Università della Calabria

Corso di Laurea in Ingegneria Elettronica

Programmazione Orientata agli Oggetti

A.A. 2006/2007

Stringhe

Paolo Trunfio *

* DEIS, Università della Calabria – http://si.deis.unical.it/~trunfio

Inizializzazione e assegnamento

La stringa può essere inizializzata contestualmente alla dichiarazione utilizzando le parentesi tonde, oppure successivamente utilizzando l'operatore di assegnamento **=**.

```
string a("Stringa1");
cout << a << endl;  // Stringa1

string b;
cout << b.size() << endl;  // 0
b = "Stringa2";
cout << b << endl;  // Stringa2

string c = "Stringa3";
cout << c << endl;  // Stringa3

string *d = new string("Stringa4");
cout << *d << endl;  // Stringa4
```

Confronti fra stringhe

Due stringhe possono essere confrontate mediante gli operatori relazionali **==**, **!=**, **>**, **<**, **>=**, **<=**.

Esempi:

```
string s1 = "Introduzione";
string s2 = "Conclusioni";
string s3 = s1;

bool a = s1 != s2;  // true
bool b = s1 < s2;   // false
bool c = s1 >= "Indice";  // true
bool d = s1 == s2;  // false
bool e = s1 == s3;  // true
```


Principali funzioni

Le principali funzioni della classe *string* sono le seguenti:

```
size_type find(const string &s, size_type pos = 0)
string& insert(size_type pos, const string &s)
string substr(size_type pos = 0, size_type num = npos)
string& erase(size_type pos = 0, size_type num = npos)
string& replace(size_type pos, size_type num, const string &s)
```

Il tipo *size_type* (che generalmente corrisponde ad *unsigned long*) è utilizzato per gli interi ricevuti o restituiti dalle funzioni della classe *string*.

La costante *npos* di tipo *size_type*, definita nella classe *string*, rappresenta la massima dimensione allocabile per una stringa. Il valore della costante *string::npos* è uguale al valore massimo del tipo *size_type*.

Ricerca di una sottostringa

size_type find(const string &s, size_type pos = 0)

Cerca, a partire dalla posizione *pos*, la posizione della stringa *s* nella stringa corrente. Il valore di pos non deve essere maggiore della lunghezza di *s*. Se *s* è presente ne restituisce la posizione, altrimenti restituisce *npos.*

Esempio:

```
string s1 = "Programmazione Orientata agli Oggetti";
string s2 = "Orientata";
unsigned long pos = s1.find(s2);  // ricerca effettuata dalla posizione 0
if (pos == string::npos)
  cout << "Stringa non presente" << endl;
else
  cout << "Stringa in posizione " << pos << endl;  // Stringa in posizione 15
```


Inserimento di una stringa

string& insert(size_type pos, const string &s)

Inserisce la stringa *s* nella stringa corrente a partire dalla posizione *pos*. Restituisce un riferimento alla stringa corrente.

Esempio:

```
string s1 = "Programmazione Oggetti";

string s2 = "Orientata agli ";

s1.insert(15, s2);

cout << s1 << endl;  // Programmazione Orientata agli Oggetti
```

Estrazione di una sottostringa

string substr(size_type pos = 0, size_type num = npos)

Restituisce una nuova stringa contenente i caratteri della stringa corrente nell'intervallo [*pos*, *pos+num*[.

Esempio:

```
string s1 = "Programmazione Orientata agli Oggetti";

string s2 = s1.substr(15, 9);

cout << s2 << endl;  // Orientata

string s3 = s1.substr(30, 50);  // non esistono 50 caratteri dalla posizione 30

cout << s3 << endl;  // Oggetti
```


Cancellazione di una sottostringa

string& erase(size_type pos = 0, size_type num = npos)

Cancella dalla stringa corrente i caratteri nell'intervallo [*pos*, *pos+num*[. Restituisce un riferimento alla stringa corrente.

Esempio:

```
string s = "Programmazione Orientata agli Oggetti";

s.erase(15, 10);

cout << s << endl;  // Programmazione agli Oggetti
```

Sostituzione di una sottostringa

string& replace(size_type pos, size_type num, const string &s)

Rimuove dalla stringa corrente i caratteri nell'intervallo [*pos*, *pos+num*[e li sostituisce con la stringa *s*. Restituisce un riferimento alla stringa corrente.

Esempio:

```
string s1 = "Programmazione Orientata agli Oggetti";

string s2 = "ad";

s1.replace(15, 14, s2);

cout << s1 << endl;  // Programmazione ad Oggetti
```


Stringhe in C

In C le stringhe sono implementate come un array di caratteri:

```
char s[] = "ING";
// char s[4] = "ING";
```

L'array così inizializzato viene automaticamente terminato dal carattere **'\0'**:

s[0] == 'I' s[1] == 'N' s[2] == 'G' s[3] == '\0'

Generalmente le stringhe C sono gestite mediante i puntatori:

```
char *s;  // stringa definita come puntatore a caratteri
s = new char[4];
s[0] = 'P'; s[1] = 'O'; s[2] = 'O'; s[3] = '\0';
cout << s << endl;  // POO
```

Stringhe in C

Le stringhe standard di C sono utilizzate anche nei programmi C++. Per convertire una stringa C++ nella corrispondente stringa C è possibile utilizzare la funzione **c_str**:

```
string a = "C++";
const char *b = a.c_str();  // restituisce una stringa costante
cout << b << endl;  // C++
```

Le stringhe C possono essere convertite in oggetti *string* di C++ in diversi modi, ad esempio mediante l'operatore di assegnamento:

```
char a[] = "C++";
string b = a;
cout << b << endl;  // C++
```