 3 & 7 \end{bmatrix} \tag{7.3}$$

Poiché solo $B(2, 1)$ è minore del minimo di A che è 2. Si tenga presente che, nel risolvere l'esercizio, ***non è possibile usare cicli*** `for`.

Soluzione

```
function C = funz(A, B)
    C = B;
    C(min(min(A)) > B)=A(min(min(A)) > B)
end
```

7.2 Codice ISBN

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio Matlab. Il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Introduzione

Il codice ISBN è un array di 13 cifre usata internazionalmente per la classificazione dei libri. L'ultima cifra c del codice ISBN v svolge una funzione di controllo e viene calcolata con la seguente formula:

- si moltiplica ognuna delle prime 12 cifre per un peso definito in base alla posizione della cifra stessa nella sequenza: la prima cifra si moltiplica per 1, la seconda per 3, la terza per 1, la quarta per 3 e così via
- si sommano i risultati delle 12 moltiplicazioni
- si divide la somma per 10 e si prende il resto della divisione
- si sottrae il resto della divisione da 10: la cifra che si ottiene è la cifra di controllo, ovvero la 13-esima cifra del codice ISBN.

Richiesta

Implementare in linguaggio Matlab una funzione `controllo` che riceve in ingresso un vettore numerico contenente le prime 12 cifre di un codice ISBN e ritorna la corrispondente 13-esima cifra di controllo.

Esempio

Ad esempio, `controllo([9 7 8 8 8 4 3 0 2 5 3 4])` ritorna 3 poiché:

```
9*1 + 7*3 + 8*1 + 8*3 + 8*1 + 4*3 + 3*1 + 0*3 + 2*1 + 5*3 + 3*1 + 4*3 = 117
117 mod 10 = 7
10 - 7 = 3
```

Richiesta

Implementare in linguaggio Matlab una funzione `verifica` che riceve in ingresso un vettore numerico contenente le 13 cifre di un codice ISBN e ritorna `true` se la cifra di controllo è corretta, `false` altrimenti.
Ad esempio `verifica([9 7 8 8 8 4 3 0 2 5 3 4 3])` ritorna `true` dato che, come visto sopra, la cifra di controllo corretta per l'input considerato è 3.

Soluzione

```
function c = controllo(a)
        s = sum(a(1:2:12)) + sum(3 * a(2:2:12));
        c = 10 - mod(s,10);
end
```

```
function r = verifica(a)
     r = a(13) == controllo(a(1:12));
end
```

7.3 Caldaia

In questo esercizio viene richiesta la scrittura di una funzione che, dati alcuni parametri in ingresso, calcola un valore risultante ritornato al chiamante (sia esso l'utente oppure un altro programma). Per controllare di aver eseguito l'esercizio correttamente, scrivete la funzione in un file Matlab e provate ad invocarla direttamente da linea comando passando i valori di esempio e controllando che ritorni effettivamente il valore aspettato.

Introduzione

In un cinema di Milano la temperatura nella sala è regolata in modo automatico. Un sensore monitora la temperatura rilevando un valore ogni minuto mentre la caldaia si accende solo quando la temperatura rilevata è inferiore a una certa soglia e si spegne non appena viene raggiunta tale soglia.

Richiesta

Scrivere in Matlab una funzione `calcolaCosto` che:

- riceve in ingresso un vettore `temperature`, una soglia `soglia` e un `costoAlMinuto` che indica il costo al minuto del gas consumato dalla caldaia;
- restituisce il `costoTotale` del gas consumato e un vettore `minutiAccesa` con i minuti nei quali la caldaia è rimasta accesa. Per il costo totale, si consideri che se il totale dei minuti nei quali la caldaia ha funzionato supera i 30 minuti, allora il costo del gas consumato dalla caldaia va diminuito del 20%.

Esempio

Nel caso la funzione `calcolaCosto` riceva in ingresso un vettore `temperature` con questi valori:

```
22.00
22.50
23.20
21.45
22.00
22.35
23.00
23.40
```

una soglia di temperatura pari a 23.00 e un valore di `costoAlMinuto` pari a 100, ritornerà un costo totale di 500 e `minutiAccesa = [1 2 4 5 6]`.

Richiesta

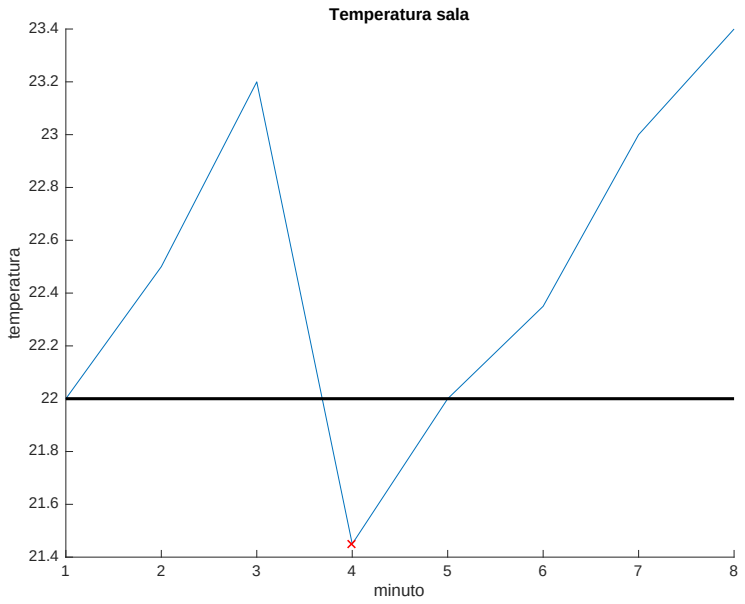
Si scriva poi uno script che:

- definisca il valore soglia di temperatura `soglia` pari a 22;
- legga dal file `temp.mat` i valori di temperatura corrispondenti al vettore `temp`;
- definisca il valore del parametro `costoAlMinuto`;
- crei una opportuna variabile `x` che indica i minuti;

- disegni il grafico (con titolo del grafico e dei due assi) della temperatura al variare del tempo, evidenziando sul grafico stesso:
  - la temperatura soglia con una retta orizzontale nera a doppio spessore;
  - i minuti nei quali la caldaia ha funzionato, marcandoli con il simbolo 'X' rosso (senza congiungerli).
- stampi a video il costo totale del gas consumato e i minuti nei quali la caldaia è rimasta accesa.

Esempio

Un esempio di grafico richiesto è il seguente:



Soluzione

```
function [costoTotale,minutiAccesa] = calcolaCosto(temp, soglia, costoAlMinuto)
    minutiAccesa = find(temp < soglia);
    if (length(minutiAccesa) > 30)
        costoTotale = length(minutiAccesa) * costoAlMinuto * 0.80;
    else
        costoTotale = length(minutiAccesa) * costoAlMinuto;
    end
end
```

```
load -ascii temp.mat;
soglia = 22;
costoAlMinuto = 10;
x = 1:size(temp, 1);
figure
hold on
```

```
ylabel('temperatura');
xlabel('minuto');
title('Temperatura sala');
plot(x, temp);
y = soglia * ones(size(x));
plot(x, y, 'k', 'LineWidth', 2);
indici = find(temp < soglia);
temp1 = temp(indici);
x1 = x(indici);
plot(x1, temp1, 'Xr');

[costoTotale, minutiAcceso] = calcolaCosto(temp, soglia, costoAlMinuto);

disp(['Costo totale del gas consumato: ', num2str(costoTotale), ' Euro.']);
disp('La caldaia ha funzionato nei seguenti minuti: ');
disp(minutiAcceso);
```

7.4 Andamento capitale

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio Matlab. Il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Introduzione

Il Sig. Rossi ha predisposto investimenti in svariati strumenti finanziari (azioni, obbligazioni, titoli di stato, etc..) e ha annotato:

- Il ***capitale iniziale*** di ogni investimento (a gennaio 2012).
- Il ***valore corrente*** di ogni investimento (a gennaio 2013).

Il Sig. Rossi a gennaio 2012 ha preventivato di ottenere una ***rendita minima*** dai suoi investimenti. La rendita minima è uguale per tutti gli investimenti ed è espressa come una *percentuale*; è quindi numero nell'intervallo tra 0 e 1 (ad esempio +3% = 0.03). Si assuma che il capitale iniziale e il valore corrente degli investimenti vengano ordinatamente salvati in due vettori di uguale lunghezza chiamati, rispettivamente, `ci` e `vc`, e che quindi l'indice in questi vettori identifichi univocamente lo strumento finanziario nel quale il Sig. Rossi ha investito.

Richiesta

- Sviluppare una funzione Matlab/Octave chiamata `controllaRendita` che restituisce, in un vettore `tb` gli indici degli investimenti che sono cresciuti, a gennaio 2013, di una percentuale maggiore o uguale alla rendita minima `rm` che il Sig. Rossi sperava di ottenere quando ha investito:

Richiesta

- Si costruisca uno script che:
  - carica dal file ascii `capitaleIniziale.txt` gli importi degli investimenti a gennaio 2012 e dal file ascii `valoreCorrente.txt` i valori correnti degli investimenti (ogni file contiene un numero per ogni riga).
  - Richieda all'utente di inserire una rendita minima percentuale `rm`.
  - Genera una tabella `tab` contenente il capitale iniziale e finale dei soli titoli buoni e la sua rendita calcolata come:

```
r(ii) = (vc(ii) / ci(ii)) - 1
```

Soluzione

```
function tb = controllaRendita(ci, vc, rm)
         tb = find(vc ./ ci >= 1 + rm)
end
```

```
ci      = load('ci.txt');
vc      = load('vc.txt');
rm      = input('Inserire il valore della rendita minima (in intervallo [0,1]): ');

tb      = controllaRendita(ci, vc, rm);

rendita = vc ./ ci - 1;
tab     = [ci' vc' rendita']
tab     = tab(tb)
```

7.5 Cinematica

Introduzione

In questo esercizio progetteremo un programma Matlab per il calcolo di parte della traiettoria di un braccio robotico che controlla una penna su un foglio. Entrambe le coordinate x e y della penna nel piano cartesiano sono funzione di un vettore $[p_1, p_2]$ di due parametri interni del robot (ad esempio, gli angoli delle due braccia).
Se consideriamo solo l'asse x, sappiamo che per una piccola variazione $[\Delta p_1, \Delta p_2]$ in un certo istante dei parametri interni, si ha una corrispondente variazione della posizione Δx della penna calcolabile come:

$$\Delta x = J_1(p_1, p_2) * \Delta p_1 + J_2(p_1, p_2) * \Delta p_2$$

Richiesta

Supponiamo che il robot muova il braccio a ogni secondo ("a scatti") e che al tempo $t = 0$ si abbia $x = 0$, $p_1 = 0$, $p_2 = 0$. Supponiamo inoltre che

$$J_1(p_1, p_2) = J_2(p_1, p_2) = p_2 + p_1 + 1\,.$$

Scrivere uno script Matlab che:

- Inizializzi le variabili x e il vettore $p = [p_1, p_2]$
- Calcoli il valore di x e p all'istante 1 considerando una variazione $[\Delta p_1, \Delta p_2] = [2, 3]$
- Calcoli il valore di x e p all'istante 2 considerando una variazione $[\Delta p_1, \Delta p_2] = [1, 2]$

Richiesta

Vogliamo ora generalizzare con un algoritmo il calcolo della posizione in un generico istante. Si assuma di avere una matrice $n \times 2$ il cui nome è DP e in cui ogni riga r rappresenta la variazione $[\Delta p_1, \Delta p_2]$ dei parametri del robot all'istante $t = r$. Supponendo che i valori iniziali delle variabili siano $x = 0, p_1 = 0, p_2 = 0$, si scriva una funzione con la seguente intestazione:

```
function [X] = calcolaPos(DP)
```

che restituisce una matrice X in cui alla riga r sia contenuta la posizione x del braccio all'istante $t = r$.

Soluzione

```
x = 0
p = [0,0]

J1 = p(1)+p(2)+1
J2 = p(1)+p(2)+1
x = x + J1*2 + J2*3
p = p + [2,3]

J1 = p(1)+p(2)+1
J2 = p(1)+p(2)+1
x = x + J1*1 + J2*2
p = p + [1,2]
```

```
function [ X ] = calcolaPos(DP)
  p = [0,0];
  x = 0;
  s = size(DP)
  for i = 1:s(1)
    J1 = p(1)+p(2)+1;
    J2 = p(1)+p(2)+1;
    dp = DP(i,:)';
    dx = J1 * dp(1) + J2 * dp(2);
    p = p + dp;
    x = x + dx;
    X(i,:) = x;
  end
end
```

Capitolo 8

Esercizi Matlab - Funzioni ricorsive

8.1 Mele al mercato

Introduzione

Una signora al mercato compra un sacchetto di mele, che purtroppo le cade durante il tragitto verso casa. Il commerciante si offre di darle un altro sacchetto contenente lo stesso numero di mele del precedente e le chiede quindi quale fosse questo numero. La signora, abilissima negli indovinelli matematici, risponde così:

- Organizzandole in file da 5 mele, ne rimangono fuori 2
- Organizzandole in file da 7 mele, ne rimangono fuori 3

Intuizione

Quanto indicato dalla signora è rappresentabile dalle seguenti equazioni, dove m è il numero di mele che vogliamo trovare:

$$\begin{cases} \text{mod}(m, 5) &= 2 \\ \text{mod}(m, 7) &= 3 \end{cases} \tag{8.1}$$

Formalizzazione

Le equazioni di sopra sono un esempio di *equazione alle congruenze (in m)*:

$$\begin{cases} \text{mod}(m, a) &= w_1 \\ \text{mod}(m, b) &= w_2 \end{cases} \tag{8.2}$$

che ha soluzione $m = b * w_1 * y + a * w_2 * x$, dove x e y sono calcolati con l'***algoritmo esteso di Euclide***:

$$(x, y) = \text{calcolaCoeff}(a, b) = \begin{cases} (1, 0) & \text{quando } (b = 0) \\ (r_x, r_y) & \text{negli altri casi} \end{cases} \tag{8.3}$$

$$\begin{aligned} r_x &= t_y \\ r_y &= t_x - t_y * (a \text{ div } b) \\ (t_x, t_y) &= \text{calcolaCoeff}(b, a \bmod b) \end{aligned} \tag{8.4}$$

Domanda 1

Implementare la funzione `calcolaCoeff` in Matlab/Octave (si usi `fix(a/b)` per la divisione intera):

Domanda 2

Si chiede di scrivere una funzione Matlab `numero_di_mele` che riceve i valori a, b, w_1 e w_2 e calcoli il valore risultante delle mele m utilizzando la funzione di cui sopra.

Domanda 3

Come invochereste la funzione `numero_di_mele` per risolvere il problema iniziale della signora del mercato? Quante mele verrebbero calcolate?

Soluzione

```
function [x, y] = calcolaCoeff(a, b)
    if b == 0
        x = 1;
        y = 0;
    else
        [x1, y1] = calcolaCoeff(b, mod(a,b))
        x = y1;
        y = x1 - y1 * (fix(a/b));
    end
end
```

```
function [m] = numero_di_mele(a,b,w_1,w_2)
    [ x, y ] = calcolaCoeff(a, b)
    c1 = b*w_1*y
    c2 = a*w_2*x
    m = (c1 + c2)
end
```

```
> numero_di_mele(5, 7, 2, 3)
ans = 17
```

8.2 Quadrati concentrici

Introduzione

Si considerino il seguente esempio di matrice costruita da ”quadrati concentrici”:

```
matr1 =
     2     2     2     2     2     2
     2     3     3     3     3     2
     2     3     4     4     3     2
     2     3     4     4     3     2
     2     3     3     3     3     2
     2     2     2     2     2     2
```

Come si vede dall' esempio, si tratta di matrice ***quadrata*** in cui i valori che si trovano sulla n-esima riga, n-esima colonna, e sulle righe e colonne simmetriche a queste sono uguali tra loro.

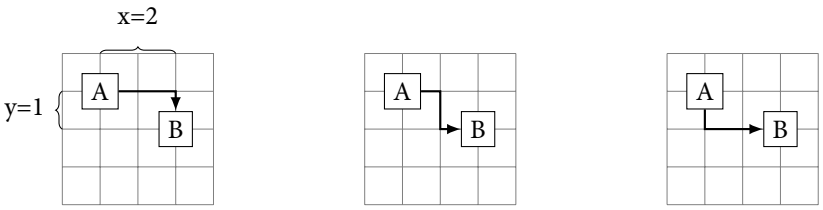
Richiesta

Si sviluppi in Matlab una funzione ricorsiva `quadratiConcentrici` che, data una generica matrice, restituisca 1 se la matrice è costituita da quadrati concentrici, 0 altrimenti.
Per sviluppare questa funzione si assuma di avere a disposizione la funzione `valoriDiCorniceUguali` che, data una matrice quadrata, restituisce 1 se tutti i valori disposti sulla sua cornice esterna (costituita dalla prima e dall'ultima riga e dalla prima e dall'ultima colonna) sono uguali tra loro, 0 altrimenti. Per esempio: `valoriDiCorniceUguali(matr1)` restituisce 1.
NB: non si chiede di sviluppare `valoriDiCorniceUguali`. Ci si focalizzi solo sulla funzione ricorsiva.

Soluzione

```
function [ris] = quadratiConcentrici(m)
    [r, c] = size(m);
    if r ~= c
        ris = false;
    else
        if r == 1 || r == 0
            ris = true;
        elseif valoriDiCorniceUguali(m)
            ris = quadratiConcentrici(m(2:end-1, 2:end-1));
        else
              ris = false;
        end
    end
end
```

Figura 8.1: I 3 percorsi a distanza minima che collegano due incroci A e B caratterizzati da una distanza lungo l'asse X di 2 e lungo l'asse Y di 1.



8.3 Grigliopoli

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio Matlab. Il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Introduzione

Le strade della città di ***Grigliopoli*** sono organizzate come una griglia (alcune strade attraversano la città da est a ovest e altre da nord a sud). Dati due incroci che distano `X` isolati lungo l'asse est-ovest della città e `Y` isolati lungo l'asse nord-sud, siete stati incaricati di calcolare il numero di percorsi a distanza minima che collegano i due incroci.

Richiesta

Il vostro obiettivo è quello di implementare ***una funzione ricorsiva*** `calcola` in Matlab/Octave che ricevuti `X` e `Y` in ingresso restituisce il numero totale di percorsi corrispondenti. Ovvero, nell'esempio di sopra `calcola(2,1)` deve ritornare 3.

Soluzione

Nello scrivere la soluzione, consideriamo innanzitutto il caso base, ovvero quando `X` = 0 o `Y` = 0, c'è soltanto un cammino a distanza minima (1) che collega i due incroci. Altrimenti, esiste più di un cammino minimo dal momento che è possibile sia avvicinarsi alla destinazione lungo l'asse est-ovest (riducendo quindi la distanza `X`) oppure avvicinarsi lungo l'asse nord-sud (riducendo la distanza `Y`).

```
function [ p ] = calcola(x,y)
    if (x == 0 || y == 0)
      p = 1;
    else
      p = calcola(x-1,y) + calcola(x,y-1);
    end
end
```

8.4 Rolling text

Richiesta

Si implementi in linguaggio Matlab un funzione ***ricorsiva*** `roll(testo, n)` che ritorni una matrice di caratteri dove:

- la prima riga consiste nei primi `n` caratteri di `testo`.
- la seconda riga consiste negli `n` caratteri di `testo` a cui è stato tolto il primo carattere.
- la terza riga consiste negli `n` caratteri di `testo` a cui sono stati tolti i primi due caratteri e così via.
- Nel caso la stringa `testo` non abbia `n` caratteri inserire il carattere `-`.
- La matrice è considerata completa quando sono stati tolti tutti i caratteri dalla stringa originaria

Esempio

Per esempio, la matrice risultante di `roll('buon viaggio', 10)` deve essere:

```
buon viagg
uon viaggi
on viaggio
n viaggio-
 viaggio--
viaggio---
iaggio----
aggio-----
ggio------
gio-------
io--------
o---------
----------
```

Soluzione

```
function m = roll(txt, n)
  riga(1:n) = '-';
  if(length(txt) ~= 0)
    u = min(length(txt), n)
    riga(1:u) = txt(1:u)
    m = [ riga; roll(txt(2:end), n) ];
  else
    m = riga;
  end
end
```

8.5 Matrice con frazioni

In questo esercizio viene richiesta la scrittura di una funzione che, dati alcuni parametri in ingresso, calcola un valore risultante ritornato al chiamante (sia esso l'utente oppure un altro programma). Per controllare di aver eseguito l'esercizio correttamente, scrivete la funzione in un file Matlab e provate ad invocarla direttamente da linea comando passando i valori di esempio e controllando che ritorni effettivamente il valore aspettato.

Introduzione

La funzione che viene richiesta in questo esercizio lavora con valori in ingresso rappresentati da una matrice di 2 righe ed un numero arbitrario di colonne, ad esempio:

$$M = \begin{bmatrix} 1 & 3 & 1 \\ 2 & 4 & 3 \end{bmatrix} \tag{8.5}$$

Ogni colonna della matrice rappresenta una frazione (ad esempio, per `M`):

$$\frac{1}{2}, \frac{3}{4}, \frac{1}{3} \tag{8.6}$$

Richiesta

Si scriva una funzione ***ricorsiva*** `frac` che, ricevuta una qualsiasi matrice `M` in ingresso, ritorni il numeratore `n` ed il denominatore `d` della frazione risultante dalla somma delle frazioni contenute in `M`.

Esempio

Ad esempio:

```
[n d] = frac([1 3 1; 2 4 3])
```

ritorna

```
n = 19
d = 12
```

poiché:

$$\frac{1}{2} + \frac{3}{4} + \frac{1}{3} = \frac{12 \times 1 + 6 \times 3 + 2 \times 4}{2 \times 4 \times 3} = \frac{19}{12} \tag{8.7}$$

La frazione deve essere minimizzata, ovvero non vi devono essere divisori comuni tra `n` e `d`. Nel progettare `frac`, si può utilizzare la funzione `gcd(a,b)` che ritorna il massimo comune divisore tra `a` e `b`.

1. Soluzione

```
function [n, d] = frac(M)
    if size(M,2) == 1
        n = M(1,1);
        d = M(2,1);
    else
        [c d] = frac(M(:,2:end));
        a = M(1,1);
        b = M(2,1);

        k = a*d + c*b;
        l = b * d;

        n = k/gcd(k,l);
        d = l/gcd(k,l);
    end
end
```

8.6 Funzione sconosciuta

Introduzione

Data la seguente funzione ricorsiva Matlab:

```
function [c, d] = fun(x, y)
        if (x < y)
            d = 0;
            c = x;
        else
            [a, b] = fun(x - y, y);
            d = b + 1;
            c = a;
        end
end
```

Richieste

1. Domanda 1

   Calcolare i seguenti valori:

   - `fun(3,6)`
   - `fun(6,6)`
   - `fun(4,2)`
   - `fun(5,2)`

2. Domanda 2

   Spiegare con ***massimo 10 parole*** cosa fa la funzione `fun`.

3. Domanda 3

   Come potreste scrivere la stessa funzione in maniera non ricorsiva e non utilizzando nessun ciclo?

Soluzione

La funzione calcola il resto `c` e il quoziente `d` dei parametri in entrata, ovvero per i valori di esempio:

- `fun(3,6) = [3, 0]`
- `fun(6,6) = [0, 1]`
- `fun(4,2) = [0, 2]`
- `fun(5,2) = [1, 2]`

Una equivalente espressione non ricorsiva è la seguente:

```
function [c, d] = fun2(x, y)
  c = mod(x,y);
  d = floor(x/y);
end
```

8.7 Estrazione cifra

In questo esercizio viene richiesta la scrittura di alcuni frammenti di programma in linguaggio Matlab. Il testo non dichiara esplicitamente la firma dell'eventuale algoritmo da scrivere, il tipo dei dati in ingresso e di quelli da elaborare; tali informazioni sono infatti da dedurre dal testo stesso.

Introduzione

L'esercizio è composto da due punti; nelle soluzioni non è consentito l'uso della funzione `num2str` di Matlab/Octave:

1. Scrivere la funzione ricorsiva `cifra()` che riceve come parametri due numeri interi `num` e `k` (si supponga che entrambi i numeri siano sempre strettamente positivi). La funzione `cifra` restituisce la `k`-esima cifra del numero `num` a partire da destra. Esempi:
   - `cifra(1456, 1)` deve restituire 6
   - `cifra(5136, 4)` deve restituire 5
   - `cifra(512, 2)` deve restituire 1
2. Riscrivere la funzione ricorsiva del punto precedente in modo tale che nel caso in cui `k` sia maggiore del numero effettivo di cifre che compongono `num` la funzione restituisca -1.

Soluzione

```
function ris = cifra(num, k)
    if k == 1
        ris = mod(num, 10);
    else
        ris = cifra(floor(num/10), k-1);
    end
end
```

```
function ris = cifraConControllo(num, k)
    if (k > 1 && num < 10)
        ris = -1
    else
        if k == 1
            ris = mod(num, 10);
        else
            ris = cifraConControllo(floor(num/10), k-1);
        end
    end
end
```

8.8 Numeri di Catalan

Introduzione

I numeri di Catalan sono una successione di interi positivi C_n. Il nome di questi numeri è stato scelto in onore del matematico belga Eugène Charles Catalan (1814-1884) che li aveva studiati intorno al 1838 per stabilire in quanti modi si può ricondurre il prodotto di r fattori ad una successione di prodotti a coppie.
I numeri di Catalan si possono ottenere ricorsivamente, secondo la seguente relazione:

$$C_n = \begin{cases} 1 & n = 0 \\ \sum_{i=0}^{n-1} C_i C_{n-1-i} & \text{per } n \geq 1 \end{cases} \tag{8.8}$$

Richiesta

1. Domanda 1 Quanto vale ciascuno dei seguenti numeri di Catalan?
   - C_1
   - C_2
   - C_3
2. Domanda 2 Si descriva una funzione Matlab ricorsiva `catalan` che, ricevuto un numero `n` come parametro formale, ritorna il valore di C_n.

Soluzioni

- $C_1 =$ `1`
- $C_2 =$ `2`
- $C_3 =$ `5`

```
function c = catalan(n)
  if n == 0
    c = 1;
  else
    for i = 0:n-1
      v(i+1) = catalan(i) * catalan(n - 1 - i);
    end
    c = sum(v);
end
```

8.9 Comprensione funzione

In questo esercizio, cosidetto *di tracing*, viene richiesto di simulare ”mentalmente” il programma seguente e predire che cosa stamperà a terminale durante la sua esecuzione. In pratica, fate finta di essere voi il calcolatore ed eseguite le istruzioni partendo dalla prima fino a che non raggiungete l'ultima. Utilizzate un foglio di carta per annotare il valore corrente delle variabili e abbiate cura di tenerlo aggiornato ogni volta che eseguite ”mentalmente” una istruzione.

Funzione da studiare

```
function r = f(a)
    if a == 0
      r = [];
    else
      r = [f(floor(a/2)) mod(a,2)];
    end
end
```

Richiesta

- Qual è il valore ritornato dalla chiamata `f(5)`?
- Qual è il valore ritornato dalla chiamata `f(10)`?
- Ipotizzando che la funzione `f(a)` venga chiamata con un argomento a intero e positivo, descrivere sinteticamente cosa calcola la funzione

Soluzione

- *Il valore ritornato è* `[1 0 1]`
- *Il valore ritornato è* `[1 0 1 0]`
- *La funzione* `f(a)` *calcola le cifre della codifica binaria del numero* `a`.

8.10 Comprensione funzione

In questo esercizio, cosidetto *di tracing*, viene richiesto di simulare ”mentalmente” il programma seguente e predire che cosa stamperà a terminale durante la sua esecuzione. In pratica, fate finta di essere voi il calcolatore ed eseguite le istruzioni partendo dalla prima fino a che non raggiungete l'ultima. Utilizzate un foglio di carta per annotare il valore corrente delle variabili e abbiate cura di tenerlo aggiornato ogni volta che eseguite ”mentalmente” una istruzione.

Funzione da studiare

```
function [qua] = paperone(a, b)
    if b==1 || b==0
        qua = false;
    else
        if (mod(a, b) == 0) && a~=b
            qua = true;
        else
            qua = paperone(a, b-1);
        end
    end
end

function [x] = paperino(aaargh)
    x = true;
    if paperone(aaargh,aaargh) ~= 1;
        x=false;
        return
    end
end
```

Richiesta

1. A cosa corrisponde la funzione ricorsiva `paperone`?
2. Si dica a cosa corrisponde il caso in cui `paperino(k)` (per `k` intero) ritorna `true`:
3. Si indichi l'output del seguente codice:

```
for d = 1:3:20
    printf('%d - %d\n', d, paperino(d))
end
```

Risposta

1. *La funzione* `paperone(a, b)` *controlla se*:
    - c'è un divisore di `a`
    - ed è minore di `b`
    - ed è diverso da `a` e `1`

2. *Indica se il numero* `k` *è un numero non primo.*
3. Output:

```
1 - 0
4 - 1
7 - 0
10 - 1
13 - 0
16 - 1
19 - 0
```

8.11 Traiettoria aereo

Introduzione

Si assuma di avere una matrice di 2 righe ed un numero arbitrario di colonne che descrive la traiettoria di un aereo. Un esempio di tale matrice è la seguente:

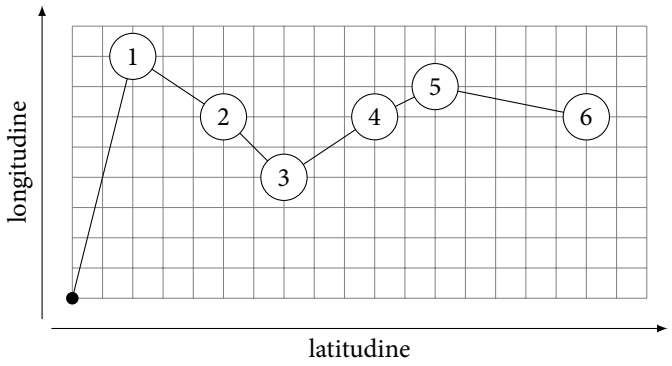
$$M = \begin{bmatrix} 2 & 3 & 2 & 3 & 2 & 5 & \dots \\ 8 & -2 & -2 & 2 & 1 & -1 & \dots \end{bmatrix} \tag{8.9}$$

Ciascuna colonna i contiene due valori $\begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix}$ dove:

- Δx_i rappresenta lo spostamento in latitudine dell'aereo fra il minuto i ed il minuto $i-1$.
- Δy_i rappresenta analogamente lo spostamento in longitudine.

Dettagli

La matrice definisce quindi una traiettoria nello spazio; si noti quindi che ciascuna colonna descrive lo spostamento rispetto al punto precedente e che la traiettoria così descritta è indipendente dal punto iniziale.



Richiesta

Si chiede di scrivere una ***funzione ricorsiva*** `distanza(M)` che ricevuta una matrice M in ingresso come parametro formale, ritorna la distanza totale percorsa dall'aereo lungo tutta la traiettoria descritta da M, partendo dall'istante 0 — La distanza è da intendersi come ***la somma delle lunghezze dei segmenti*** che compongono la traiettoria.

Soluzione

```
function d = distanza(M)
   dx = M(1,1);
   dy = M(2,1);
   d = sqrt(dx^2 + dy^2);
   if size(M,2) > 1
      d = d + distanza(M(:,2:end));
   end
end
```

8.12 Coefficiente binomiale

Richiesta

Data la definizione di coefficiente binomiale:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \quad \text{per tutti gli interi } n, k : 1 \leq k \leq n-1,$$

dove:

$$\binom{n}{0} = \binom{0}{n} = \binom{n}{n} = 1 \quad \text{per tutti gli interi } n \geq 0,$$

- Si definisca, in Matlab, una funzione ricorsiva `coeffbinom` che lo implementa senza utilizzare cicli.
- Quanti *workspace* locali vengono allocati (al massimo) per il calcolo di $\binom{3}{2}$.

Soluzione

```
function b = binom(n,k)
  if(n == 0 || k == 0 || n == k)
    b = 1
  else
    b = binom(n-1, k) + binom(n-1, k-1)
  end
end
```

8.13 Massimo comun divisore

Richiesta

Si implementi, in Matlab, una funzione ricorsiva:

```
function m = mcd(v)
```

che, ricevuto un vettore v di n valori, ritorni il ***massimo comun divisore*** di tali numeri senza utilizzare cicli. A tal fine, si noti che vale la seguente uguaglianza:

$$mcd(v_0, v_1, \ldots, v_{n-1}) = gcd(v_0, mcd(v_1, \ldots, v_{n-1}))$$

ove gcd è la funzione ***greatest common divisor*** di Matlab.

Soluzione

```
function m = mcd(v)
  if(length(v) == 1)
    m = v;
  else
    m = gcd(v(1), mcd(v(2:end)));
  end
end

mcd([9 18 6 27 30 42])
```

Capitolo 9

Esercizi su tabelle della verità

9.1 Esercizio 1

Richiesta n. 1

Si consideri la seguente espressione in algebra Booleana:

$$\text{NOT}(B \text{ AND } \text{NOT } A) \text{ OR } (\text{NOT } B \text{ OR } C)$$

Si compili la tabella della verità utilizzando 0 per rappresentare il valore logico FALSO e 1 per il valore VERO

Richiesta n. 2

Si consideri ora la condizione, scritta in linguaggio C, in cui `x` e `y` siano due variabili `int`:

```
1  ! ( (y > 7) && ! (x > 3) ) || ( ! (y > 7) || (x < 0) )
```

ottenuta dalla prima formula sostituendo la variabile A con `(x > 3)`, la variabile B con `(y > 7)`, la variabile C con `(x < 0)`. Si risponda quindi alle seguenti domande:

- L'espressione è vera o falsa quando `x = 1` e `y = 8`? (giustificare la risposta)
- Nella condizione in linguaggio C della domanda precedente, se `x = 2`, per quali valori di `y` l'espressione è vera? (giustificare la risposta)

Soluzione

A	B	C	NOT A	B AND NOT A	NOT B	NOT B OR C	NOT$(B$ AND NOT $A)$ OR $(\text{NOT } B \text{ OR } C)$
0	0	0	1	0	1	1	1
0	0	1	1	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	1	1	0	1	1
1	0	0	0	0	1	1	1
1	0	1	0	0	1	1	1
1	1	0	0	0	0	0	1
1	1	1	0	0	0	1	1

- Per `x = 1` e `y = 8`, abbiamo A = falso, B = vero, C = falso per cui, dalla tabella della verità, l'espressione risulta falsa.
- Se `x = 2` allora A = falso e C = falso. In questo caso, l'espressione è vera (si veda tabella della verità) solo se B è falsa quindi solo per y <= 7.

9.2 Esercizio 2

Introduzione

Si consideri il seguente frammento di script MATLAB:

```
if ( any([a,b] > [0,3]) || b < 0) && all(c~=[-1,2,3])
    disp ('VERO');
else
    disp ('FALSO');
end
```

Richiesta

Scegliere la/le opzioni corrette. Nota bene, più di una delle seguenti opzioni può essere corretta. Dovete segnarle tutte. Si ricorda che il simbolo $\wedge$ indica il connettivo logico AND mentre $\vee$ indica il connettivo OR.

- Lo script stampa VERO quando $a > 0 \ \vee \ b < 0 \ \vee \ b > 3 \ \wedge \ c \notin \{-1, 2, 3\}$
- Lo script stampa VERO quando $(a > 0 \ \vee \ b < 0 \ \vee \ b > 3) \ \wedge \ c \notin \{-1, 2, 3\}$
- Lo script stampa VERO quando $(a > 0 \ \wedge \ b < 0 \ \wedge \ b > 3) \ \wedge \ c \in \{-1, 2, 3\}$
- Lo script stampa sempre FALSO
- Nessuna delle opzioni di cui sopra

9.3 Esercizio 3

Richiesta

Per ognuna delle espressioni logiche riportate nelle righe della tabella, assumendo le seguenti dichiarazioni:

```
int a = 8, b = 11;
char c='d';
```

Indicare se l'espressione è vera o falsa, se l'espressione è vera per qualsiasi valore delle variabili o se l'espressione è falsa per qualsiasi valore delle variabili

1. `(-a==a) && (a < 11)`
2. `(c>'a' || c<'z' ) && (a<7 && b>8)`
3. `!(b<10 && a>7) || (c!='h' && c>'a')`

Soluzione

Espressione	Vera o falsa	Sempre vera	Sempre falsa
n. 1	F	NO	NO
n. 2	F	NO	NO
n. 3	V	NO	NO

1. L'espressione è vera nel caso `a=0` e falsa in tutti gli altri casi
2. L'espressione è falsa per i valori dati per la presenza del termine `a<7`; è vera, per valori di `a<7` e `b>8` perché il termine `(c>'a' || c<'z')` è sempre vero. Di conseguenza, l'espressione non è sempre falsa.
3. L'espressione è vera perché, indipendentemente dai valori di `a` e `b`, si ha che `c!='h'` e `c>'a'` sono entrambe vere; è falsa, per esempio, per `a=8, b=9` (che rendono falso il termine `!(b<10 && a>7)`) e `c='h'` (che rende falso il termine `(c!='h' && c>'a')`), quindi non è sempre vera.

Capitolo 10

Codifica binaria dei numeri interi

10.1 Codifica numeri

Calcolare il numero minimo di bit da usare per codificare i numeri nella colonna sinistra di questa tabella nella ***codifica binaria naturale***. Calcolare inoltre tale codifica.

Numero	***N. di bit***	***Codifica***
1	1	$[\![\texttt{1}]\!]_2$
2	2	$[\![\texttt{10}]\!]_2$
7	3	$[\![\texttt{111}]\!]_2$
8	4	$[\![\texttt{1000}]\!]_2$
15	4	$[\![\texttt{1111}]\!]_2$
22	5	$[\![\texttt{10110}]\!]_2$

10.2 Codifica numeri

Calcolare il numero minimo di bit da usare per codificare i numeri nella colonna sinistra di questa tabella nella ***codifica in complemento a due***. Calcolare inoltre tale codifica.

Numero	***N. di bit***	***Codifica***
-1	1	$[\![\texttt{1}]\!]_{C2}$
-2	2	$[\![\texttt{10}]\!]_{C2}$
-4	3	$[\![\texttt{100}]\!]_{C2}$
-7	4	$[\![\texttt{1001}]\!]_{C2}$
-8	4	$[\![\texttt{1000}]\!]_{C2}$
-9	5	$[\![\texttt{10110}]\!]_{C2}$

10.3 Numero minimo di bit

Si definisca il minimo numero di bit necessari per rappresentare in complemento a 2 tutti i seguenti valori interi:

Numero	*N. di bit*	*Codifica*
149	9	$[\![010010101]\!]_{C2}$
108	8	$[\![01101100]\!]_{C2}$
12	5	$[\![01100]\!]_{C2}$
42	7	$[\![0101010]\!]_{C2}$
92	8	$[\![01011100]\!]_{C2}$

Calcolare -149–108 in codifica binaria complemento a 2:

- ***Passo 1*** - Converto -149: Bastano 9 bit ovvero $[\![-149]\!]_{10}$ = $[\![101101011]\!]_{C2}$ poiché è ottenuto da $[\![149]\!]_{10}$=$[\![010010101]\!]_{C2}$ invertendo tutti i bit e aggiungendo 1.
- ***Passo 2*** - Converto -108: Servono 9 bit ovvero $[\![-108]\!]_{10}$ = $[\![110010100]\!]_{C2}$ poiché è ottenuto da $[\![108]\!]_{10}$=$[\![01101100]\!]_{C2}$ invertendo tutti i bit e aggiungendo 1; infine estendo il segno negativo per avere la codifica su 9 bit e poter quindi fare la somma.
- ***Passo 3*** - faccio la somma -149 + (-108);

```
   101101011 (codifica c2 di −149)
   110010100 (codifica c2 di −108)
   _________
(1)011111111
```

Ho un riporto perduto e overflow, poiché il risultato è discorde rispetto agli addendi. Una controprova è il fatto che su 9 bit posso rappresentare al minimo -256, mentre il risultato è -257.

Capitolo 11

Informazioni utili

In questo capitolo sono riportate alcune tabelle e guide di riferimento utili alla soluzione degli esercizi.

REGULAR ASCII CHART (character codes 0 – 127)

000d	00h		(nul)	016d	10h	►	(dle)	032d	20h		048d	30h	0	064d	40h	@	080d	50h	P	096d	60h	‘	112d	70h	p
001d	01h	☺	(soh)	017d	11h	◄	(dc1)	033d	21h	!	049d	31h	1	065d	41h	A	081d	51h	Q	097d	61h	a	113d	71h	q
002d	02h	☻	(stx)	018d	12h	↕	(dc2)	034d	22h	"	050d	32h	2	066d	42h	B	082d	52h	R	098d	62h	b	114d	72h	r
003d	03h	♥	(etx)	019d	13h	‼	(dc3)	035d	23h	#	051d	33h	3	067d	43h	C	083d	53h	S	099d	63h	c	115d	73h	s
004d	04h	♦	(eot)	020d	14h	¶	(dc4)	036d	24h	$	052d	34h	4	068d	44h	D	084d	54h	T	100d	64h	d	116d	74h	t
005d	05h	♣	(enq)	021d	15h	§	(nak)	037d	25h	%	053d	35h	5	069d	45h	E	085d	55h	U	101d	65h	e	117d	75h	u
006d	06h	♠	(ack)	022d	16h	▬	(syn)	038d	26h	&	054d	36h	6	070d	46h	F	086d	56h	V	102d	66h	f	118d	76h	v
007d	07h	•	(bel)	023d	17h	↨	(etb)	039d	27h	'	055d	37h	7	071d	47h	G	087d	57h	W	103d	67h	g	119d	77h	w
008d	08h	◘	(bs)	024d	18h	↑	(can)	040d	28h	(	056d	38h	8	072d	48h	H	088d	58h	X	104d	68h	h	120d	78h	x
009d	09h		(tab)	025d	19h	↓	(em)	041d	29h	)	057d	39h	9	073d	49h	I	089d	59h	Y	105d	69h	i	121d	79h	y
010d	0Ah	◙	(lf)	026d	1Ah		(eof)	042d	2Ah	*	058d	3Ah	:	074d	4Ah	J	090d	5Ah	Z	106d	6Ah	j	122d	7Ah	z
011d	0Bh	♂	(vt)	027d	1Bh	←	(esc)	043d	2Bh	+	059d	3Bh	;	075d	4Bh	K	091d	5Bh	[	107d	6Bh	k	123d	7Bh	{
012d	0Ch		(np)	028d	1Ch	└	(fs)	044d	2Ch	’	060d	3Ch	<	076d	4Ch	L	092d	5Ch	\	108d	6Ch	l	124d	7Ch	|
013d	0Dh	♪	(cr)	029d	1Dh	↔	(gs)	045d	2Dh	-	061d	3Dh	=	077d	4Dh	M	093d	5Dh	]	109d	6Dh	m	125d	7Dh	}
014d	0Eh	♫	(so)	030d	1Eh	▲	(rs)	046d	2Eh	.	062d	3Eh	>	078d	4Eh	N	094d	5Eh	^	110d	6Eh	n	126d	7Eh	~
015d	0Fh	☼	(si)	031d	1Fh	▼	(us)	047d	2Fh	/	063d	3Fh	?	079d	4Fh	O	095d	5Fh	_	111d	6Fh	o	127d	7Fh	⌂

EXTENDED ASCII CHART (character codes 128 – 255) LATIN1/CP1252

128d	80h		€	144d	90h			160d	A0h		176d	B0h	°	192d	C0h	À	208d	D0h	Ð	224d	E0h	à	240d	F0h	ð
129d	81h			145d	91h	‘		161d	A1h	¡	177d	B1h	±	193d	C1h	Á	209d	D1h	Ñ	225d	E1h	á	241d	F1h	ñ
130d	82h		‚	146d	92h	’		162d	A2h	¢	178d	B2h	²	194d	C2h	Â	210d	D2h	Ò	226d	E2h	â	242d	F2h	ò
131d	83h		ƒ	147d	93h	“		163d	A3h	£	179d	B3h	³	195d	C3h	Ã	211d	D3h	Ó	227d	E3h	ã	243d	F3h	ó
132d	84h		„	148d	94h	”		164d	A4h	¤	180d	B4h	´	196d	C4h	Ä	212d	D4h	Ô	228d	E4h	ä	244d	F4h	ô
133d	85h		…	149d	95h	•		165d	A5h	¥	181d	B5h	µ	197d	C5h	Å	213d	D5h	Õ	229d	E5h	å	245d	F5h	õ
134d	86h		†	150d	96h	–		166d	A6h	¦	182d	B6h	¶	198d	C6h	Æ	214d	D6h	Ö	230d	E6h	æ	246d	F6h	ö
135d	87h		‡	151d	97h	—		167d	A7h	§	183d	B7h	·	199d	C7h	Ç	215d	D7h	×	231d	E7h	ç	247d	F7h	÷
136d	88h		ˆ	152d	98h	˜		168d	A8h	¨	184d	B8h	¸	200d	C8h	È	216d	D8h	Ø	232d	E8h	è	248d	F8h	ø
137d	89h		‰	153d	99h	™		169d	A9h	©	185d	B9h	¹	201d	C9h	É	217d	D9h	Ù	233d	E9h	é	249d	F9h	ù
138d	8Ah		Š	154d	9Ah	š		170d	AAh	ª	186d	BAh	º	202d	CAh	Ê	218d	DAh	Ú	234d	EAh	ê	250d	FAh	ú
139d	8Bh		‹	155d	9Bh	›		171d	ABh	«	187d	BBh	»	203d	CBh	Ë	219d	DBh	Û	235d	EBh	ë	251d	FBh	û
140d	8Ch		Œ	156d	9Ch	œ		172d	ACh	¬	188d	BCh	¼	204d	CCh	Ì	220d	DCh	Ü	236d	ECh	ì	252d	FCh	ü
141d	8Dh			157d	9Dh			173d	ADh		189d	BDh	½	205d	CDh	Í	221d	DDh	Ý	237d	EDh	í	253d	FDh	ý
142d	8Eh		Ž	158d	9Eh	ž		174d	AEh	®	190d	BEh	¾	206d	CEh	Î	222d	DEh	Þ	238d	EEh	î	254d	FEh	þ
143d	8Fh			159d	9Fh	Ÿ		175d	AFh	¯	191d	BFh	¿	207d	CFh	Ï	223d	DFh	ß	239d	EFh	ï	255d	FFh	ÿ

Hexadecimal to Binary

0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Groups of ASCII-Code in Binary

Bit 6	Bit 5	Group
0	0	Control Characters
0	1	Digits and Punctuation
1	0	Upper Case and Special
1	1	Lower Case and Special

C Reference Card (ANSI)

Program Structure/Functions

type fnc(*type*$_1$, ...);	function prototype
type name;	variable declaration
`int main(void) {`	main routine
declarations	local variable declarations
statements	
`}`	
type fnc(*arg*$_1$, ...) {	function definition
declarations	local variable declarations
statements	
`return` *value*;	
`}`	
`/* */`	comments
`int main(int argc, char *argv[])`	main with args
`exit(`*arg*`);`	terminate execution

C Preprocessor

include library file	`#include <`*filename*`>`
include user file	`#include "`*filename*`"`
replacement text	`#define` *name text*
replacement macro	`#define` *name*`(`*var*`)` *text*
Example. `#define max(A,B) ((A)>(B) ? (A) : (B))`	
undefine	`#undef` *name*
quoted string in replace	`#`
Example. `#define msg(A) printf("%s = %d", #A, (A))`	
concatenate args and rescan	`##`
conditional execution	`#if, #else, #elif, #endif`
is *name* defined, not defined?	`#ifdef, #ifndef`
name defined?	`defined(`*name*`)`
line continuation char	`\`

Data Types/Declarations

character (1 byte)	`char`
integer	`int`
real number (single, double precision)	`float, double`
short (16 bit integer)	`short`
long (32 bit integer)	`long`
double long (64 bit integer)	`long long`
positive or negative	`signed`
non-negative modulo 2^m	`unsigned`
pointer to `int, float,...`	`int*, float*,...`
enumeration constant	`enum` *tag* {*name*$_1$=*value*$_1$,...};
constant (read-only) value	*type* `const` *name*;
declare external variable	`extern`
internal to source file	`static`
local persistent between calls	`static`
no value	`void`
structure	`struct` *tag* {...};
create new name for data type	`typedef` *type name*;
size of an object (type is `size_t`)	`sizeof` *object*
size of a data type (type is `size_t`)	`sizeof(`*type*`)`

Initialization

initialize variable	*type name*=*value*;
initialize array	*type name*`[]`={*value*$_1$,...};
initialize char string	`char` *name*`[]="`*string*`";`

Constants

suffix: long, unsigned, float	`65536L, -1U, 3.0F`
exponential form	`4.2e1`
prefix: octal, hexadecimal	`0, 0x` or `0X`
Example. `031` is 25, `0x31` is 49 decimal	
character constant (char, octal, hex)	`'a', '\`*ooo*`', '\x`*hh*`'`
newline, cr, tab, backspace	`\n, \r, \t, \b`
special characters	`\\, \?, \', \"`
string constant (ends with `'\0'`)	`"abc...de"`

Pointers, Arrays & Structures

declare pointer to *type*	*type* `*`*name*;
declare function returning pointer to *type*	*type* `*f();`
declare pointer to function returning *type*	*type* `(*pf)();`
generic pointer type	`void *`
null pointer constant	`NULL`
object pointed to by *pointer*	`*`*pointer*
address of object *name*	`&`*name*
array	*name*`[`*dim*`]`
multi-dim array	*name*`[`*dim*$_1$`][`*dim*$_2$`]`...

Structures

`struct` *tag* {	structure template
declarations	declaration of members
};	

create structure	`struct` *tag name*
member of structure from template	*name*`.`*member*
member of pointed-to structure	*pointer* `->` *member*
Example. `(*p).x` and `p->x` are the same	
single object, multiple possible types	`union`
bit field with *b* bits	`unsigned` *member*`:` *b*;

Operators (grouped by precedence)

struct member operator	*name*`.`*member*
struct member through pointer	*pointer*`->`*member*
increment, decrement	`++, --`
plus, minus, logical not, bitwise not	`+, -, !, ~`
indirection via pointer, address of object	`*`*pointer*`, &`*name*
cast expression to type	`(`*type*`)` *expr*
size of an object	`sizeof`
multiply, divide, modulus (remainder)	`*, /, %`
add, subtract	`+, -`
left, right shift [bit ops]	`<<, >>`
relational comparisons	`>, >=, <, <=`
equality comparisons	`==, !=`
and [bit op]	`&`
exclusive or [bit op]	`^`
or (inclusive) [bit op]	`|`
logical and	`&&`
logical or	`||`
conditional expression	*expr*$_1$ `?` *expr*$_2$ `:` *expr*$_3$
assignment operators	`+=, -=, *=, ...`
expression evaluation separator	`,`

Unary operators, conditional expression and assignment operators group right to left; all others group left to right.

Flow of Control

statement terminator	`;`
block delimiters	`{ }`
exit from `switch, while, do, for`	`break;`
next iteration of `while, do, for`	`continue;`
go to	`goto` *label*`;`
label	*label*`:` `statement`
return value from function	`return` *expr*

Flow Constructions

`if` statement	`if (`*expr*$_1$`)` *statement*$_1$ `else if (`*expr*$_2$`)` *statement*$_2$ `else` *statement*$_3$
`while` statement	`while (`*expr*`)` *statement*
`for` statement	`for (`*expr*$_1$`;` *expr*$_2$`;` *expr*$_3$`)` *statement*
`do` statement	`do` *statement* `while(`*expr*`);`

`switch` statement	`switch (`*expr*`) {`
	`case` *const*$_1$`:` *statement*$_1$ `break;`
	`case` *const*$_2$`:` *statement*$_2$ `break;`
	`default:` *statement*
	`}`

ANSI Standard Libraries

`<assert.h>`	`<ctype.h>`	`<errno.h>`	`<float.h>`	`<limits.h>`
`<locale.h>`	`<math.h>`	`<setjmp.h>`	`<signal.h>`	`<stdarg.h>`
`<stddef.h>`	`<stdio.h>`	`<stdlib.h>`	`<string.h>`	`<time.h>`

Character Class Tests `<ctype.h>`

alphanumeric?	`isalnum(c)`
alphabetic?	`isalpha(c)`
control character?	`iscntrl(c)`
decimal digit?	`isdigit(c)`
printing character (not incl space)?	`isgraph(c)`
lower case letter?	`islower(c)`
printing character (incl space)?	`isprint(c)`
printing char except space, letter, digit?	`ispunct(c)`
space, formfeed, newline, cr, tab, vtab?	`isspace(c)`
upper case letter?	`isupper(c)`
hexadecimal digit?	`isxdigit(c)`
convert to lower case	`tolower(c)`
convert to upper case	`toupper(c)`

String Operations `<string.h>`

`s` is a string; `cs`, `ct` are constant strings

length of `s`	`strlen(s)`
copy `ct` to `s`	`strcpy(s,ct)`
concatenate `ct` after `s`	`strcat(s,ct)`
compare `cs` to `ct`	`strcmp(cs,ct)`
only first `n` chars	`strncmp(cs,ct,n)`
pointer to first `c` in `cs`	`strchr(cs,c)`
pointer to last `c` in `cs`	`strrchr(cs,c)`
copy `n` chars from `ct` to `s`	`memcpy(s,ct,n)`
copy `n` chars from `ct` to `s` (may overlap)	`memmove(s,ct,n)`
compare `n` chars of `cs` with `ct`	`memcmp(cs,ct,n)`
pointer to first `c` in first `n` chars of `cs`	`memchr(cs,c,n)`
put `c` into first `n` chars of `s`	`memset(s,c,n)`

C Reference Card (ANSI)

Input/Output <stdio.h>

Standard I/O

standard input stream	stdin
standard output stream	stdout
standard error stream	stderr
end of file (type is int)	EOF
get a character	getchar()
print a character	putchar(*chr*)
print formatted data	printf("*format*",arg_1,...)
print to string s	sprintf(s,"*format*",arg_1,...)
read formatted data	scanf("*format*",&$name_1$,...)
read from string s	sscanf(s,"*format*",&$name_1$,...)
print string s	puts(s)

File I/O

declare file pointer	FILE *fp;
pointer to named file	fopen("*name*","*mode*")
modes: r (read), w (write), a (append), b (binary)	
get a character	getc(*fp*)
write a character	putc(*chr*,*fp*)
write to file	fprintf(*fp*,"*format*",arg_1,...)
read from file	fscanf(*fp*,"*format*",arg_1,...)
read and store n elts to *ptr	fread(*ptr,eltsize,n,*fp*)
write n elts from *ptr to file	fwrite(*ptr,eltsize,n,*fp*)
close file	fclose(*fp*)
non-zero if error	ferror(*fp*)
non-zero if already reached EOF	feof(*fp*)
read line to string s (< max chars)	fgets(s,max,*fp*)
write string s	fputs(s,*fp*)

Codes for Formatted I/O: "%-+ 0*w.pmc*"

-	left justify
+	print with sign
space	print space if no sign
0	pad with leading zeros
w	min field width
p	precision
m	conversion character: h short, l long, L long double
c	conversion character:
d,i	integer
c	single char
f	double (printf)
f	float (scanf)
o	octal
p	pointer
g,G	same as f or e,E depending on exponent
u	unsigned
s	char string
e,E	exponential
lf	double (scanf)
x,X	hexadecimal
n	number of chars written

Variable Argument Lists <stdarg.h>

declaration of pointer to arguments	va_list *ap*;
initialization of argument pointer	va_start(*ap*,*lastarg*);
lastarg is last named parameter of the function	
access next unnamed arg, update pointer	va_arg(*ap*,*type*)
call before exiting function	va_end(*ap*);

Standard Utility Functions <stdlib.h>

absolute value of int n	abs(n)
absolute value of long n	labs(n)
quotient and remainder of ints n,d	div(n,d)
returns structure with div_t.quot and div_t.rem	
quotient and remainder of longs n,d	ldiv(n,d)
returns structure with ldiv_t.quot and ldiv_t.rem	
pseudo-random integer [0,RAND_MAX]	rand()
set random seed to n	srand(n)
terminate program execution	exit(status)
pass string s to system for execution	system(s)

Conversions

convert string s to double	atof(s)
convert string s to integer	atoi(s)
convert string s to long	atol(s)
convert prefix of s to double	strtod(s,&endp)
convert prefix of s (base b) to long	strtol(s,&endp,b)
same, but unsigned long	strtoul(s,&endp,b)

Storage Allocation

allocate storage	malloc(size), calloc(nobj,size)
change size of storage	newptr = realloc(ptr,size);
deallocate storage	free(ptr);

Array Functions

search array for key	bsearch(key,array,n,size,cmpf)
sort array ascending order	qsort(array,n,size,cmpf)

Time and Date Functions <time.h>

processor time used by program	clock()
Example. clock()/CLOCKS_PER_SEC is time in seconds	
current calendar time	time()
$time_2$-$time_1$ in seconds (double)	difftime($time_2$,$time_1$)
arithmetic types representing times	clock_t,time_t
structure type for calendar time comps	struct tm

tm_sec	seconds after minute
tm_min	minutes after hour
tm_hour	hours since midnight
tm_mday	day of month
tm_mon	months since January
tm_year	years since 1900
tm_wday	days since Sunday
tm_yday	days since January 1
tm_isdst	Daylight Savings Time flag

convert local time to calendar time	mktime(tp)
convert time in tp to string	asctime(tp)
convert calendar time in tp to local time	ctime(tp)
convert calendar time to GMT	gmtime(tp)
convert calendar time to local time	localtime(tp)
format date and time info	strftime(s,smax,"*format*",tp)
tp is a pointer to a structure of type tm	

Mathematical Functions <math.h>

Arguments and returned values are double

trig functions	sin(x), cos(x), tan(x)
inverse trig functions	asin(x), acos(x), atan(x)
arctan(y/x)	atan2(y,x)
hyperbolic trig functions	sinh(x), cosh(x), tanh(x)
exponentials & logs	exp(x), log(x), log10(x)
exponentials & logs (2 power)	ldexp(x,n), frexp(x,&e)
division & remainder	modf(x,ip), fmod(x,y)
powers	pow(x,y), sqrt(x)
rounding	ceil(x), floor(x), fabs(x)

Integer Type Limits <limits.h>

The numbers given in parentheses are typical values for the constants on a 32-bit Unix system, followed by minimum required values (if significantly different).

CHAR_BIT	bits in char	(8)
CHAR_MAX	max value of char	(SCHAR_MAX or UCHAR_MAX)
CHAR_MIN	min value of char	(SCHAR_MIN or 0)
SCHAR_MAX	max signed char	(+127)
SCHAR_MIN	min signed char	(−128)
SHRT_MAX	max value of short	(+32,767)
SHRT_MIN	min value of short	(−32,768)
INT_MAX	max value of int	(+2,147,483,647) (+32,767)
INT_MIN	min value of int	(−2,147,483,648) (−32,767)
LONG_MAX	max value of long	(+2,147,483,647)
LONG_MIN	min value of long	(−2,147,483,648)
UCHAR_MAX	max unsigned char	(255)
USHRT_MAX	max unsigned short	(65,535)
UINT_MAX	max unsigned int	(4,294,967,295) (65,535)
ULONG_MAX	max unsigned long	(4,294,967,295)

Float Type Limits <float.h>

The numbers given in parentheses are typical values for the constants on a 32-bit Unix system.

FLT_RADIX	radix of exponent rep	(2)
FLT_ROUNDS	floating point rounding mode	
FLT_DIG	decimal digits of precision	(6)
FLT_EPSILON	smallest x so $1.0f + x \neq 1.0f$	($1.1E-7$)
FLT_MANT_DIG	number of digits in mantissa	
FLT_MAX	maximum float number	($3.4E38$)
FLT_MAX_EXP	maximum exponent	
FLT_MIN	minimum float number	($1.2E-38$)
FLT_MIN_EXP	minimum exponent	
DBL_DIG	decimal digits of precision	(15)
DBL_EPSILON	smallest x so $1.0 + x \neq 1.0$	($2.2E-16$)
DBL_MANT_DIG	number of digits in mantissa	
DBL_MAX	max double number	($1.8E308$)
DBL_MAX_EXP	maximum exponent	
DBL_MIN	min double number	($2.2E-308$)
DBL_MIN_EXP	minimum exponent	

Send comments and corrections to J.H. Silverman, Math. Dept., Brown Univ., Providence, RI 02912 USA. ⟨jhs@math.brown.edu⟩

Aiuto!

`help x`	mostra la documentazione su x
`doc`	apre la documentazione di matlab
`docsearch x`	cerca x nella documentazione

Comandi generali di matlab

Informative

`whos`	mostra tutte le variabili nel workspace
`ans`	mostra l'ultimo risultato

Pulizia

`clc`	pulisci il contenuto della finestra comandi
`clear`	cancella tutte le variabili dal workspace
`clear x`	cancella solo `x` dal workspace
`close all`	chiude le figure
`close(H)`	chiude la figura H

Caricamento e salvataggio

`save filename`	salva le variabili nel file `filename`
`save filename x,y`	salva solo le variabili x,y nel file `filename`
`save -append filename x`	salva x in un file già esistente
`load filename`	carica le variabili da file

Sistema

`addpath(string)`	aggiunge una directory dove cercare gli script
`pwd`	directory corrente
`mkdir`	crea una directory
`tempdir`	crea una directory temporanea
`exit`	esci da matlab
`dir`	stampa contenuto directory corrente

Funzioni e variabili già presenti in matlab

Generiche

`sum(x)`	somma elementi del vettore x
`prod(x)`	prodotto degli elementi del vettore x
`diff(x)`	differenze fra elementi adiacenti di x
`abs(x)`	valore assoluto; abs(-3) = 3

Arrotondamento

`floor(x)`	tronca x (floor(0.7) = 0)
`ceil(x)`	tronca per eccesso x (ceil(0.1) = 1)
`round(x)`	arrotonda x
`round(x,n)`	arrotonda x alla n-esima cifra decimale

Variabili

`pi`	3.1415...
`inf`	∞
`eps`	floating point accuracy
`1e6`	10^6

Vettori e matrici

Creazione

`j:k`	vettore riga [j, j + 1, ..., k]
`j:i:k`	vettore riga [j, j + **i**, ...]
`ones(a,b)`	matrice a×b di 1
`zeros(a,b)`	matrice a×b di 0
`x[1, 2, 3]=`	vettore riga 1x3
`x[1; 2; 3]=`	vettore colonna 3x1
`x[1, 2; 3, 4]=`	matrice 2x2

Accesso e modifica

`x(2)=4`	scrivi 4 nel secondo elemento di x
`x(:)`	tutti gli elementi di x
`x(j: end )`	gli elementi di x da j fino alla fine
`x(2:5)`	dal secondo al quinto elemento di x
`x([3,2,5])`	sottovettore di x (3^o, poi 2^o poi 5^o elemento di x)
`x(j,:)`	tutti gli elementi della riga j
`x(:,j)`	tutti gli elementi della colonna j

Operatori

`x.*y`	moltiplicazione elemento per elemento
`x./y`	divisione elemento per elemento
`x+y`	somma elemento per elemento
`x-y`	sottrazione elemento per elemento
`A'`	trasposta
`size(x)`	[righe, colonne] di x

Ricerca

`x(x>5)`	gli elementi di x maggiori di 5
`x(x>5)=0`	cambia gli elementi di x maggiori di 5 in 0
`find(A>5)`	trova gli indici degli elementi di A maggiori di 5

Layout

`[A,B]`	concatena orizzontalmente A e B
`[A;B]`	concatena verticalmente A e B

Operatori logici

Semplici valori logici

`&&`	0 && 1 == 0 etc..
`||`	0 || 1 == 1 etc..
`~`	NOT

Vettori di valori logici

`&`	AND elemento per elemnto
`|`	OR elemento per elemento
`~`	NOT elemento per elemento

Operatori relazionali

`==`	Uguaglianza
`~=`	Vero se sono differenti
`>=`	Maggiore uguale
`<=`	Maggiore uguale

Stampa

`format short`	Usa 4 cifre dopo la virgola
`format long`	Usa 16 cifre dopo la virgola
`disp(x)`	Mostra la stringa x
`num2str(x)`	Converte il numero x in una stringa
`mat2str(x)`	Converte una matrice in una stringa
`int2str(x)`	Converte un intero in una stringa
`sprintf(x)`	Converte un oggetto generico in stringa

Grafici

Creazione/Manipolazione Grafici

`fig1 = plot(x,y)`	crea plot 2d e assegna handle a fig1
`fig1 = gcf()`	assegna handle figura corrente a fig1
`fig1 = figure`	crea una nuova figura vuota
`hold on`	abilita sovrascrittura immagini
`hold off`	chiude la figura corrente

Modifica stili grafici

`set(fig1, 'LineWidth', 2)`	cambia dimensione linea
`set(fig1, 'LineStyle', '-')`	cambia stile linea
`-, --, :, -.`	possibili stili di linea
`set(fig1, 'Marker', '.')`	cambia il marker per i punti
`', +, *, x, o, square`	possibili markers
`set(fig1, 'color', 'red')`	cambia colore della linea
`red, blue, green, yellow, black`	possibili colori
`set(fig1, 'MarkerSize', 10)`	cambia la dimensione dei markers
`set(fig1, 'FontSize', 14)`	cambia la dimensione del font

Assi, griglie e leggende

`xlabel('\mu line','FontSize',14)`	assegna un nome all'asse X
`ylim([a b])`	assegna dei limiti all'asse y
`title('name','fontsize',22)`	assegna un titolo al grafico
`grid on/off;`	aggiunge/toglie una griglia
`legend('y1','y2')`	aggiunge una legenda per i plot y1 e y2

Programmazione

if/elseif/else

Esegue `bodyTrue1` se `cond1!=0`, altrimenti se `cond2!=0` esegue `bodyTrue2`, altrimenti esegue `bodyFalse12`. `elseif` così come `else` è opzionale.

```
if(cond1)
   bodyTrue1
elseif cond2
   bodyTrue2
else
   bodyFalse12
end
```

for

Esegue `body` `n` volte; ad ogni iterazione la variabile `i` viene incrementata di 1 fino ad arrivare ad `n`:

```
for i=1:n
   body
end
```

while

Esegue `body` ripetutamente finche' l'espressione `cond` non vale 0:

```
while(cond)
   body
end
```

switch/case

Esegue `bodyA` se `exp` è uguale ad `a`; oppure esegue `bodyB` se `exp` è uguale ad `b`. Se nessun caso è verificato esegue `bodyDefault`.

```
switch exp
case a
   bodyA
case b
   bodyB
...
otherwise
   bodyDefault
end
```

Data import/export

`xlsread/xlswrite`	Spreadsheets (.xls,.xlsm)
`readtable/writetable`	Spreadsheets (.xls,.xlsm)
`dlmread/dlmwrite`	text files (txt,csv)
`load/save -ascii`	text files (txt,csv)
`load/save`	matlab files (.m)
`imread/imwrite`	Image files

Revision: 0.7 - November 20, 2017