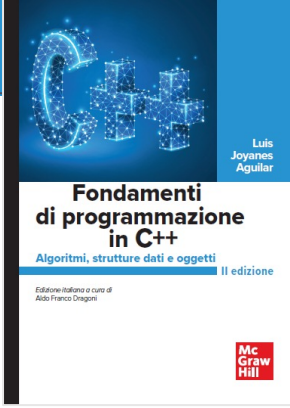
Capitolo 9

Stringhe

Pag. 236-253

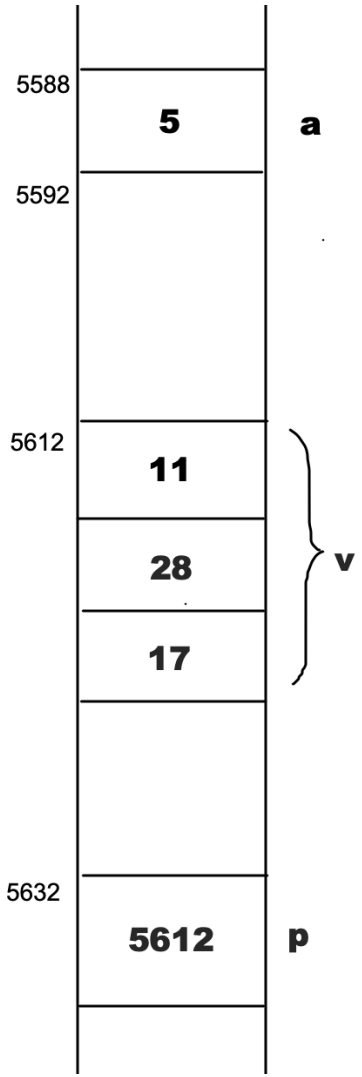
Presenta: Prof. Misael Mongiovì

Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.

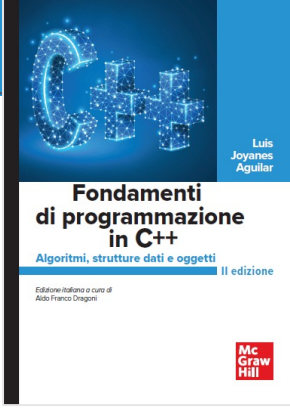


Vettori e puntatori

- *nome, tipo, indirizzo di una variabile*
  int a;
- *vettori*
  int v[3] = {11, 28, 17};
- *accesso agli elementi*
  v[0] elemento 0 di v
- *referenziazione e dereferenziazione*
  &a *ritorna l'indirizzo di a*
  *p *ritorna il valore all'indirizzo* p
- *su array*
  v *rappresenta il puntatore (indirizzo) al primo elemento*
  *v *rappresenta il valore* (v[0])



Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.
Mc Graw Hill

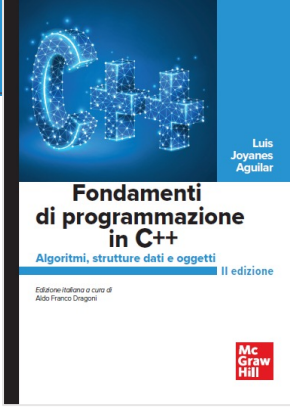


le stringhe

- *stringa letterale*
  sequenza di caratteri, ad esempio "ABC"
- *C-string*, o *stringa*
  - array di char che contiene il carattere *nullo* (`'\0'`)
  - la sequenza iniziale di caratteri dalla posizione 0 fino al carattere `\0` rappresenta una stringa
  - dereferenziando una stringa letterale ed utilizzando l'aritmetica dei puntatori:
- `*"ABC"` *è lo stesso che* `ABC[0]` *cioè* `'A'`
- `*("ABC" + 1)` *è lo stesso che* `ABC[1]` *cioè* `'B'`
- `*("ABC" + 2)` *è lo stesso che* `ABC[2]` *cioè* `'C'`

  `*("ABC" + 3)` *è lo stesso che* `ABC[3]` *cioè* `'\0'`

Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.
Mc Graw Hill



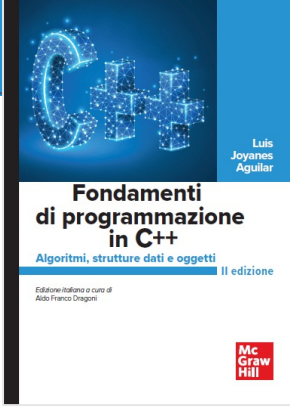
definizione di variabili stringa

- array di `char`, o `unsigned char` quando possono essere presenti caratteri speciali con il bit di ordine alto settato ad `1`

```
unsigned char dati[80]
```

- ci sono vari modi di inizializzare una stringa
  - `char UnaStringa[7]= {'A', 'B', 'C', 'D', 'E', 'F'};`
    `char UnaStringa[7]= "ABCDEF";`
  - `char testo[81] = "Questa è una stringa";`
  - `char testodemo[255] = "Questa è una stringa più lunga";`
  - `char stringatest[] = "Quale è la lunghezza di questa stringa?";`
- come i vettori, le stringhe non possono essere assegnate (lo si fa con la funzione di libreria denominata `strcpy()`)
  - `UnaStringa = "ABC"; // errore in fase di compilazione`

Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.
Mc Graw Hill



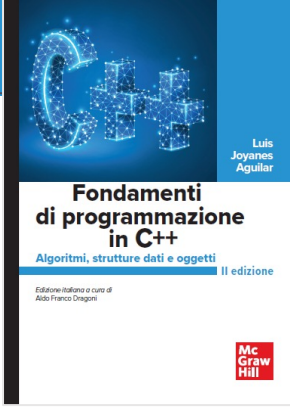
Input/Output di stringhe

- gli operatori di estrazione >> e di inserimento << con un vettore di caratteri si comportano in maniera speciale:

```
#include <iostream>
using namespace std;
void main()
{
 char nome[30]; // Definisce un array di caratteri
 cin >> nome;     // Legge caratteri e li mette dentro l'array
 cout << nome; // visualizza tutto il contenuto dell'array
}
```

- l'operatore >> termina la lettura quando trova uno spazio; per leggere una sequenza di caratteri che comprende spazi bianchi si deve utilizzare la funzione di libreria `getline()` invece che l'operatore >>.
- i segni d'interpunzione (apostrofi, virgole, punti, ecc.) fanno parte delle stringhe, mentre i caratteri di spaziatura (bianchi, tabulazioni, nuove righe, ecc.) no

Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.
Mc Graw Hill

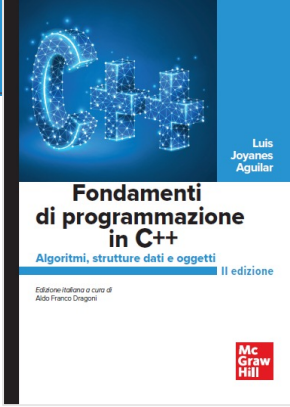


cin.get()

- `cin.get(car);`
- copia il carattere seguente del flusso di input `cin` nella variabile `car` e restituisce `1`, a meno che non riconosca la fine del flusso, caso in cui restituisce `0`

```
main()
{
    char car;
    int contatore = 0;
    while (cin.get(car))
        if (car == 't') ++contatore;
    cout << contatore << "lettere t\n";
}
```


Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.
Mc Graw Hill

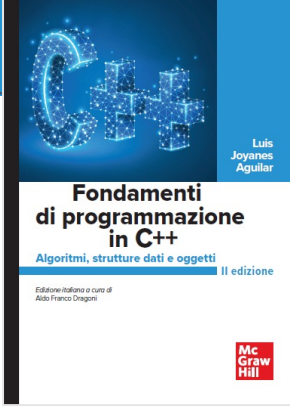


cin.getline()

- `cin.getline(var_str, max_lung_string+1, 'separatore');`
  - identificatore della variabile stringa
  - lunghezza massima della stringa
  - carattere successivo all'ultimo carattere della stringa (la funzione `getline()` inserisce automaticamente il carattere nullo come ultimo carattere della stringa).

- `cin.getline(str, n, car)` legge tutti gli inputs fino alla prima occorrenza del carattere `car` e li mette in `str`
  - se `car` è `'\n'`, la chiamata è equivalente a `cin.getline(str, n)`
  - `car` non viene inserito in `str` ma viene estratto dal flusso

Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.
Mc Graw Hill



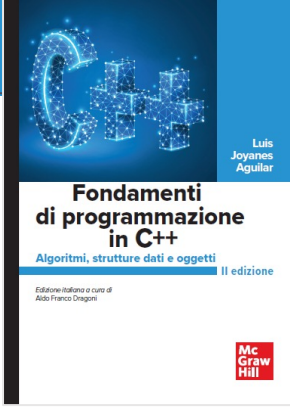
cin.getline()

```
#include <iostream>

using namespace std;

int main()
{
      char stringa[20];
      cout << "Please, enter your full
name: ";
      cin.getline(stringa, 9);
      cout << "Hello, " << stringa <<
"!\n";
      return 0;
}
```


Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.
Mc Graw Hill



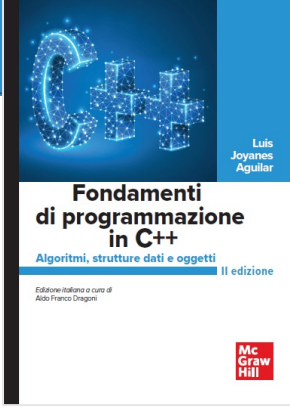
getline()

```
#include <iostream>

using namespace std;

int main()
{
   string name;
   cout << "Please, enter your full name:
";
   getline (cin, name);
   cout << "Hello, " << name << "!\n";
   return 0;
}
```


Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.

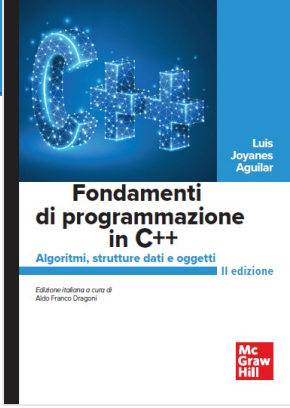


stringhe come argomenti a funzioni

- come tutti gli arrays, le stringhe possono essere passate alle funzioni fornendo il valore del puntatore al loro primo elemento

```
void sottostringa(char* dest, char* sorgente, int num_car)
{
    int contatore;
    for(contatore = 1; contatore <= num_car; contatore++)
        *dest++ = *sorgente++;
    *dest = '\0';
}
int main(int argc, char **argv)
{
    short lung;
    char s1[40], s2[40];
    cout << "Immetta una stringa \n";
    cin.getline(s1,42);
    cout << "Immetta lunghezza sottostringa \n";
    cin >> lung;
    sottostringa(&s2[0], &s1[0], lung);
    cout << s2 << endl;
    return 0;
}
```


Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.
Mc Graw Hill

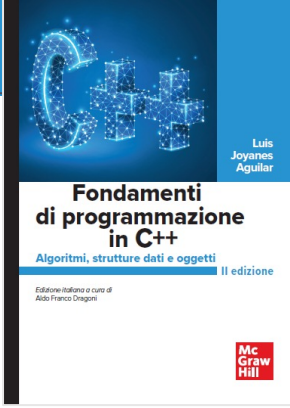


stringhe come argomenti a funzioni

- ... ovvero:

```
void sottostringa(char* dest, char* sorgente, int num_car)
{
   int contatore;
   for(contatore = 1; contatore <= num_car; contatore++)
      *dest++ = *sorgente++;
   *dest = '\0';
}
int main(int argc, char **argv)
{
   short lung;
   char s1[40], s2[40];
   cout << "Immetta una stringa \n";
   cin.getline(s1,42);
   cout << "Immetta lunghezza sottostringa \n";
   cin >> lung;
   sottostringa(s2, s1, lung);
   cout << s2 << endl;
   return 0;
}
```


Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.
Mc Graw Hill



passaggio stringhe per riferimento

- come tutti gli arrays, le stringhe possono essere passate alle funzioni anche per riferimento

```
typedef char str80[80];

int Lunghezza(str80& str)
{
    int contatore = 0;
    while (str[contatore++] != '\0');
    return contatore;
}
int main()
{
    str80 s = "C++ è meglio del C";
    cout << Lunghezza(s) << endl;
}
```


Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.
Mc Graw Hill

la libreria <cstring>

• **strcpy()**	• **char* strcpy(char* *destinazione*, const char* *sorgente*)** Copia la stringa *sorgente* nella stringa *destinazione*. Restituisce *destinazione*.
• **strncpy()**	• **char* strncpy(char* *destinazione*, const char* *sorgente*, size_t *num*)** Copia *num* caratteri da *sorgente* a *destinazione*. Restituisce *destinazione*
• **strlen()**	• **size_t strlen (const char* s)** Restituisce la lunghezza della stringa s.
• **strcat()**	• **char* strcat(char* destinazione, const char* sorgente)** Aggiunge una copia della stringa *sorgente* alla fine di *destinazione*. Restituisce *destinazione*.
• **strncat()**	• **char *strncat(char* s1, const char* s2, size_t n)** Aggiunge i primi *n* caratteri di s2 a s1. Restituisce s1. Se n >= strlen(s2), allora strncat(s1, s2, n) ha lo stesso effetto che strcat(s1, s2).
• **strcmp()**	• **int strcmp(const char* s1, const char* s2)** Confronta le stringa *s1* ed *s2* e restituisce: 0 se s1 = s2 <0 se s1 < s2 >0 se s1 > s2
• **stricmp()**	• **int stricmp(const char* s1, const char* s2)** come strcmp() ma *case insensitive*
• **strncmp()**	• **int strncmp(const char* s1, const char* s2, size_t n)** come strcmp() ma solo sui primi *n* caratteri.
• **strnicmp()**	• **int strncmp(const char* s1, const char* s2, size_t n)** come strncmp()ma *case insensitive*.
• **strrev()**	• **char* strrev(char* s)** inverte la stringa passata come argomento.
• **strupr()**	• **char* strupr(char* s)** mette in maiuscolo la stringa passata come argomento.
• **strlwr()**	• **char* strlwr(char* s)** mette in minuscolo la stringa passata come argomento.
• **strchr()**	• **const char* strchr(const char* str, int c)** Restituisce un puntatore alla prima occorrenza del carattere c in s. Restituisce NULL se c non è in s.

Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.
Mc Graw Hill

la libreria <cstring>

• **strspn()**	• **size_t strspn(const char* s1, const char* s2)** • Restituisce la lunghezza della sottostringa più lunga di s1 che comincia con s1[0] e contiene unicamente caratteri presenti in s2
• **strstr()**	• **const char* strstr(const char* s1, const char* s2)** Cerca la stringa *s2* in *s1* e restituisce un puntatore al carattere dove comincia *s2*
• **strcspn()**	• **size_t strcspn(const char* s1, const char* s2)** Restituisce la lunghezza della sottostringa più lunga di s1 che comincia con s1[0] e non contiene alcuno dei caratteri presenti in s2
• **strpbrk()**	• **const char *strpbrk(const char* s1, cont char* s2)** Restituisce l'indirizzo della prima occorrenza in s1 di qualunque dei caratteri di s2. Restituisce NULL se nessuno dei caratteri di s2 appare in s1
• **memcpy()**	• **void* memcpy(void* s1, const void* s2, size_t n)** Rimpiazza i primi *n* bytes di *s1 con i primi *n* bytes di *s2. Restituisce s1
• **strnset()**	• **char* strnset(char* s, int ch, size_t n)** Utilizza strcmp() su una stringa esistente per fissare *n* bytes della stringa al carattere *ch*
• **strtok()**	• **char* strtok(char* s1, const char* s2)** Divide la stringa s1 in sottostringhe delimitate dai caratteri presenti nella stringa s2. Dopo la chiamata iniziale strtok(s1, s2), ogni chiamata successiva a strtok(NULL, s2) restituisce un puntatore alla successiva sottostringa in s1. Queste chiamate cambiano la stringa s1, rimpiazzando ogni separatore con il carattere NULL ('\0')

Copyright © 2021
McGraw-Hill Education (Italy) S.r.l.
Mc Graw Hill