



ELEMENTI DI INFORMATICA

DOCENTE: FRANCESCO MARRA

INGEGNERIA CHIMICA

INGEGNERIA ELETTRICA

SCIENZE ED INGEGNERIA DEI MATERIALI

INGEGNERIA GESTIONALE DELLA LOGISTICA E DELLA PRODUZIONE

INGEGNERIA NAVALE

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

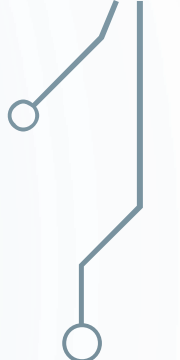


TIPI STRUTTURATI: STRINGHE

GENERAZIONE NUMERI PSEUDO-CAUSALI



AGENDA

- Tipi strutturati
 - Stringhe
 - Generazione di numeri pseudo-casuali
- 

TIPI STRUTTURATI: STRINGHE



LE STRINGHE DI CARATTERI

- Una stringa è una sequenza di caratteri dotata di una lunghezza

- Le stringhe sono definite come array di `char`

```
char nome_stringa[dimensione];
```

- Esempio:

```
char s[10];
```

- È possibile definire un ***tipo stringa***

```
const int N=10;
```

```
typedef char Stringa[N];
```

```
Stringa s;
```

LE STRINGHE DI CARATTERI

- La stringa viene conclusa con un carattere speciale detto *terminatore di stringa*, corrispondente al carattere ASCII NULL
 - Si indica con `'\0'`
- La seguente dichiarazione definisce una sequenza formata da massimo 9 caratteri

```
char s[10];
```

INIZIALIZZAZIONE DI UNA STRINGA

- Una stringa può essere inizializzata elencando i caratteri che devono farne parte, compreso il terminatore

```
char s[10] = {'c', 'a', 's', 'a', '\\0'};
```

- Solo per le stringhe è consentita una forma di inizializzazione più compatta usando le costanti stringa
 - Il terminatore viene aggiunto automaticamente in fondo

```
char s[10] = "casa";
```

ASSEGNAZIONE DI VALORI AD UNA STRINGA

- Doppia modalità di assegnazione:

```
nome = "Angelo";
```

```
nome[0] = 'A';
```

```
nome[1] = 'n';
```

```
nome[2] = 'g';
```

```
nome[3] = 'e';
```

```
nome[4] = 'l';
```

```
nome[5] = 'o';
```

```
nome[6] = '\\0';
```


OPERAZIONI DI I/O PER UNA STRINGA

- La seguente operazione stampa a video tutti i caratteri fino al terminatore

```
cout << nome;
```

- La seguente operazione carica nel vettore tutti i caratteri inseriti da tastiera e aggiunge il terminatore in coda automaticamente

```
cin >> nome;
```

TIPICHE OPERAZIONI SU STRINGHE

- Elaborazioni tipiche delle stringhe sono le seguenti:
 - Calcolo della loro lunghezza
 - Assegnazione di una stringa ad un'altra
 - Concatenazione di una stringa ad un'altra
 - Confronto tra due stringhe
 - Estrazione di una sottostringa da una stringa fissando la posizione da cui iniziare ad estrarre il numero di caratteri

CALCOLO DELLA LUNGHEZZA DI UNA STRINGA

```
lung = 0;  
while (s[lung] != '\0') {  
    lung++;  
}
```

CONFRONTO TRA DUE STRINGHE

```
uguali = (lun1 == lun2);    // solo se le lunghezze sono uguali
                             // la variabile è TRUE, altrimenti è FALSE

i = 0;
while (uguali && (s1[i] != '\0')) {    // si noti che il ciclo ha inizio
                                       // solo se le lunghezze sono uguali
    uguali = (s1[i] == s2[i]);
    i++;
}

/* il ciclo termina quando due caratteri sono diversi rendendo uguali FALSE
   o perché si è raggiunta la posizione della terminazione con la variabile
   uguali rimasta TRUE
*/
```

FUNZIONI DI LIBRERIA

Funzione	Obiettivo
strlen	Calcola la lunghezza di una stringa
strcpy	Copia una stringa in un'altra
strncpy	Copia n caratteri di una stringa in un'altra
strcat	Accoda una stringa ad un'altra
strncat	Accoda n caratteri di una stringa ad un'altra
strcmp	Confronta due stringhe
strncmp	Confronta i primi n caratteri di due stringhe
strchr	Cerca un carattere in una stringa
strstr	Cerca una stringa in un'altra

FUNZIONI DI LIBRERIA

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

```
stringa s1 = "abcdef", s2 = "ghilmn", s3;
```

Funzione	Argomenti	Risultati	Esempi di uso	Valori calcolati
strlen	1) Stringa di cui calcolare la lunghezza	La lunghezza della stringa di tipo int	<code>lung = strlen(s1);</code>	•6
strcpy	1) Stringa in cui copiare 2) Stringa da cui copiare	La stringa copia nel caso in cui è in grado di contenere l'altra	<code>strcpy(s3,s1);</code> <code>strcpy(s3,"Angelo");</code>	•La stringa s3 diventa uguale a "abcdef" •Assegna a s3 la stringa costante
strncpy	1) Stringa in cui copiare 2) Stringa da cui copiare 3) Numero di caratteri da copiare	La stringa copia nel caso in cui è in grado di contenere gli n caratteri dell'altra	<code>strncpy(s3,s1,3);</code>	•La stringa s3 diventa uguale a "abc"

FUNZIONI DI LIBRERIA

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

```
stringa s1 = "abcdef", s2 = "ghilmn", s3;
```

Funzione	Argomenti	Risultati	Esempi di uso	Valori calcolati
strcat	1) Stringa a cui accodare 2) Stringa da cui accodare	La stringa a cui è stata accodata la seconda nel caso in cui è in grado di contenerla	<code>strcat(s1,s2);</code> <code>strcat(s1,"123");</code>	•s1 diventa uguale a "abcdefghilmn"; •s1 diventa uguale a "abcdef123"
strncat	1) Stringa a cui accodare 2) Stringa da cui accodare 3) Numero di caratteri da accodare	La stringa a cui sono stati accodati gli n caratteri della seconda nel caso in cui è in grado di contenerli	<code>strncat(s1,s2,3);</code>	•s1 diventa uguale a "abcdefghi"

FUNZIONI DI LIBRERIA

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

```
stringa s1 = "abcdef", s2 = "ghilmn", s3;
```

Funzione	Argomenti	Risultati	Esempi di uso	Valori calcolati
strcmp	1) Prima stringa 2) Seconda stringa da confrontare con la prima	Un valore intero con l'esito del confronto: •=0 se $s1=s2$ •<0 se $s1<s2$ •>0 se $s1>s2$	<code>esito=strcmp(s1,s2);</code> <code>esito=strcmp(s1,"abcdefg");</code>	•minore di zero •uguale a zero
strncmp	1) Prima stringa 2) Seconda stringa da confrontare con la prima 3) I primi n caratteri da confrontare	Un valore intero con l'esito del confronto: •=0 se $s1=s2$ •<0 se $s1<s2$ •>0 se $s1>s2$	<code>esito=strncmp(s1,"abczwr",3);</code> <code>esito=strncmp(s2,"a",1);</code>	•uguale a zero •maggiore di zero

FUNZIONI DI LIBRERIA

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

```
stringa s1 = "abcdef", s2 = "ghilmn", s3;
```

Funzione	Argomenti	Risultati	Esempi di uso	Valori calcolati
strchr	1) Stringa in cui cercare 2) Carattere da cercare	Un puntatore con la posizione del carattere se è presente, o con valore NULL se non è presente	<code>ptr=strchr(s1,'f');</code> <code>ptr=strchr(s1,'l');</code>	•diverso da NULL •uguale a NULL
strstr	1) Stringa in cui cercare 2) Stringa da cercare	Un puntatore con la posizione della stringa se è presente, o con valore NULL se non è presente	<code>ptr=strstr(s1,s2);</code> <code>ptr=strstr(s1,"bcd");</code>	•uguale a NULL •diverso da NULL

ESERCIZI

- Calcolo frequenze iniziali dei nomi

- Scrivere un programma tramite il quale:

- Si inseriscono dei nomi da tastiera, finché non si inserisce il carattere '.'
- Si trasforma in maiuscola l'iniziale se è minuscola
- Si trova un vettore di frequenze delle iniziali dei nomi
- Si stampa a video il numero di elementi totali
- Si stampano a video gli elementi non nulli di questo vettore

■ Esempio di esecuzione

```
Calcolo frequenza delle iniziali dei nomi di un elenco

Inserisci nome, oppure '.' se l'elenco e' finito: Marco
Inserisci nome, oppure '.' se l'elenco e' finito: Luca
Inserisci nome, oppure '.' se l'elenco e' finito: marianna
Il nome inserito non comincia con la maiuscola, diventa: Marianna
Inserisci nome, oppure '.' se l'elenco e' finito: .

Numero di elementi: 3

Frequenza delle iniziali dei nomi inseriti:
L: 1
M: 2

-----
Process exited after 6.828 seconds with return value 0
Premere un tasto per continuare . . .
```

ESERCIZI

- Ricerca di stringhe
 - Scrivere un programma tramite il quale:
 - Si inseriscono dei nomi da tastiera, finché non si inserisce il carattere '.'
 - Si costruisce un vettore di stringhe, con i nomi inseriti
 - Si inserisce da tastiera un ulteriore nome, per verificare se è stato già inserito
 - Si stampa a video un risultato

■ Esempio di esecuzione

Questo programma consente di verificare la presenza di un nome tra quelli inseriti in precedenza.

*Inserisci nome, oppure '.' se l'elenco e' finito: **Marco***

*Inserisci nome, oppure '.' se l'elenco e' finito: **Luca***

Inserisci nome: .

*Inserisci nome da cercare: **Luca**
Il nome cercato e' contenuto nell'elenco.*

GENERAZIONI NUMERI PSEUDO-CASUALI



NUMERI RANDOM

- In C++ è possibile generare un numero casuale, più precisamente pseudo-random

- Sintassi:

```
#include<cstdlib>
```

```
int random;
```

```
random=rand();
```

- Tramite la funzione `rand()`, viene generato un numero **intero** compreso nell'intervallo `[0, RAND_MAX]`.
- Il valore di `RAND_MAX` dipende dal compilatore.

NUMERI RANDOM

- A partire dagli interi generati nell'intervallo $[0, \text{RAND_MAX}]$, è possibile:

- Generare un numero **intero** nell'intervallo desiderato $[\text{Nmin}, \text{Nmax}]$:

```
int Nrand;
```

```
Nrand=rand()%(Nmax-Nmin+1)+Nmin;
```

- Generare un numero **reale** nell'intervallo $[0, 1]$:

```
float Rrand01;
```

```
Rrand01=float(rand())/RAND_MAX;
```

- Generare un numero **reale** nell'intervallo desiderato $[\text{Rmin}, \text{Rmax}]$:

```
float Rrand;
```

```
Rrand = Rrand01*(Rmax-Rmin)+Rmin;
```

NUMERI RANDOM: IL SEME

- La generazione di numeri pseudo-random viene fatta attraverso algoritmi deterministici che producono una sequenza di numeri «apparentemente» randomica.
- Prima di usare `rand()` bisogna inizializzare il valore chiamato «seed» (seme), usando la funzione `srand(seed)`
- Per ovviare, è accettato di inizializzare il valore del seme con un valore legato all'orologio di sistema:

```
#include<cstdlib>

#include<ctime>

using namespace std;

srand((unsigned int)time(NULL));

int random;

random=rand();
```

ESERCIZI

- Realizzare un programma che:
 - genera casualmente un numero N compreso tra 1 e 10;
 - genera casualmente N valori reali compresi nell'intervallo $[-1\,000, 1\,000]$ e li assegna agli elementi di un vettore;
 - calcola la media degli elementi del vettore.

DOMANDE, DUBBI, PERPLESSITÀ

