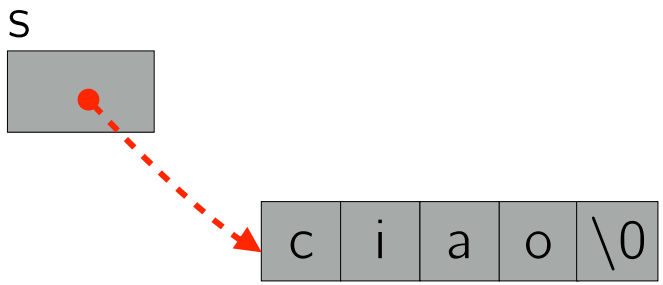
Stringhe

```
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 *));
```

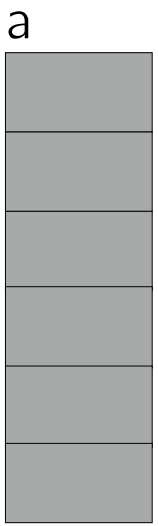
Array di Stringhe

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 *));
```



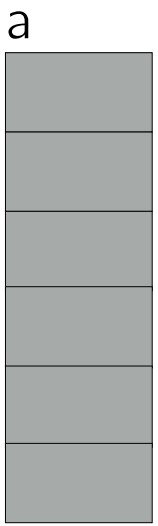
Array di Stringhe

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]);
}
```



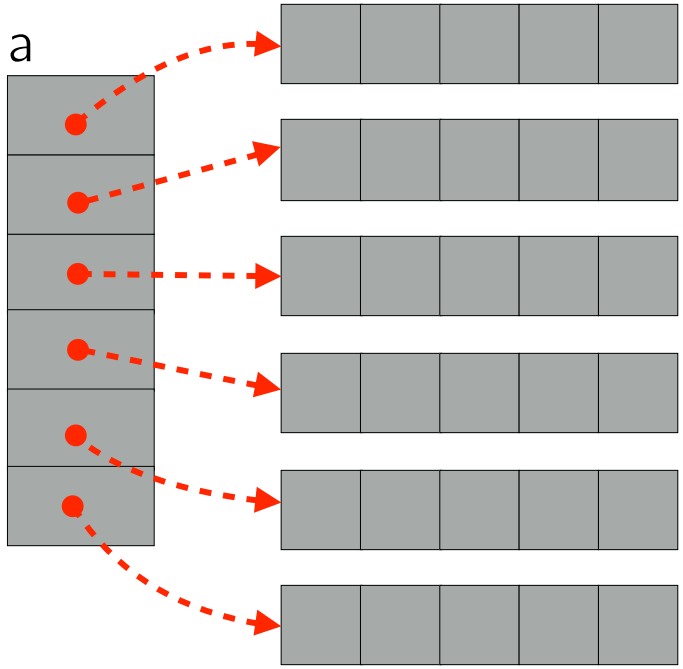
Array di Stringhe

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.

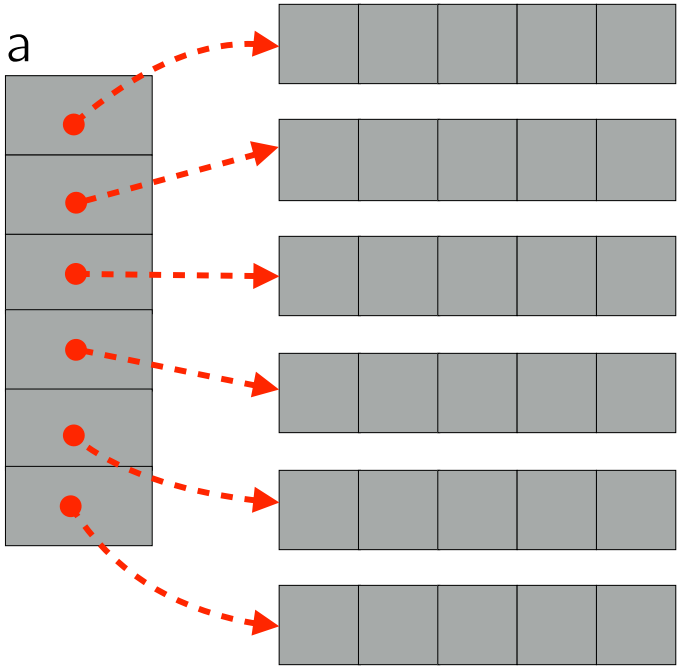
```
int i, N;
scanf("%d", &N);

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

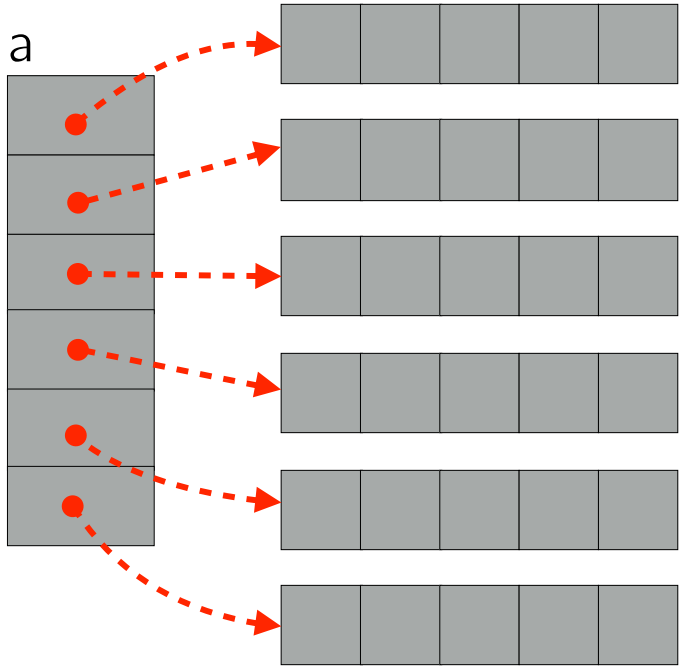
```
for(i=0; i < N; i++) {
    printf("%s", a[i]);
}
```

```
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

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

```
void insertionSort(char **A, int len) {
  int i, j;
  char *key;

  for(i = 1; i < len; i++) {
    key = A[i];
    j = i - 1;
    while (( j >= 0 ) && (strcmp(A[j], key) > 0)) {
      A[j+1] = A[j]; // scambia i puntatori
      j--;
    }
    A[j+1] = key;
  }
}
```

Esercizio

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.

Esercizio

```
int binsearch(char **dict, int left, int right, char *str) {

  if (left > right) {
    return -1;
  }

  int pos = (left+right)/2; // floor
  int cmp = strcmp(str, dict[pos]);

  if (cmp == 0) return pos;
  if (cmp < 0) {
     return binsearch(dict, left, pos-1, str);
  } else {
     return binsearch(dict, pos+1, right, str);
  }
}
```