Gli array di caratteri e le stringhe

Docente
Mario Perna
prof.perna.mario@darzo.net

A.S.
2025/2026
Materia
Informatica

Introduzione

Una stringa è una **sequenza di caratteri**.

In generale abbiamo due modi per gestire questo tipo di informazione:

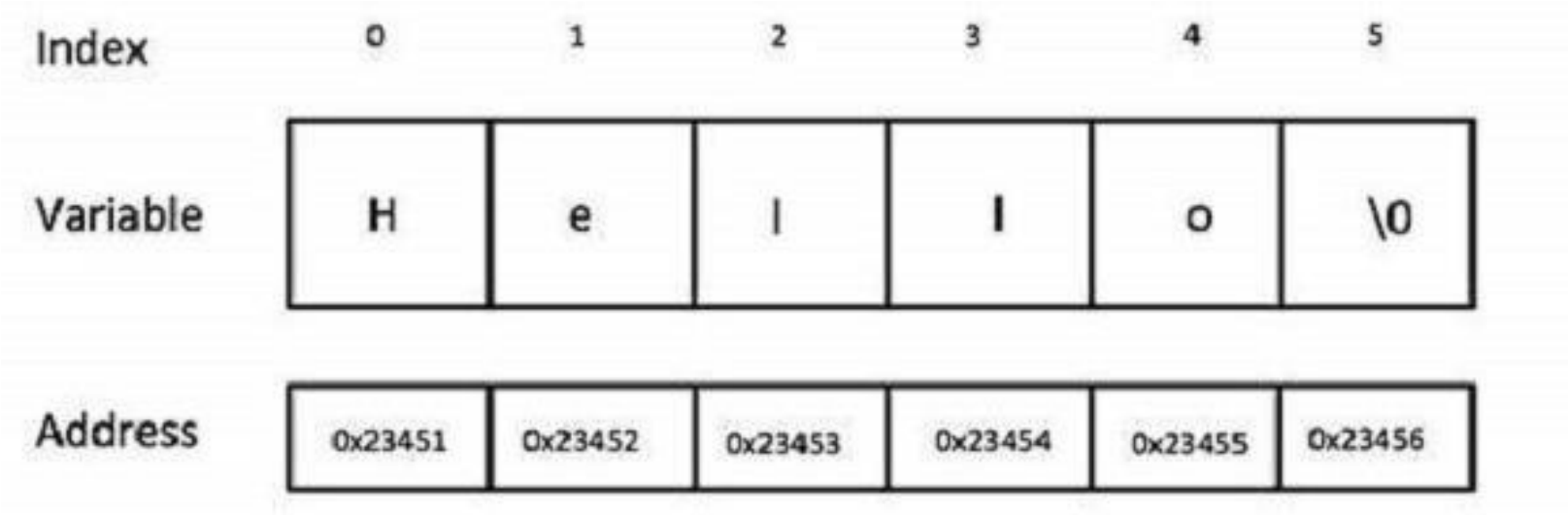
Come array di caratteri (in C++ `char[]`)

Come tipo stringa (in C++ `string`)

Per ora ci limitiamo a vedere il primo approccio.

Array di char

Se consideriamo le stringhe come array di caratteri, la definizione di stringa diventa:
Una sequenza di caratteri che termina con il carattere speciale '\0'



Array di char

Alcuni esempi di stringa possono essere:

- "cane" rappresentato da un array di char di 5 elementi
- "ciao\n" rappresentato da un array di char di 6 elementi
- "" rappresentato da un array di char di 1 elemento

Perché proprio la lunghezza della stringa più uno? Perché **dobbiamo memorizzare il carattere delimitatore '\0'**

c	a	n	e	\0
0	1	2	3	4

char[5]

c	i	a	o	\n	\0
0	1	2	3	4	5

char[6]

Array di char

Se allochiamo più spazio per memorizzare la stringa otteniamo che le celle in eccedenza conterranno **valori indefiniti**.

c	a	n	e	\0
0	1	2	3	4

char[5]

valori indefiniti

c	a	n	e	\0	?	?
0	1	2	3	4	5	6

char[7]

Array di char: inizializzazione

Possiamo inizializzare un array di charin tre modi:

Inizializzazione come insieme:

char anime[10] = {'O', 'n', 'e', ' ', 'P', 'i', 'e', 'c', 'e', '\0'}

Inizializzazione diretta con lunghezza esplicita:

char anime[10] = "One Piece";

Inizializzazione diretta con lunghezza implicita:

char anime[] = "One Piece";

Il carattere terminatore '\0' viene inserito automaticamente

La lunghezza della stringa viene dedotta in automatico

Array di char: accesso ai singoli caratteri

Possiamo accedere ai singoli elementi dell'array di caratteri come un normale array che abbiamo già studiato:

```
1  #include <iostream>
2  using namespace std;
3
4  int main(){
5
6      char stringa[6];
7
8      stringa[0] = 'G';
9      stringa[1] = 'a';
10     stringa[2] = 't';
11     stringa[3] = 't';
12     stringa[4] = 'o';
13     stringa[5] = '\0';
14
15     stringa[0] = 'M';
16
17 }
```

Notate come per gli array di tipo `char` è possibile stampare direttamente, senza iterare elemento per elemento

Anche la stampa per singolo elemento è possibile

Array di char: output

Possiamo mostrare in output un array di caratteri in vari modi:

```cpp
#include <iostream>
using namespace std;

int main(){

    const int LEN = 6;
    char stringa[LEN] = "Gatto";

    // 1 - Stampa diretta mediante cout
    cout << stringa << endl;

    // 2 - Stampa dei singoli caratteri
    // fino al carattere terminatore
    int i = 0;
    while (stringa[i] != '\0'){
        cout << stringa[i];
        i++;
    }
    cout << endl;

    // 3 - Stampa dei singoli caratteri
    // fino alla lunghezza della stringa
    i = 0;
    while (i < LEN){
        cout << stringa[i];
        i++;
    }
    cout << endl;
}
```

Notate come per gli array di tipo charè possibile stampare direttamente, senza iterare elemento per elemento

Anche la stampa per singolo elemento è possibile

Array di char: input

Possiamo leggere in input un array di caratteri in vari modi:

```cpp
#include <iostream>
using namespace std;

int main(){

    const int LEN = 5;
    char stringa[LEN];

    // 1 - Leggo fino a quando trovo uno spazio o andata a capo
    cin >> stringa;

    // 2 - Leggo fino a quando trovo una andata a capo o raggiungo la lunghezza massima
    cin.getline(stringa, LEN);

    // 3 - Leggo fino a quando trovo una andata a capo o il carattere '.' o raggiungo la lunghezza massima
    char carattere = cin.get();
    int i = 0;
    while (carattere != '\n' && carattere != '.' && i < LEN-1){
        stringa[i] = carattere;
        i++;
        carattere = cin.get();
    }
    stringa[i] = '\0';

}
```

Lettura stringa: semplice, ma poco flessibile

Lettura carattere per carattere: complessa, ma molto flessibile

Array di char: funzioni

Sono disponibili quattro operazioni per manipolare gli array di caratteri nella libreria cstring:

- strlen(s): dato l'array di char sviene restituita la sua lunghezza
- strcpy(s1, s2): l'array di char s2viene copiato in s1
- strcat(s1, s2): l'array di char s2viene concatenato ad s1
- strcmp(s1, s2):
  - restituisce -1 se s1è lessicograficamente precedente a s2
  - restituisce 0 se s1è uguale a s2
  - restituisce 1 se s1è lessicograficamente successivo a s2

ATTENZIONE: quando si esegue la `strcpy` o `strcat`, la stringa di destinazione deve essere sufficientemente capiente. Ad esempio non posso copiare in una stringa di 5 caratteri un'altra da 6 o più caratteri!

Array di char: funzione strlen

```cpp
#include <iostream>
using namespace std;
#include <cstring>

int main(){

    char s1[] = "Mango";
    char s2[] = "Barbabietola da zucchero";
    char s3[] = "";

    cout << "lunghezza s1: " << strlen(s1) << endl;
    cout << "lunghezza s2: " << strlen(s2) << endl;
    cout << "lunghezza s3: " << strlen(s3) << endl;

}
```

```
lunghezza s1: 5
lunghezza s2: 24
lunghezza s3: 0
```

Array di char: funzione strcpy

```cpp
#include <iostream>
using namespace std;
#include <cstring>

int main(){

    // Notare la dimensione impostata a 50
    // Necessario per avere capienza sufficiente quando invoco la strcpy
    char s1[50] = "Mango";
    char s2[50] = "Barbabietola da zucchero";
    char s3[50] = "";

    strcpy(s3, s1);

    cout << "s1: " << s1 << endl;
    cout << "s2: " << s2 << endl;
    cout << "s3: " << s3 << endl;

    strcpy(s1, s2);

    cout << "s1: " << s1 << endl;
    cout << "s2: " << s2 << endl;
    cout << "s3: " << s3 << endl;



}
```

```
s1: Mango
s2: Barbabietola da zucchero
s3: Mango
s1: Barbabietola da zucchero
s2: Barbabietola da zucchero
s3: Mango
```

Array di char: funzione strcat

```cpp
#include <iostream>
using namespace std;
#include <cstring>

int main(){

    // Notare la dimensione impostata a 50
    // Necessario per avere capienza sufficiente quando invoco la strcat
    char s1[100] = "Coccodrillo";
    char s2[100] = "Volante";
    char s3[100] = "Verde";

    strcat(s3, s1);

    cout << "s1: " << s1 << endl;
    cout << "s2: " << s2 << endl;
    cout << "s3: " << s3 << endl;

    strcat(s1, s2);

    cout << "s1: " << s1 << endl;
    cout << "s2: " << s2 << endl;
    cout << "s3: " << s3 << endl;


}
```

```
s1: Coccodrillo
s2: Volante
s3: VerdeCoccodrillo
s1: CoccodrilloVolante
s2: Volante
s3: VerdeCoccodrillo
```

Array di char: funzione strcmp

```cpp
#include <iostream>
using namespace std;
#include <cstring>

int main(){

    char s1[] = "Limone";
    char s2[] = "Tamburo";
    char s3[] = "Limo";

    cout << "strcmp(s1,s2): " << strcmp(s1,s2) << endl;
    cout << "strcmp(s1,s3): " << strcmp(s1,s3) << endl;
    cout << "strcmp(s2,s3): " << strcmp(s2,s3) << endl;
    cout << "strcmp(s2,s1): " << strcmp(s2,s1) << endl;
    cout << "strcmp(s3,s1): " << strcmp(s3,s1) << endl;
    cout << "strcmp(s3,s2): " << strcmp(s3,s2) << endl;

    char s4[] = "Limone";
    cout << "strcmp(s1,s4): " << strcmp(s1,s4) << endl;
    cout << "strcmp(s4,s1): " << strcmp(s4,s1) << endl;


}

```

```
strcmp(s1,s2): -1
strcmp(s1,s3): 1
strcmp(s2,s3): 1
strcmp(s2,s1): 1
strcmp(s3,s1): -1
strcmp(s3,s2): -1
strcmp(s1,s4): 0
strcmp(s4,s1): 0
```

Le stringhe

Una stringa è una **sequenza di caratteri**.

In generale abbiamo due modi per gestire questo tipo di informazione:

Come array di caratteri (in C++ `char[]`)

Come tipo stringa (in C++ `string`)

Ora che abbiamo studiato gli array di caratteri (cstring), analizziamo il secondo approccio.

Le stringhe

La libreria standard ci fornisce la **classe** stringche implementa al suo interno un array di caratteri nascondendo al programmatore tutte le problematiche di gestione dell'array.

Inoltre, fornisce al programmatore tutta una serie di funzioni (**metodi**) per manipolare la stringa.

```
La lunghezza della stringa Pesca e' 5
La lunghezza della stringa Arancia e' 7
```

```cpp
#include <iostream>
using namespace std;
#include <string>

int main(){
    string frutto;

    frutto = "Pesca";
    cout << "La lunghezza della stringa " << frutto << " e' " << frutto.length() << endl;

    frutto = "Arancia";
    cout << "La lunghezza della stringa " << frutto << " e' " << frutto.length() << endl;
}
```

Posso assegnare a piacere senza preoccuparmi della dimensione massima. Con le cstringusavamo strcpy.

Metodo length
Con le cstring usavamo strlen.

Le stringhe: accesso ai singoli caratteri

Come per l'array di caratteri, anche le stringhe possono essere trattati come degli array a tutti gli effetti garantendo un **accesso diretto**:

```cpp
#include <iostream>
using namespace std;
#include <string>

int main(){
    string frutto;
    frutto = "Pesca";

    cout << "Le lettere di " << frutto << " sono:" << endl;
    for (int i = 0; i < frutto.length(); i++)
        cout << frutto[i] << endl;

    frutto[1] = 'o';
    cout << frutto;
}
```

```
Le lettere di Pesca sono:
P
e
s
c
a
Posca
```

In alternativa esiste il metodo s.at(i) che si comporta come s[i]

Le stringhe: concatenazione

```cpp
#include <iostream>
using namespace std;
#include <string>

int main(){
    string s1 = "Ciao ";
    string s2 = "a tutti";

    // Posso concatenare le stringhe con l'operatore +
    cout << s1 + s2 << endl;

    cout << s1 + s2 + " ragazze e ragazzi!" << endl;
}
```

```
Ciao a tutti
Ciao a tutti ragazze e ragazzi!
```

Con le cstringusavamo strcat

Le stringhe: confronto

```cpp
#include <iostream>
using namespace std;
#include <string>

int main(){
    string anime1 = "Naruto";
    string anime2 = "One Piece";

    cout << "Anime 1: " << anime1 << endl;
    cout << "Anime 2: " << anime2 << endl;

    // Confronto con operatore > < ==
    if (anime1 < anime2)
        cout << "La stringa " << anime1 << " precede lessicograficamente la stringa " << anime2 << endl;
    if (anime1 > anime2)
        cout << "La stringa " << anime1 << " segue lessicograficamente la stringa " << anime2 << endl;
    if (anime1 == anime2)
        cout << "La stringa " << anime1 << " è uguale alla stringa " << anime2 << endl;


    // Confronto con metodo compare

    cout << "anime1.compare(anime2): " << anime1.compare(anime2) << endl;
    cout << "anime2.compare(anime1): " << anime2.compare(anime1) << endl;


}
```

```
Anime 1: Naruto
Anime 2: One Piece
La stringa Naruto precede lessicograficamente la stringa One Piece
anime1.compare(anime2): -1
anime2.compare(anime1): 1
```

Con le cstringusavamo strcmp

Le stringhe: metodi extra

La potenza delle stringhe come classe è dovuta anche alla quantità di metodi a nostra disposizione come:

- s1.insert(pos, s2): aggiunge alla stringa s1dalla posizione posla stringa s2
- s1.erase(pos, num): rimuove dalla stringa s1dalla posizione posper numcaratteri
- s_dest = s1.substr(pos, num): restituisce in s_dest la sottostringa di s1dalla posizione posper numcaratteri
- s1.replace(pos, num, s2): rimpiazza nella stringa s1dalla posizione pos, num caratteri con la stringa s2
- pos_dest = s1.find(s2, pos): cerca nella stringa s1dalla posizione posla sottostringa s2e restituisci il risultato in pos_dest(restituisce -1 se s2non è trovato)

Le stringhe: metodo insert

```cpp
#include <iostream>
using namespace std;
#include <string>

int main(){
    string s1 = "Robin";

    s1.insert(s1.length(), " Hood");

    cout << s1 << endl;

    s1.insert(0, "Le avventure di ");

    cout << s1 << endl;

}
```

Robin Hood

Le avventure di Robin Hood

Le stringhe: metodo erase

```cpp
#include <iostream>
using namespace std;
#include <string>

int main(){
    string s1 = "Barbagianni";

    s1.erase(5, 6);

    cout << s1 << endl;

    s1.erase(3, 2);

    cout << s1 << endl;

}
```

Barba

Bar

Le stringhe: metodo substr

```cpp
#include <iostream>
using namespace std;
#include <string>

int main(){
    string s1 = "Pannocchia";


    cout << s1.substr(0,5) << endl;


    cout << s1.substr(4,5) << endl;

}
```

Panno

occhi

Le stringhe: metodo replace

```cpp
#include <iostream>
using namespace std;
#include <string>

int main(){
    string s1 = "Serpente";
    s1.replace(5, 0, "verde");
    cout << s1 << endl;

    s1 = "Serpente";
    s1.replace(5, 1, "verde");
    cout << s1 << endl;

    s1 = "Serpente";
    s1.replace(5, 2, "verde");
    cout << s1 << endl;

    s1 = "Serpente";
    s1.replace(5, 3, "verde");
    cout << s1 << endl;

    s1 = "Serpente";
    s1.replace(5, 4, "verde");
    cout << s1 << endl;


}
```

Serpeverdente

Serpeverdete

Serpeverdee

Serpeverde

Serpeverde

Le stringhe: metodo find

```cpp
#include <iostream>
using namespace std;
#include <string>

int main(){
    string s1 = "Cioccolato";

    int pos = s1.find("c", 0);

    if (pos != -1)
        cout << pos << endl;
    else
        cout << "Non trovato" << endl;

    pos = s1.find("c", 5);

    if (pos != -1)
        cout << pos << endl;
    else
        cout << "Non trovato" << endl;


    pos = s1.find("cola", 0);

    if (pos != -1)
        cout << pos << endl;
    else
        cout << "Non trovato" << endl;


}
```

3

Non trovato

4

Un meme vale più di 1000 parole…

Riassumendo:cstring vs string

cstring o array di char	tipo string
#include <cstring>	#include <string>
Implementato come un array di char e in carico al programmatore la gestione del carattere terminatore	Classe che nasconde l'implementazione come un array di char
Presente in C e C++	Presente solo in C++
Memoria necessaria fissata dal programmatore e generalmente consuma meno memoria rispetto alle string	Memoria necessaria gestita in automatico e generalmente consuma più memoria rispetto alle cstring
In generale più veloce come tempi di accesso	In generale più lento come tempi di accesso
Poche funzioni predefinite. Molte a carico del programmatore.	Tantissime funzioni a disposizione del programmatore.

Informazioni sull’utilizzo dei materiali didattici

Queste slides si basano su materiali originariamente elaborati dal **Prof. Andrea Melioli**, opportunamente modificati e integrati secondo specifiche esigenze riguardanti la programmazione disciplinare.

L’autore autorizza al **prof. Mario Perna** l’utilizzo, la modifica, la pubblicazione e qualsiasi altra forma di operazione sui materiali a scopo didattico e formativo.

L’uso o la diffusione di questi materiali è **vietato** senza il preventivo contatto con il proprietario del documento, Prof. Mario Perna (prof.mario.perna@darzo.net).