Lezione 4

Selection/Insertion Sort su interi e stringhe

Rossano Venturini

rossano.venturini@unipi.it

Pagina web del corso

http://didawiki.cli.di.unipi.it/doku.php/informatica/all-b/start

Esercizio 2

Intersezione tra array ordinati

Scrivere un programma che accetti in input due array di interi distinti e restituisca in output il numero di elementi che occorrono in entrambi gli array. Si assuma che la lunghezza di ogni array sia fornita prima dell'immissione degli elementi.

Si assuma che questa volta gli array vengano inseriti *ordinati in maniera strettamente crescente*. Si può calcolare l'intersezione in maniera più efficiente?

Dopo aver superato i test sul sito, si ripetano gli esperimenti sul dataset utilizzato nel precedente esercizio (gli array vengono forniti in ordine crescente) e si confrontino i risultati ottenuti.

`http://didawiki.cli.di.unipi.it/lib/exe/fetch.php/informatica/all-b/test_set.zip`

Esercizio 2

```
int intersection_ord(int *x, int lenx, int *y, int leny) {
  int i = 0, j = 0;
  int size = 0;
  while ( (i < lenx) && (j < leny) ) {
    if (x[i] < y[j]) {
      i++;
      continue;
    }
    if (x[i] > y[j]) {
      j++;
      continue;
    }
    size++;
    i++;
    j++;
  }
  return size;
}
```

Esercizio 4

Fusione di array ordinati

Scrivere un programma che accetti in input due array *ordinati* di interi e restituisca in output l'unione ordinata dei due array. Si assuma che la lunghezza di ogni array sia fornita prima dell'immissione degli elementi.

Per semplicità si assuma che l'intersezione tra i due array sia vuota.

Esercizio 4

```
int* sorted_union(int *x, int lenx, int *y, int leny) {
  int *u = (int *) malloc( (lenx + leny) * sizeof(int) );
  int i = 0, j = 0, pos = 0;

  while ( ( i < lenx ) && (j < leny) ) {
    if (x[i] < y[j]) {
      u[pos++] = x[i++];
      continue;
    }
    if (x[i] > y[j]) {
      u[pos++] = y[j++];
      continue;
    }
   // elementi distinti per assunzione, altrimenti?
  }
// continua :-)
```

Esercizio 4

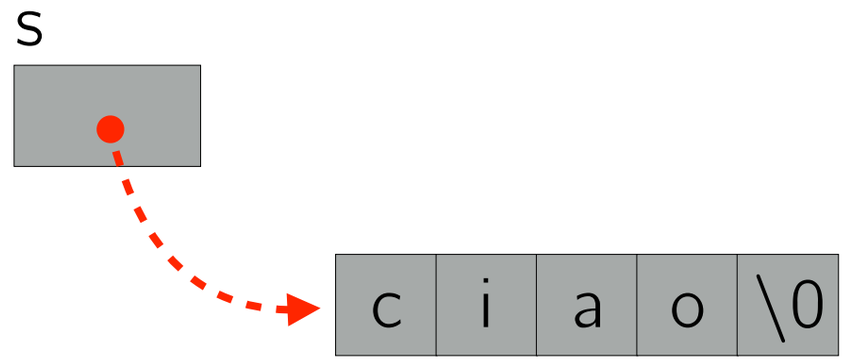
```
while ( i < lenx ) {
  u[pos++] = x[i++];
}

while ( j < leny ) {
  u[pos++] = y[j++];
}

  return u;
}
```

Stringhe

```
char *s;
s = (char *) malloc(5*sizeof(char));
scanf("%s", s);
```

Stringhe

```
char *s;
s = (char *) malloc(5*sizeof(char));
scanf("%s", s);
```



Array di Stringhe

Array di Stringhe

Scrivere un programma che legga da input un array
di N stringhe.

...

La prima riga dell'input contiene la dimensione N
dell'array. Le righe successive contengono gli elementi
dell'array, una stringa per riga. Ogni stringa ha
lunghezza massima di 100 caratteri.

Array di Stringhe

Scrivere un programma che legga da input un array di N stringhe.

...

La prima riga dell'input contiene la dimensione N dell'array. Le righe successive contengono gli elementi dell'array, una stringa per riga. Ogni stringa ha lunghezza massima di 100 caratteri.

```
int i, N;
scanf("%d", &N);
```

Array di Stringhe

Scrivere un programma che legga da input un array di N stringhe.

...

La prima riga dell'input contiene la dimensione N dell'array. Le righe successive contengono gli elementi dell'array, una stringa per riga. Ogni stringa ha lunghezza massima di 100 caratteri.

```
int i, N;
scanf("%d", &N);

char **a;
a = (char **) malloc(N * sizeof(char *));
```

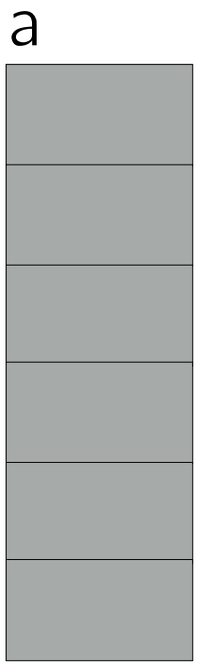
Array di Stringhe

Scrivere un programma che legga da input un array di N stringhe.

...

La prima riga dell'input contiene la dimensione N dell'array. Le righe successive contengono gli elementi dell'array, una stringa per riga. Ogni stringa ha lunghezza massima di 100 caratteri.

```
int i, N;
scanf("%d", &N);

char **a;
a = (char **) malloc(N * sizeof(char *));
```



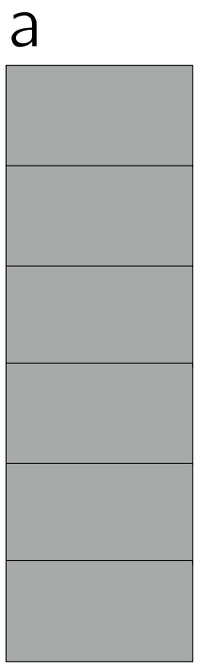
Array di Stringhe

Scrivere un programma che legga da input un array di N stringhe.

...

La prima riga dell'input contiene la dimensione N dell'array. Le righe successive contengono gli elementi dell'array, una stringa per riga. Ogni stringa ha lunghezza massima di 100 caratteri.

```
int i, N;
scanf("%d", &N);

char **a;
a = (char **) malloc(N * sizeof(char *));

for(i=0; i < N; i++) {
    a[i] = (char *) malloc(101* sizeof(char));
    scanf("%s", a[i]);
}
```



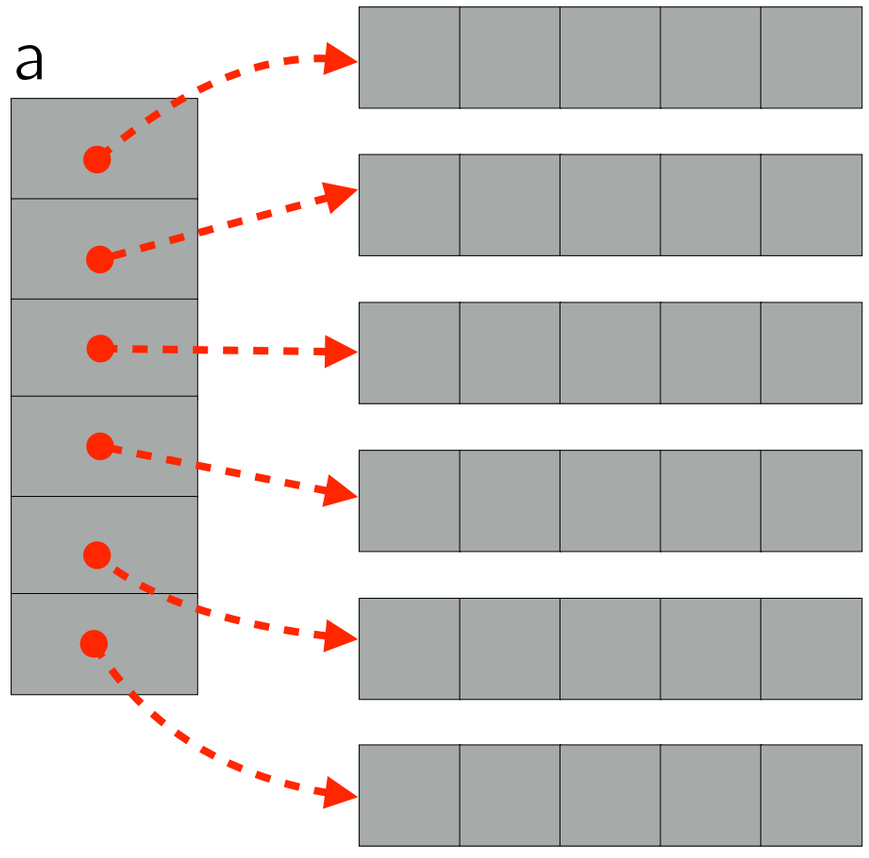
Array di Stringhe

Scrivere un programma che legga da input un array di N stringhe.

...

La prima riga dell'input contiene la dimensione N dell'array. Le righe successive contengono gli elementi dell'array, una stringa per riga. Ogni stringa ha lunghezza massima di 100 caratteri.

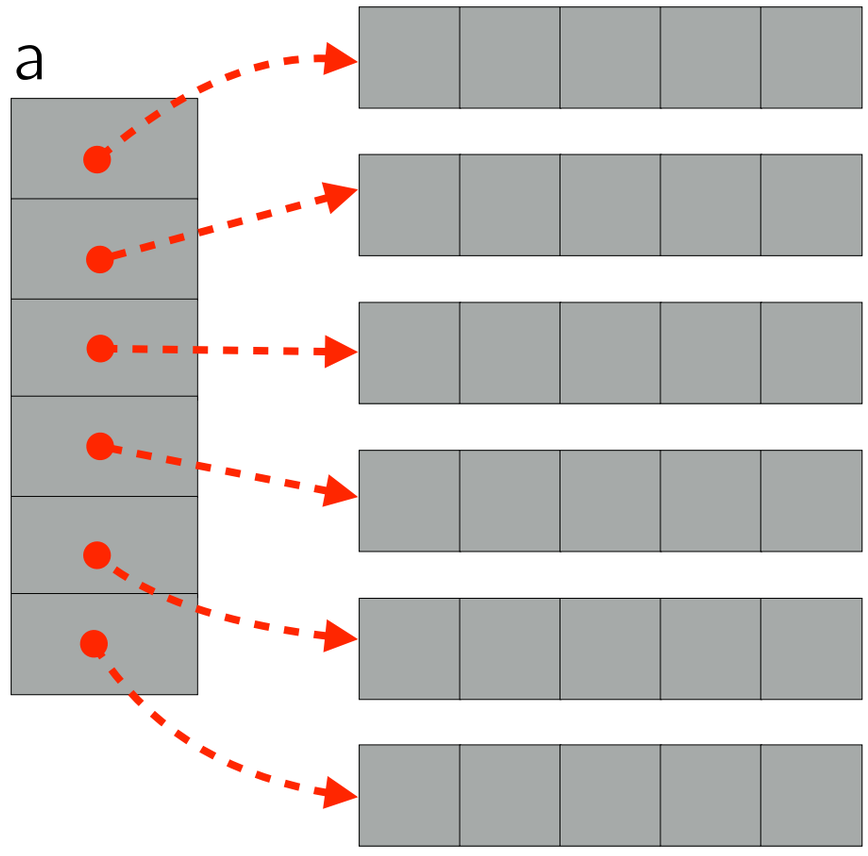
```
int i, N;
scanf("%d", &N);

char **a;
a = (char **) malloc(N * sizeof(char *));

for(i=0; i < N; i++) {
    a[i] = (char *) malloc(101* sizeof(char));
    scanf("%s", a[i]);
}
```



Array di Stringhe

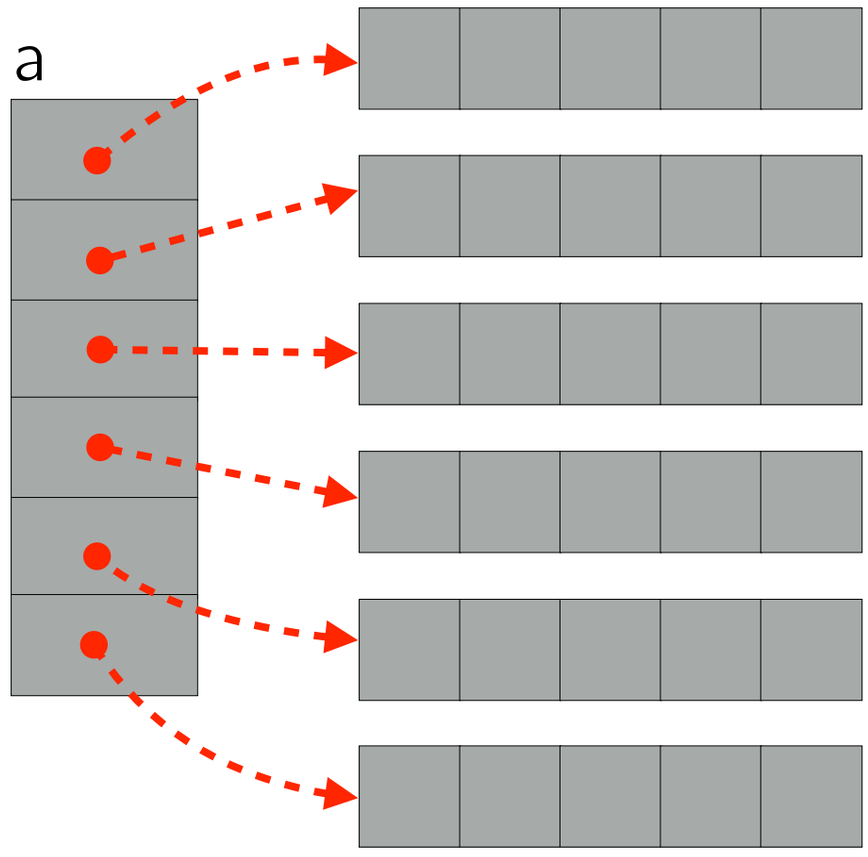
```
for(i=0; i < N; i++) {
    printf("%s", a[i]);
}
```

```
int i, N;
scanf("%d", &N);

char **a;
a = (char **) malloc(N * sizeof(char *));

for(i=0; i < N; i++) {
    a[i] = (char *) malloc(101* sizeof(char));
    scanf("%s", a[i]);
}
```



Rilasciare la memoria

```
for(i=0; i < N; i++) {
    free(a[i]); // liberare ogni singola stringa
}

free(a); // liberare l'array
```



Esercizio 1

Selection Sort su interi

Scrivere una funzione che, dato un array di interi e la sua lunghezza, lo ordini utilizzando l'algoritmo **Selection Sort**.

Scrivere un programma che utilizzi la funzione per ordinare un array di N interi letti da input e stampi in output gli elementi dell'array ordinato.

La prima riga dell'input contiene la dimensione N dell'array. Le righe successive contengono gli elementi dell'array, uno per riga.
L'output contiene gli elementi dell'array ordinato, uno per riga.

Esercizio 2

Insertion Sort su interi

Scrivere una funzione che, dato un array di interi e la sua lunghezza, lo ordini utilizzando l'algoritmo **Insertion Sort**.

Scrivere un programma che utilizzi la funzione per ordinare un array di N interi letti da input e stampi in output gli elementi dell'array ordinato.

La prima riga dell'input contiene la dimensione N dell'array. Le righe successive contengono gli elementi dell'array, uno per riga.
L'output contiene gli elementi dell'array ordinato, uno per riga.

Esercizio 3

Insertion Sort su stringhe

Scrivere una funzione che, dato un array di stringhe e la sua lunghezza, lo ordini utilizzando l'algoritmo **Insertion Sort**.

Scrivere un programma che utilizzi la funzione per ordinare un array di N stringhe lette da input e stampi in output gli elementi dell'array ordinato. Assumere che la lunghezza massima di una stringa sia 100 caratteri.

Si può utilizzare la funzione `strcmp` in `string.h` per confrontare lessicograficamente due stringhe. Utilizzare il comando `man strcmp` per maggiori informazioni.

La prima riga dell'input contiene la dimensione N dell'array. Le righe successive contengono gli elementi dell'array, una stringa per riga. Ogni stringa ha lunghezza massima di 100 caratteri.
L'output contiene gli elementi dell'array ordinato, una stringa per riga.

Esercizio 4

Ricerca binaria su stringhe

Scrivere una funzione che, data una stringa, un array di stringhe distinte e ordinate lessicograficamente e la sua lunghezza, cerchi la stringa nell'array utilizzando la ricerca binaria. La funzione restituisce la posizione della stringa se essa è presente, il valore `-1` altrimenti.

Scrivere un programma che implementi il seguente comportamento. L'input è formato da una prima riga contenente la lunghezza N dell'array. Le successive N righe contengono le stringhe dell'array ordinate lessicograficamente.

Segue una sequenza di dimensione non nota di richieste espresse con coppie. La prima riga di ogni coppia è un valore che può essere "`0`" o "`1`". Se il valore è `0`, il programma termina (non ci sono più richieste). Se il valore è "`1`", sulla riga successiva si trova una stringa da cercare.

Per ciascuna richiesta ci si aspetta in output l'esito della ricerca: la posizione della stringa nell'array se essa è presente, `-1` altrimenti.

Puzzle

Cavalli
(*Folklore puzzle and Facebook's interview process.*)

Si vuole stabilire quali sono i 3 cavalli più veloci in un insieme di 25. Possono essere organizzate gare tra 5 cavalli scelti in qualunque modo dall'insieme. Alla fine di ogni gara si conosce l'ordine di arrivo ma non il tempo impiegato dai cavalli. Qual è il numero minimo di gare necessario per identificare i 3 cavalli più veloci?

Puzzle

Luci al Pirellone
(da *Olimpiadi Italiane di Informatica, 2005)*

Il Pirellone è un noto grattacielo di Milano, in cui le finestre sono disposte ordinatamente per N righe (piani) e M colonne. Le righe sono numerate da 1 a N (dall'alto in basso) e le colonne da 1 a M (da sinistra a destra). Non tutti i dipendenti spengono la luce dei loro uffici, la sera prima di uscire. Quindi alcune finestre rimangono illuminate e tocca al custode provvedere a spegnerle. Per facilitare il compito del custode, sono stati predisposti $N + M$ interruttori speciali, con un funzionamento particolare. Ci sono N interruttori di riga e M interruttori di colonna. Quando il custode agisce sull'i-esimo interruttore di riga, tutte le luci accese dell'i-esima riga si spengono ma, allo stesso tempo, quelle spente si accendono! Analogamente alle righe, un interruttore di colonna spegne le luci accese di quella colonna e accende quelle spente.

Aiuta il custode a decidere quali degli $N + M$ interruttori azionare al fine di spegnere tutte le luci delle finestre del Pirellone. Data la configurazione iniziale di luci, il custode deve verificare se sia possibile spegnere le luci con gli interruttori speciali e, in tal caso, deve specificare anche su quali interruttori agire. Altrimenti, stabilisce che è necessario utilizzare gli interruttori tradizionali.