Alessio Bechini
- Corso di Fondamenti di Informatica II -

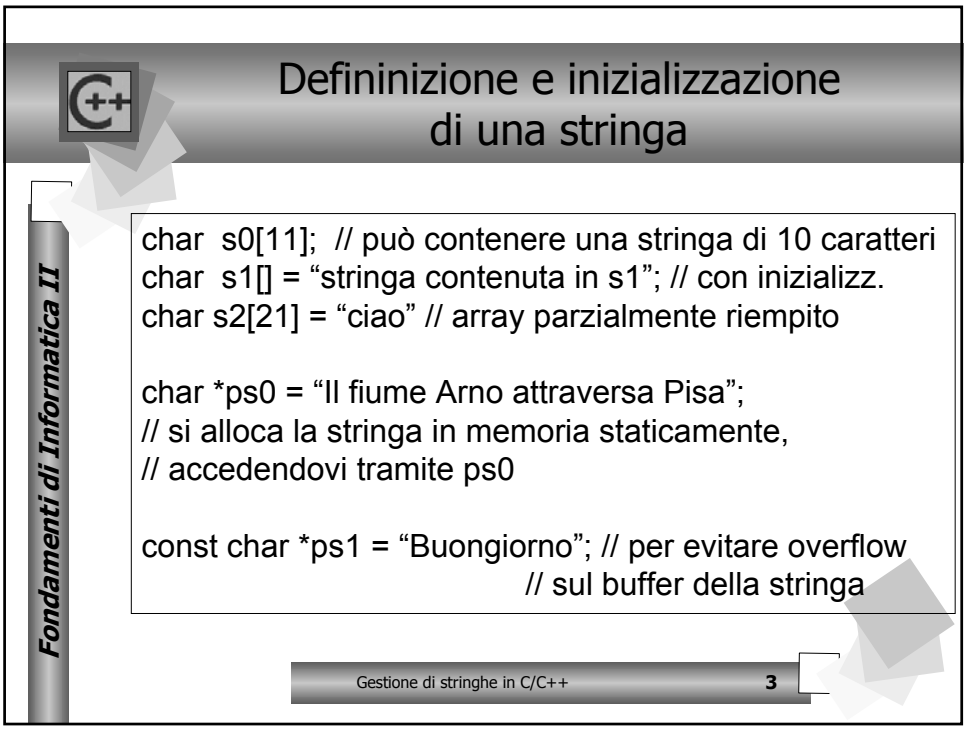
Gestione di stringhe

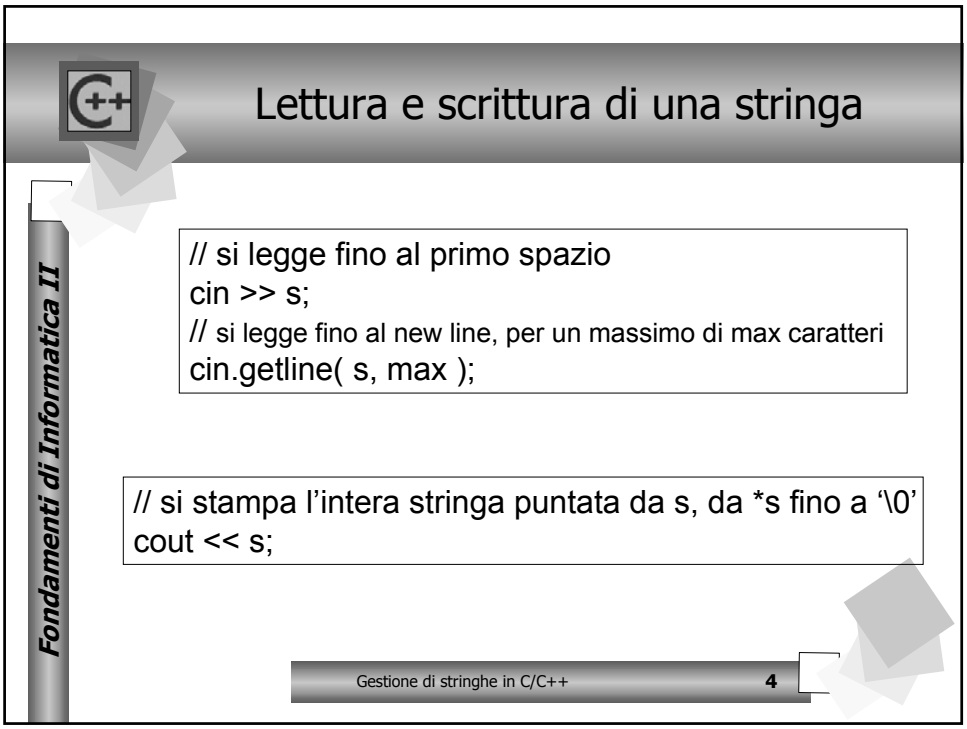
Parte del materiale proposto
è stato gentilmente fornito da G. Lipari

Le stringhe in C/C++

Fondamenti di Informatica II

- In C e in C++ non esiste un vero e proprio tipo stringa.
- Una stringa è vista come una sequenza di caratteri terminata da '\0', contenuta in uno spazio in memoria (buffer, spesso implementato come array).
- La gestione delle stringhe così concepite è facilitata da un insieme di funzioni in <string.h>, <stdlib.h>
- Le librerie standard del C++ forniscono la classe *string*, che solleva il programmatore dalle problematiche di allocazione dei buffer.


Definizione e inizializzazione di una stringa
Fondamenti di Informatica II
char s0[11]; // può contenere 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 in memoria staticamente,
// accedendovi tramite ps0
const char *ps1 = “Buongiorno”; // per evitare overflow
// sul buffer della stringa
Gestione di stringhe in C/C++ 3


Lettura e scrittura di una stringa
Fondamenti di Informatica II
// si legge fino al primo spazio
cin >> s;
// si legge fino al new line, per un massimo di max caratteri
cin.getline(s, max);
// si stampa l’intera stringa puntata da s, da *s fino a ‘\0’
cout << s;
Gestione di stringhe in C/C++ 4

Assegnazione tra stringhe

Fondamenti di Informatica II

- L'assegnazione tra stringhe non può essere fatta semplicemente usando l'operatore = (consentita solo per l'inizializzazione).
- Si utilizza invece una funzione che provvede a copiare i caratteri ad uno ad uno.
- Si possono avere due versioni della funzione **strcpy**:

```
void   strcpy(char *s, const char *t)
char *strcpy(char *s, const char *t)
```

Gestione di stringhe in C/C++ 5

Due implementazioni di *strcpy*

Fondamenti di Informatica II

```
char  *strcpy(char *s, const char *t)  {
    char *p;
    for(p=s; (*p++ = *t++) != '\0'; );
    return s;
}
```

Permette concatenazione

```
void strcpy(char *s, const char *t)  {
    while(*s++ = *t++);
}
```

Gestione di stringhe in C/C++ 6

Altre operazioni su stringhe

Fondamenti di Informatica II

- Concatenazione: *char *strcat(char *s, const char *t)*
- Determinazione di lunghezza:
  *int strlen(char *s)* oppure anche attraverso *sizeof*
- Comparazione:
  *int strcmp(const char *s, const char *t)*
  // 0 se uguali, <0 se s<t, >0 se s>t
- Ricerca di carattere in stringa:
  char *strchr(const char *s, const char c)
- Ricerca di sottostringa in stringa:
  *char *strchr(const char *s, const char *t)*

La classe *string*

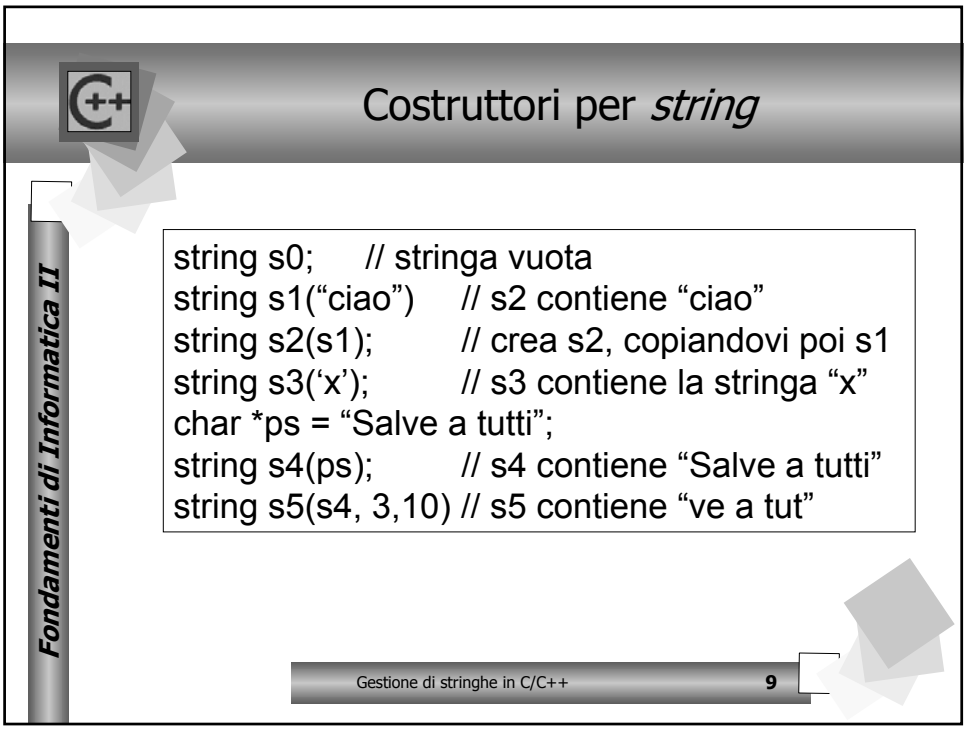
Fondamenti di Informatica II

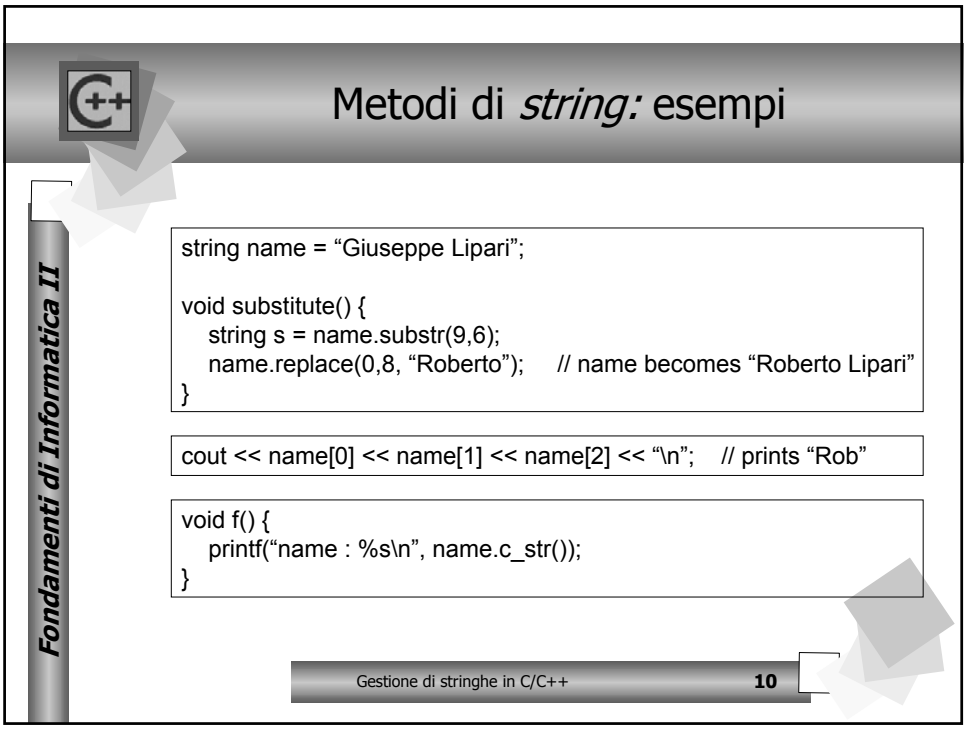
- Le librerie standard forniscono il tipo *string;*
  il programmatore può disinteressarsi
  del problema della lunghezza delle stringhe.

```
#include <string>
using namespace std;
int main() {
   string s1 = "Hello";
   string s2 = "world";

   string s3 = s1 + " " + s2;

   cout << s3 << "\n";
}
```

```
void respond(const string &answer) {
    if (answer = = "yes") {...}
    else if (answer = = "no") {...}
    else cout << "Please answer y/n\n";
}
```

- La classe *string* è stata scritta
  per la piena compatibilità
  con le stringhe C "classiche"


Costruttori per *string*
Fondamenti di Informatica II
string s0; // stringa vuota
string s1("ciao") // s2 contiene "ciao"
string s2(s1); // crea s2, copiandovi poi s1
string s3('x'); // s3 contiene la stringa "x"
char *ps = "Salve a tutti";
string s4(ps); // s4 contiene "Salve a tutti"
string s5(s4, 3,10) // s5 contiene "ve a tut"
Gestione di stringhe in C/C++ 9


Metodi di *string:* esempi
Fondamenti di Informatica II
string name = "Giuseppe Lipari";
void substitute() {
string s = name.substr(9,6);
name.replace(0,8, "Roberto"); // name becomes "Roberto Lipari"
}
cout << name[0] << name[1] << name[2] << "\n"; // prints "Rob"
void f() {
printf("name : %s\n", name.c_str());
}
Gestione di stringhe in C/C++ 10

Comparazione di oggetti *string*

- Le istanze di *string* possono essere comparate con gli operatori standard;
- L'ordinamento standard tra stringhe è quello alfabetico

```
string a = "Peppe";
string b = "Gianni";
string c = "Gianni";

void cmp(const string &s1, const string &s2) {
    cout << s1;
    if (s1 = = s2) cout << " = = ";
    else if (s1 < s2) cout << " < ";
    else cout << " > ";
    cout << s2 << "\n";
}

cmp(a,b);       // prints   "Peppe > Gianni"
cmp(b,c);       // prints   "Gianni == Gianni"
cmp(c,a);       // prints   "Gianni < Peppe"
```


Input/Output con oggetti *string*

- Come leggere una parola

```
int main () {
    string str;
    cout << "please, enter your name ";
    cin >> str;
    cout << "Hello " << str << "!\n";
}
```

- Come leggere/scrivere un'intera linea

```
int main () {
    string str;
    cout << "please, enter your name ";
    getline(cin, str);
    cout << "Hello " << str << "!\n";
}
```


Altri metodi di *string*
Fondamenti di Informatica II
La classe string fornisce metodi per tutte le operazioni più comunemente usate con le stringhe:
• Inserimento di sottostringa (insert)
• Sostituzione/rimozione di caratteri (replace, remove)
• Accesso a una sottostringa (substr)
• Comparazione (compare)
• Ricerca di sottostringhe (find)
• ...e molte altre ancora (v. libreria)
Gestione di stringhe in C/C++ 13