Fondamenti di Informatica

Ing. Biomedica

Esercitazione n.9

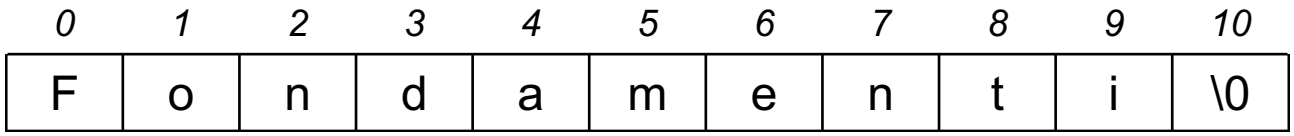
Stringhe & Algoritmi di ordinamento

Antonio Arena
antonio.arena@ing.unipi.it

Stringhe

- In C++ non esiste un vero e proprio tipo stringa
- Viene considerata con un array di caratteri terminata dal carattere speciale '\0', o meglio noto come *fine stringa*. Ogni elemento dell'array contiene un singolo carattare!
- *Come si definisce?*

1. `char str[100];` → array di 100 char, posso inserirci al più 99 caratteri (compresi gli spazi), perché l'ultimo è riservato al fine stringa
2. `char str[] = ''Fondamenti'';`
   → inizializzazione con doppi apici. Il compilatore riesce a capire la lunghezza giusta per l'array. (In questo caso la lunghezza dell'array sarebbe **11:** 10 per la parola + 1 per il fine stringa



Passaggio di parametri stringhe

- Il passaggio di una stringa è equivalente al passaggio di un qualunque array monodimensionale già visto nelle esercitazioni precedenti.

- Esempio:
  supponiamo di avere una funzione che conta quanti caratteri contiene la stringa, escludendo il fine stringa

```
int lunghezza(char *str);
```

  In un main si avrà:

```
char stringa[100];
…. altre operazioni …
int lun = lunghezza(stringa);
oppure
int lun = lunghezza(&stringa[0]);
```

Input/Output di stringhe

- In C++ le operazioni di lettura da tastiera (cin) e stampa a video (cout) sono ottimizzate per le stringhe.
- Supponendo di avere
  `char stringa[100];` → posso salvarci dentro massimo 99 char
- Lettura: basta fare `cin >> stringa;` → Il cin si occuperà di salvare carattere per carattere nell'array
  ATTENZIONE: se proviamo a leggere una frase contenente più parole separate da spazi, il comando di su salverebbe nella stringa SOLO la prima parola, ignorando il resto.
- Esempio: supponiamo di scrivere da tastiera «Winter is coming», nell'array dopo la lettura troveremmo la stringa «Winter\0»…
- Per poter leggere senza ignorare gli spazi, il comando da tastiera è `cin.getline(stringa,sizeof(stringa));`
- Scrittura: basta fare `cout << stringa;` → Il cout si occuperà di andare a leggere carattere per carattere nell'array e stamparli a video, senza ignorare gli spazi.

Esercizio 1

- Scrivere una funzione che prende in ingresso una stringa e ritorna la lunghezza della stringa, escludendo dal conteggio il fine stringa. *Non usare funzioni di libreria!!*
- Scrivere un main che legga da tastiera la stringa, la stampi, e poi richiami la funzione scritta.
- Intestazione: `int lunghezza(char* str);`

- Esempio: supponiamo di passare la seguente stringa

W	i	n	t	e	r		i	s		c	o	m	i	n	g	\0	@	#	!	?

L'output della funzione con questa stringa verrà **16**.

Ricordate, l'array non deve essere necessariamente pieno, è il fine stringa che fa capire dove termina la stringa.

Esercizio 1 - soluzione

```
 1 #include <iostream>
 2 using namespace std;
 3
 4 int lunghezza(char* str){
 5     if(str==NULL){
 6         return 0;
 7     }
 8
 9     int conta=0;
10     while(str[conta]!='\0')
11         conta++;
12     return conta;
13 }
14
15 int main(){
16     char stringa[100];
17
18     cin >> stringa;
19     //cin.getline(stringa,sizeof(stringa));
20     cout << stringa << endl;
21
22     int len = lunghezza(stringa);
23     cout << "lunghezza: " << len << endl;
24
25     return 0;
26 }
```

Esercizio 2

- Scrivere una funzione che prende in ingresso due stringhe e copi la seconda stringa all'interno della prima.
  Non usare funzioni di libreria!!
- Scrivere un main che legga da tastiera due stringhe, la stampi, e poi richiami la funzione scritta e che ristampi poi le due stringhe.

- Intestazione:
  `void copia_stringa(char* dest, char* source);`

Esercizio 2

- Esempio:
- `char source[100] = ‘‘Winter is coming’’;`

W	i	n	t	e	r		i	s		c	o	m	i	n	g	\0	@	#	!	?

- `char dest[100] = ‘‘Hear me roar!’’;`

H	e	a	r		m	e		r	o	a	r	!	\0	2	g	6	r	Z	h	^

. . . dopo aver chiamato la funzione:

- source

W	i	n	t	e	r		i	s		c	o	m	i	n	g	\0	@	#	!	?

- dest

W	i	n	t	e	r		i	s		c	o	m	i	n	g	\0	r	Z	h	^

Esercizio 2 - Soluzione

```
1  #include <iostream>
2  using namespace std;
3
4  void copia_stringa(char* dest, char* source){
5      if(dest==NULL || source == NULL){
6          return;
7      }
8
9      int i=0;
10     while(source[i]!='\0'){
11         dest[i] = source[i];
12         i++;
13     }
14     dest[i] = '\0';
15 }
16
17 int main(){
18     char dest[100];
19     char source[100];
20
21     cin.getline(source, sizeof(source));
22     cin.getline(dest, sizeof(dest));
23
24     cout << "Stringa source: " << source << endl;
25     cout << "Stringa dest: " << dest << endl;
26
27     copia_stringa(dest, source);
28
29     cout << "Stringa source: " << source << endl;
30     cout << "Stringa dest: " << dest << endl;
31
32     return 0;
33 }
```

E dopo averle risolte….

- Esistono le librerie!

```
 1 #include <iostream>
 2 #include <cstring>
 3 using namespace std;
 4
 5 int main(){
 6     char source[100] = "Winter is coming";
 7     char dest[100] = "What is dead may never die";
 8
 9     int len_dest = strlen(dest);
10     int len_source = strlen(source);
11
12     cout << "Source: " << source << " len: " << len_source << endl;
13     cout << "Dest: " << dest << " dest: " << len_dest << endl;
14
15     strcpy(dest,source);
16     cout << "Source: " << source << " len: " << strlen(source) << endl;
17     cout << "Dest: " << dest << " dest: " << strlen(dest) << endl;
18     return 0;
19 }
```

Ordinamento – Bubble Sort

```
1  #include <iostream>
2  using namespace std;
3
4  const int N = 5;
5
6  void scambia(int *vettore, int i, int j){
7      int temp = vettore[i];
8      vettore[i] = vettore[j];
9      vettore[j] = temp;
10 }
11
12
13 void bubble(int vettore[], int n)
14 {
15     for (int i = 0 ; i < n-1; i++)
16         for (int j = n-1; j > i; j--)
17             if (vettore[j] < vettore[j-1])
18                 scambia(vettore, j, j-1);
19 }
20
21 int main(){
22     int vettore[N];
23
24     cout << "Inserisci 5 numeri elementi" << endl;
25
26     for(int i=0;i<N;i++)
27         cin >> vettore[i];
28
29     cout << "Vettore non ordinato: ";
30     for(int i=0;i<N;i++)
31         cout << vettore[i] << " ";
32     cout << endl;
33
34     bubble(vettore,N);
35
36     cout << "Vettore ordinato: ";
37     for(int i=0;i<N;i++)
38         cout << vettore[i] << " ";
39     cout << endl;
40     return 0;
41 }
```

Ordinamento – Bubble Sort Ricorsiva

```
 1 #include <iostream>
 2 using namespace std;
 3
 4 const int N = 5;
 5
 6 void scambia(int *vettore, int i, int j){
 7     int temp = vettore[i];
 8     vettore[i] = vettore[j];
 9     vettore[j] = temp;
10 }
11
12
13 void bubble_ricorsiva(int vettore[], int n)
14 {
15     if (n == 1)
16         return;
17
18     for (int i=0; i<n-1; i++)
19         if (vettore[i] > vettore[i+1])
20             scambia(vettore, i, i+1);
21
22     bubble_ricorsiva(vettore, n-1);
23 }
24 int main(){
25     int vettore[N];
26
27     cout << "Inserisci 5 numeri elementi" << endl;
28
29     for(int i=0;i<N;i++)
30         cin >> vettore[i];
31
32     cout << "Vettore non ordinato: ";
33     for(int i=0;i<N;i++)
34         cout << vettore[i] << " ";
35     cout << endl;
36
37     bubble_ricorsiva(vettore,N);
38
39     cout << "Vettore ordinato: ";
40     for(int i=0;i<N;i++)
41         cout << vettore[i] << " ";
42     cout << endl;
43     return 0;
44 }
```

Ragionamento di funzionamento:

1. Caso base: se l'array è composto di un solo elemento, non lo ordino (perché lo è già)
2. Se ho più di un elemento, lo scorro tutto, e confronto a due a due gli elementi, ed eventualmente faccio lo scambio in modo da avere il maggiore in fondo all'array (ciclo for).
3. A questo punto, chiamo la bubble su un array di dimensione n-1

Ordinamento – Bubble Sort Ricorsiva

Esempio di funzionamento: ho un array di 4 interi [7, 3, 0, -10]

chiamo bubble_rec(vettore,4)
non è il caso base, eseguo il ciclo for
da [7, 3, 0, -10] ottengo [3, 0, -10, 7] → il più grande è sistemato

chiamo bubble_rec(vettore,3), cioè la chiamo sul vettore [3, 0, -10]
non è il caso base, eseguo il ciclo for
da [3, 0, -10] ottengo [0, -10, 3]

richiamo bubble_rec(vettore,2), cioè la chiamo sul vettore [0, -10]
non è il caso base, eseguo il ciclo for
da [0, -10] ottengo [-10, 0]

richiamo bubble_rec(vettore,1), cioè la chiamo sul vettore [-10]
è il caso base, non faccio nulla.

tutte le chiamate terminano, e il mio vettore ordinato è [-10, 0, 3, 7]

Esercizi per casa - 1

- Scrivere una funzione che prenda in ingresso un array di 5 stringhe di al più 100 caratteri e lo ordini in ordine crescente di lunghezza delle stringhe.

- Suggerimenti:
  1. considerare l'array di stringhe come matrice di char la cui prima dimensione è 5, e la seconda è 100.
  2. le stringhe vanno lette da tastiera nel main
  3. attenzione al confronto nella bubble, cosa vanno confrontate?
  4. attenzione alla scambia, non si può usare l'assegnamento (cioè il comando «=») tra stringhe…

Esercizi per casa - 1

- Esempio:

Prima dell'ordinamento

Stringa 0 len: 16	W	i	n	t	e	r		i	s		c	o	m	i	n	g	\0		
Stringa 1 len: 12	H	e	a	r		m	e		r	o	a	r	\0						
Stringa 2 len: 14	F	i	r	e		a	n	d		B	l	o	o	d	\0				
Stringa 3 len: 13	W	e		d	o		n	o	t		s	o	w	\0					
Stringa 4 len: 20	O	u	r		b	l	a	d	e	s		a	r	e		s	h	a	r	p	\0

Dopo l'ordinamento

Stringa 0 len: 12	H	e	a	r		m	e		r	o	a	r	\0						
Stringa 1 len: 13	W	e		d	o		n	o	t		s	o	w	\0					
Stringa 2 len: 14	F	i	r	e		a	n	d		B	l	o	o	d	\0				
Stringa 3 len: 16	W	i	n	t	e	r		i	s		c	o	m	i	n	g	\0		
Stringa 4 len: 20	O	u	r		b	l	a	d	e	s		a	r	e		s	h	a	r	p	\0