Soluzione esercizio 1 su Array

```
#include <iostream.h>
int main ()
 { int voti[6], i, max, min;                                  // dichiarazione variabili
   float media;

   cout << "Immissione Voti Studenti" << endl;    // avvisa che si devono inserire voti
   for(i=0;i<6;i++) {
      cout << "Dammi il “<< i+1 << "° voto:";              // chiede un voto alla volta
      cin >> voti[i];                          // ogni voto viene memorizzato nell’array
   }
   max = min = voti[0];            // assumo che voti[0] sia il valore massimo e minimo
   media = voti[0];      // inizializzo media a voti[0], poi aggiungero’ gli altri voti
   for(i=1;i<6;i++) {
      if(voti[i]>max) max = voti[i];                   // aggiorno il valore di massimo
      if(voti[i]<min) min = voti[i];                    // aggiorno il valore di minimo
      media += voti[i];  // aggiorno la somma dei voti; per avere media va diviso per 6
   }
   media /= 6;                              // divido per 6 in modo da ottenere la media
   cout << "Voto maggiore: " << max << “ Voto minore: " << min << endl;
   cout << “Media: " << media;                                      // stampa risultati
   return 0;
}
```

Soluzione esercizio 2 su Array

```
#include <iostream.h>
void leggidatiingresso (int dati[], int lunghezza)
 { int i;
   cout << "Immissione Voti Studenti" << endl;    // avvisa che si devono inserire voti
   for(i=0;i<lunghezza;i++) {
      cout << "Dammi il “<< i+1 << "° voto:";              // chiede un voto alla volta
      cin >> dati[i];                          // ogni voto viene memorizzato nell’array
   }
 }
int main ()
 { int voti[6], i, max, min;                                  // dichiarazione variabili
   float media;
   leggidatiingresso (voti, 6);
   max = min = voti[0];            // assumo che voti[0] sia il valore massimo e minimo
   media = voti[0];      // inizializzo media a voti[0], poi aggiungero’ gli altri voti
   for(i=1;i<6;i++) {
      if(voti[i]>max) max = voti[i];                   // aggiorno il valore di massimo
      if(voti[i]<min) min = voti[i];                    // aggiorno il valore di minimo
      media += voti[i];  // aggiorno la somma dei voti; per avere media va diviso per 6
   }
   media /= 6;                              // divido per 6 in modo da ottenere la media
   cout << "Voto maggiore: " << max << “ Voto minore: " << min << endl;
   cout << “Media: " << media;                                      // stampa risultati
   return 0;
}
```

Altro esercizio su Array

- Esercizio 3:

  Scrivere un programma che riceve da tastiera i valori associati a due matrici A (di dimensione n x p) e B (di dimensione p x m), le cui dimensioni sono definite nel programma attraverso una direttiva #define, e stampi la matrice prodotto C=AB. Vi ricordo la formula per il prodotto di matrici:

$$C_{ij} = \sum_k A_{ik} B_{kj}$$

Soluzione esercizio 3 su Array

```
#include <iostream.h>
#define N 4
#define P 3
#define M 5
int main ()
 {
   int i, j, k;
   int mat1[N][P], mat2[P][M], pmat[N][M];

   cout << "Immissione Prima Matrice" << endl;
   for(i=0;i<N;i++)
    for(j=0;j<P;j++) {
      cout << "Inserisci il valore dell'elemento "<< i+1 << "," << j+1;
      cin >> mat1[i][j];
    }

   cout << "Immissione Seconda Matrice" << endl;
   for(i=0;i<P;i++)
    for(j=0;j<M;j++) {
      cout << "Inserisci il valore dell'elemento "<< i+1 << "," << j+1;
      cin >> mat2[i][j];
    }
```

Soluzione esercizio 3 su Array

```
   for(i=0;i<N;i++)
    for(j=0;j<M;j++) {
      pmat[i][j]=0;
      for(k=0;k<P;k++)
         pmat[i][j] += mat1[i][k]*pmat2[k][j];
    }
   cout << "Matrice Prodotto:" << endl;
   for(i=0;i<N;i++) {
    cout << endl;
    for(j=0;j<M;j++)
      cout << pmat[i][j] << " " ;
    }
   return 0;
}
```

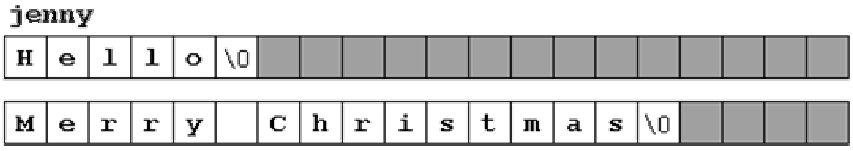
Stringhe di caratteri

- Finora abbiamo usato le stringhe di caratteri come costanti ma non abbiamo ancora visto variabili che possano contenerle.
- In C++ non vi è un tipo di variabile predefinito in grado di memorizzare delle stringhe di caratteri. Dobbiamo usare degli array di caratteri.
- Vediamo ora il trattamento standard (stile C) delle stringhe come array di caratteri.
  Il seguente array:

```
char jenny [20];
```

può memorizzare una stringa di al più 20 caratteri. Possiamo rappresentarlo come:

- Non è necessario usare tutti e 20 i caratteri dell'array. Per questo motivo occorre prevedere una indicazione del punto in cui termina la stringa. Per convenzione tale punto viene indicato dal carattere nullo che si può scrivere sia **0** sia **'\0'**:



Stringhe: inizializzazione

- Per inizializzare una stringa di si può usare la stessa notazione usata per gli array:

```
char mystring[] = { 'H', 'e', 'l', 'l', 'o', '\0' };
```

- In alternativa si può usare anche una stringa costante:

```
char mystring [] = "Hello";
```

- In entrambi i casi la dimensione dell'array **mystring** è di 6 elementi di tipo **char**: i 5 caratteri di **Hello** e il carattere nullo finale (**'\0'**).
- **Attenzione**: una stringa costante può essere usata per dare un valore iniziale ad una variabile di tipo array di **char** soltanto al momento della dichiarazione dell'array, ossia in fase di inizializzazione.
- Assegnamenti quali:

```
mystring = "Hello";
mystring[] = "Hello";
```

non sono permessi, come non è permesso:

```
mystring = { 'H', 'e', 'l', 'l', 'o', '\0' };
```


Stringhe: assegnamenti

- Siccome in un assegnamento l'lvalore può essere soltanto un elemento di un array e non l'intero array, per assegnare una stringa di caratteri ad un array di **char** dobbiamo scrivere:

```
mystring[0] = 'H';
mystring[1] = 'e';
mystring[2] = 'l';
mystring[3] = 'l';
mystring[4] = 'o';
mystring[5] = '\0';
```

- Poiché questo non è molto pratico la libreria standard **cstring** (che si può includere con la direttiva **#include <string.h>**) contiene la definizione di un certo numero di funzioni, quali **strcpy** (**str** ing **co** **py**) che si può richiamare nel seguente modo:

```
strcpy (string1, string2);
```

l'effetto è copiare il contenuto di ***string2*** in ***string1***. ***string2*** può essere sia un array sia una stringa costante, il che ci permette di assegnare la stringa costante **"Hello"** all'array di caratteri **mystring** usando la seguente notazione:

```
strcpy (mystring, "Hello");
```

Stringhe: assegnamento

Esempio di assegnamento realizzato utilizzando la libreria standard

```
// assegnamento a stringhe
#include <iostream.h>
#include <string.h>

int main ()
{
  char stMyName [20];
  strcpy (stMyName,"J. Soulie");
  cout << stMyName;
  return 0;
}
```

```
J. Soulie
```

Esempio di assegnamento realizzato "in casa": cioè definendo una nostra funzione di copia

```
// assegnamento a stringhe
#include <iostream.h>

void setstring (char stOut [], char stIn [])
{
  int n=0;
  do {
    stOut[n] = stIn[n];
  } while (stIn[n++] != '\0');
}

int main ()
{
  char stMyName [20];
  setstring (stMyName,"J. Soulie");
  cout << stMyName;
  return 0;
}
```

```
J. Soulie
```


Stringhe: assegnamenti

- Un altro modo per assegnare un valore ad un array di caratteri è quello di usare direttamente il flusso di input **cin**.
- Nella libreria **iostream** è infatti definita una funzione **getline** il cui prototipo è:

```
cin.getline ( char buffer [], int length, char delimiter = '\n');
```

dove **buffer** è l'array di caratteri in cui memorizzare l'input, **length** è la dimensione dell'array stesso e **delimiter** è il carattere usato per indicare la fine dell'input e per il quale è previsto il carattere nuova linea (**'\n'**) come valore di default.

```
// uso di cin.getline
#include <iostream.h>

int main ()
{
  char buff [100];
  cout << "Come ti chiami? ";
  cin.getline (buff,100);
  cout << "Salve " << buff << ".\n";
  cout << "La tua squadra preferita? ";
  cin.getline (buff,100);
  cout << "L'" << buff << " piace anche a me.\n";
  return 0;
}
```

```
Come ti chiami? Juan
Salve Juan.
La tua squadra preferita? Inter
L'Inter piace anche a me.
```

Stringhe: assegnamenti

- Si può anche usare l'operatore di estrazione (>>) per leggere delle stringhe da **cin** :

```
cin >> buff;
```

che funziona ma con le seguenti limitazioni che **cin.getline** non ha:

- si possono leggere soltanto parole e non intere frasi in quanto l'operatore di estrazione usa come delimitatore qualsiasi occorrenza di un carattere invisibile (spazio, tabulazione, nuova linea, ritorno carrello).
- non si può specificare la dimensione dell'array il che rende instabile il programma nel caso in cui l'input sia una parola più lunga della dimensione dell'array.

- Altre funzioni che operano su stringhe sono definite nella libreria **cstring** (**string.h**).

Esercizi su stringhe

- Esercizio 4:
  Scrivere una funzione che date due stringhe (di lunghezza massima M) restituisca la stringa risultante dalla concatenazione della prima con la seconda:

```
void concatenastringhe(char st1[], char st2[], char stOut[])
```

- Esercizio 5:
  Scrivere una funzione che date due stringhe (di lunghezza massima M) restituisca la stringa che alterna i caratteri della prima con quelli della seconda:

```
void mischiastringhe(char st1[], char st2[], char stOut[])
```

Esempi:
dati "**ancora**", "**cane**" restituisce "**acnacnoera**"
"**amo**", "**salmone**" restituisce "**asmaolmone**"