

# Le Stringhe

## *Un'introduzione "operativa"*

Luigi Palopoli

ReTiS Lab - Scuola Superiore S. Anna

Viale Rinaldo Piaggio 34

Pontedera - Pisa Tel. 050-883444

Email: [palopoli@sssup.it](mailto:palopoli@sssup.it)

URL: <http://feanor.sssup.it/> luigi

# Generalità sulle stringhe in C/C++

- In C e in C++ non esiste un vero e proprio tipo stringa.
- Una stringa é vista come una sequenza di caratteri terminata da `\0`, contenuta in uno spazio in memoria (buffer, spesso implementato come array).
- La gestione delle stringhe in standard C é facilitato da una serie di funzioni contenute in `<string.h>`, e `<stdlib.h>`
- Le librerie standard del C++ forniscono la classe `string`, (header `<string>`) che evita tutte le problematiche di allocazione dei buffer

# Richiami sulle stringhe in C

- In C una stringa é un array di caratteri (attenti alle dimensioni del buffer!!)

```
char s0[11]; // contiene una stringa di 10 caratteri
```

```
char s1[] = "stringa contenuta in s1" ; // con inizializz.
```

```
char s2[21] = "ciao" // array parzialmente riempito
```

```
char *ps0 = "Il fiume Arno attraversa Pisa" ; // si alloca la stringa
```

```
    // accedendovi tramite ps0
```

```
const char *ps1 = Buongiorno ; // per evitare overflow
```

```
    // sul buffer della stringa
```

# Le stringhe del C++

- Le stringhe del C++ sono classi *comodissime* che forniscono tutte le funzionalità delle classiche stringhe del C

```
#include <string>
```

```
....
```

```
string Name1, Name2;
```

# Le stringhe del C++

- Le stringhe del C++ sono classi *comodissime* che forniscono tutte le funzionalità delle classiche stringhe del C

```
#include <string>
```

```
....
```

```
string Name1, Name2;
```

Non confondere con

**string.h** che é l'interfaccia alle librerie del C

# Assegnamento

- Ad una variabile di tipo stringa é possibile assegnare un valore di una stringa a la C (array di char o costante tra “..”), un carattere o ad un'altra variabile di tipo stringa

```
string nome1, nome2, miastringa;  
const char a=12;  
nome1 = ' 'Che bello!!' ' ;  
nome2 = nome1;  
miastringa='n' ;  
miastringa=a;
```

# Inizializzazione

- É possibile inizializzare una variabile stringa all'inizializzazione

```
string saluto="Ciao",  
    saluto1=saluto;
```

- non é possibile inizializzazione con caratteri o interi (...vedi costruttore di copia....)

```
string s='c',        //ILLEGALE!!  
    saluto1=3;      //ILLEGALE!!
```

# Assegnamenti multilinea

- Non é possibile andare a capo in un assegnamento a costante stringa

```
string sbagliata= "stringa veramente troppo, //ERRORE!  
lunga ";
```

- ...cosí peró va bene...

```
string corretta= "stringa veramente troppo, \n "  
"lunga ";
```

# Assegnamenti multilinea

- Non é possibile andare a capo in un assegnamento a costante stringa

```
string sbagliata= "stringa veramente troppo, //ERRORE!  
lunga ";
```

- ...cosí peró va bene...

```
string corretta= "stringa veramente troppo, \n "  
"lunga ";
```



si puó inserire il carattere ritorno carrello (o il tab)

# Output di stringhe

- Una stringa si può stampare inserendolo su uno stream di output

```
string saluto="ciao mondo!!";  
cout«saluto«endl;
```

- Attenzione a inserire gli spazi....

```
string saluto="ciao";  
cout«saluto«"mondo"«endl;
```

...stampa "ciaomondo"

# Input di stringhe

- Si può usare l'operatore di estrazione da stream

```
string saluto;
```

```
cin » saluto;
```

- la lettura ignora gli spazi iniziali e poi estrae tutti i caratteri fino ad uno spazio bianco
- la quantità di memoria allocata viene automaticamente adattata ....COMODO NO???
- la funzione `getline` fornisce un maggiore controllo (ad es. carattere terminatore della stringa)

# Metodi: lunghezza di una stringa

- Le stringhe sono normali classi C++ dotati di funzioni membro
- Il primo metodo che vediamo é per misurare la lunghezza

```
int i;  
string saluto="ciao";  
i = saluto.length();
```

...produce . . .  $i = 4$

# Test di stringa vuota

- la funzione membro `bool empty()` restituisce `true` se la stringa non contiene carattere e `false` altrimenti
- Esempio

```
string s1="";  
cin » s1;  
if (s1.empty())  
    cout«"Lettura fallita!"«endl;
```

# Concatenazione

- Le stringhe possono essere concatenate tramite l'operatore (+)

```
string saluto = "ciao";
```

```
string chi="mondo";
```

```
string saluti = saluto + ", "+chi+"!"
```

equivale a `string saluti = ``ciao, mondo!``;`

- uso della concatenazione molto flessibile

```
saluto = saluto+'\n';
```

# Confronti tra stringhe

- Confronti possibili tramite operatori ==, !=, >, <, <=, >= (gli ultimi 4 realizzano l'ordinamento lessicografico)
- Esempio

```
string s1, s3;
```

```
...
```

```
if (s1==s2)
```

```
    cout<<"Stringhe uguali!"<<endl;
```

```
...
```

```
while (s1 != s2)
```

```
    cin>>s1;
```

```
...
```

# Accesso agli elementi di una stringa

- L'accesso agli elementi di una stringa può avvenire tramite funzione membro `at(int i)` o tramite operatore `[]`.
- il primo carattere é alla posizione zero
- Esempio

```
string s="ingegneria";  
char c;  
c = s.at(2); //c = 'g'  
c= s[1]; //c='n'
```

# Inserimento di una stringa

- Una stringa può essere inserita in un'altra tramite il metodo

```
string & insert(int startpos, string s)
```

- Esempio

```
string s="Vittorio Secondo";
```

```
string s1 = "Emanuele";
```

```
s.insert(8, s1); //s diviene "Vittorio Emanuele Secondo"
```

- Un'altra versione del metodo (a quattro parametri) consente di inserire parte di una stringa in un'altra

# Estrazione di una sottostringa

- L'estrazione di una sottostringa viene fatta tramite il metodo

```
string & substr(int start, int num)
```

- Esempio

```
string s="Vittorio Secondo";  
string s1 = s.substr(8,7);  
cout«s1«endl; //stampa Secondo
```

# Cancellazione di una sottostringa

- La cancellazione di una sottostringa viene fatta tramite il metodo

```
string & erase(int start, int num)
```

- Esempio

```
string s="Vittorio Secondo";  
s.erase(8,7);  
cout<<s<<endl; //stampa Secondo
```

# Sostituzione di una sottostringa

- Si può cancellare e sostituire una sottostringa all'interno di una stringa tramite il metodo `string & replace(int start, int num, string s)`

- Esempio

```
string s="Vittorio Secondo";  
s.replace(0,8,"Umberto");  
cout«s«endl; //stampa "Umberto Secondo"
```

# Ricerca di una sottostringa

- Si può cercare una sottostringa all'interno di una stringa tramite il metodo

```
int find(string s, int startSearch)
```

- Esempio

```
string s="Vittorio Secondo";  
a=s.find("S",0);  
cout«a«endl; //stampa 9
```