PROGRAMMIAMO

C++ - Il contenitore string

Classi predefinite standard: la classe string

Così come le funzioni di library standard del C++ mettonoa disposizione del programmatore codice già scritto per risolvere alcuni problemi comuni, allo stesso modo la library standard contiene alcune classi (dette in questo contesto più propriamente **contenitori**) predefinite. Un esempio semplice e particolarmente interessante è la classe string per la gestione delle stringhe.

Abbiamo visto in precedenza come in C le stringhe possano essere trattate come vettori di char. Tuttavia il C++ semplifica grandemente l'uso delle stringhe per mezzo di una classe predefinita di library, la **classe string** appunto. Si osservi il seguente programma:

```
#include <iostream>

using namespace std;

#include <string>

int main()
{
string parola;

cout<<"Fornisci una parola: ";
cin>>parola;
cout<<"La parola inserita è "<<parola<<"\n";

return 0;
}
```

Si osservi anzitutto la dichiarazione

```
#include <string>
```

posta all'inizio del programma e necessaria per poter utilizzare la classe string. A questo punto si dichiara un oggetto appartenente alla classe con la sintassi

string *nome_oggetto*;

Si noti che non è necessario dichiarare una dimensione massima per l'oggetto string. Le stringhe in C++ vengono gestite dinamicamente, riservando in modo automatico la memoria necessaria.

Acquisizione di una stringa contenente spazi (blank)

L'istruzione cin, usata nell'esempio precedente per acquisire una stringa, presenta il problema che non acquisisce stringhe contenenti al proprio interno degli spazi. Per esempio volendo acquisire la stringa

"nel mezzo del cammin di nostra vita"

con

cin>>verso;

alla fine verso conterrebbe solo "nel" poiché l'acquisizione termina automaticamente in corrispondenza del primo spazio.

Il problema può essere risolto in C++ usando la funzione getline nel seguente modo:

getline(cin, verso);

La sintassi è abbastanza semplice:

getline(cin, *nome_string_da_acquisire*)

E' importante non confondere la funzione getline con la cin.getline usata per acquisire stringhe rappresentate come vettori di char (cioè nel modo "tradizionale" del linguaggio C).

Costruttori

La dichiarazione permette anche di inizializzare l'oggetto sfruttando il costruttore della classe nel seguente modo:

string parola("valore iniziale");

oppure anche così:

string parola = "valore iniziale";

Operatori

La classe string ridefinisce tutta una serie di operatori standard del C (overloading) in modo che siano applicabili a oggetti di tipo string. Gli operatori principali della classe sono:

- **Assegnazione** (=)
  E' possibile assegnare un valore a una stringa mediante l'uguale. Qui sotto vengono mostrati alcuni esempi:

```
string nome1, nome2;

nome1 = "pippo";
nome1 = nome2;
```

- **Concatenazione** (+)
  Concatenare due stringhe vuol dire creare una terza stringa formata dall'unione delle prime due:

```
string string1 = "uno due ";
string string2 = " tre quattro";

string string3 = string1 + string2;

// Visualizza "uno due tre quattro"
cout<<string3<<"\n";
```

Si noti l'inizializzazione del valore delle stringhe contestualmente alla loro dichiarazione.

- **Operatori di confronto** (==, !=, <, <=, >, >=)
  Gli esempi seguenti dovrebbero essere sufficienti per illustrare l'uso degli operatori di confronto con le stringhe:

```
string parola1, parola2;

cout<<"Fornisci la prima parola: ";
cin>>parola1;
cout<<"Fornisci la seconda parola: ";
cin>>parola2;

if (parola1==parola2)
cout<<parola1<<" è uguale a "<<parola2<<"\n";

if (parola1!=parola2)
cout<<parola1<<" è diversa da "<<parola2<<"\n";

if (parola1<parola2)
cout<<parola1<<" precede "<<parola2<<" nell'ordine alfabetico\n";

if (parola1>parola2)
cout<<parola1<<" segue "<<parola2<<" nell'ordine alfabetico\n";
```

Principali metodi

I principali metodi della class string sono i seguenti:

- **Size e length**
  Il metodo *size* consente di determinare la lunghezza (numero di caratteri) di una stringa nel seguente modo (il numero di caratteri nell'esempio è 6):

```
string parola = "Torino";
cout<<"La stringa "<<parola<< " contiene "<<parola.size()<<" caratteri\n";
```

  Il metodo length funziona allo stesso modo e viene spesso usato in alternativa a size.

- **Substr**
  Il metodo *substr* restituisce una stringa ottenuta estraendo dalla stringa data il numero di caratteri specificato a partire dalla posizione indicata (0 indica il primo carattere della stringa). Il primo argomento tra parentesi indica la posizione da cui deve iniziare l'estrazione dei caratteri; il secondo argomento è il numero di caratteri da estrarre. L'esempio seguente dovrebbe chiarire l'applicazione del metodo:

```
string parola = "Torino";

// visualizza "Tori"
cout<<parola.substr(0,4)<<"\n";

// visualizza "rino"
cout<<parola.substr(2,4)<<"\n";
```

  Un operatore simile a size, che restituisce sempre il numero di caratteri della stringa è *length* (uso:

parola.length()).

- **Replace**
  Il metodo *replace* sostituisce una sottostringa all'interno di una stringa con un'altra sottostringa (che può anche avere un numero diverso di caratteri). Il primo argomento tra parentesi indica la posizione da cui deve iniziare l'estrazione dei caratteri; il secondo argomento è il numero di caratteri da sostituire nella stringa di partenza; il terzo argomento è la sottostringa da sostituire. Esempio:

```
string parola = "Torino";

// visualizza "Milano"
cout<<parola.replace(0,4,"Mila")<<"\n";

// visualizza "Merano"
cout<<parola.replace(1,2,"er")<<"\n";

// visualizza "Loano"
cout<<parola.replace(0,3,"Lo")<<"\n";

// visualizza "Loreto"
cout<<parola.replace(2,3,"reto")<<"\n";
```

- **Accesso ai singoli caratteri**
  E' possibile accedere ai singoli caratteri che compongono una stringa come se fossero gli elementi di un vettore (in modo analogo a quanto avviene in C). Esempio:

```
string str ("stringa di prova");

for (int i=0; i<str.size(); ++i)
  cout << str[i];
```

  Il metodo length fornisce la lunghezza della stringa (numero di caratteri). Si presti attenzione al fatto che, sebbene sia possibile in C++ accedere ai caratteri di una stringa come se fossero gli elementi di un vettore di char, ciò non significa che un oggetto di tipo string *sia* un vettore di char (vedi qui sotto la spiegazione dell'uso di c_str).

  Una possibilità alternartiva consiste nell'usare il metodo at, nel seguente modo:

```
string str ("stringa di prova");

for (int i=0; i<str.size(); ++i)
  cout << str.at(i);
```

  In entrambi i casi il risultato è lo stesso.

- **Conversione della stringa nello stile C (cioè come vettore di char)**
  In alcuni casi occorre convertire un oggetto di tipo string in un vettore di caratteri, cioè nella modalità con cui le stringhe vengono definite nel C tradizionale. Questo è utile in particolare quando è necessario passare una stringa a una funzione che richiede un vettore di char. In questi casi occorre usare la member function *c_str()* nel seguente modo:

```
string stringa ("stringa di prova");
cout<<strlen(stringa.c_str());
```

Per una spiegazione degli altri numerosi metodi e operatori della classe string rimandiamo a cplusplus.com e a Cprogramming.com.

precedente - successiva