

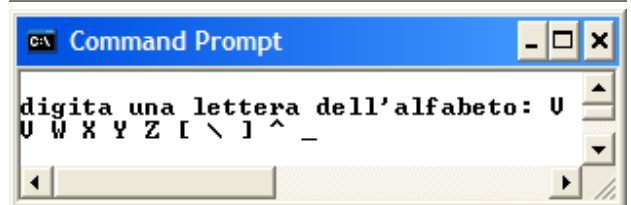
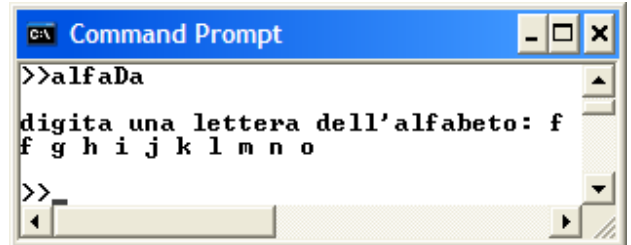
dati di tipo **carattere (CHAR)** - **ARRAY di caratteri** / **STRINGHE** - **ARRAY di stringhe**

1) tipo **carattere (CHAR)**

Il programma **alfaDa.cpp** scrive a video 10 caratteri a partire dalla lettera digitata.

La variabile **let** di tipo *char* viene **inizializzata** con il **carattere** : (*due punti*) che va racchiuso tra **apici**.

```
#define CH "digita una lettera dell'alfabeto"
#define SP ' '
#define N 10
main() {
    char let=':';
    int n;
    cout << endl << CH << let << SP;
    cin >> let;
    for (n=1; n <= N; n++) //--- scrive let
    { cout << let << SP; //--- poi spazio
      let = let + 1; //---let incrementato di 1
    } //--- let contiene il carattere successivo
    cout << endl;
}
```

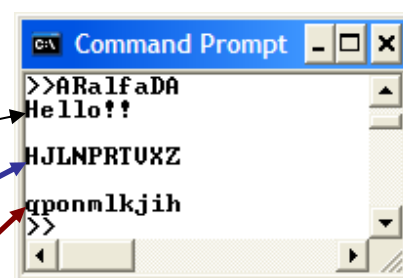


NOTA: su una variabile di tipo *char* nel linguaggio C/C++ si possono effettuare operazioni aritmetiche (operazioni che vengono effettuate sul codice ASCII del carattere in essa contenuto; in ordine di codice ASCII, dopo Z maiuscolo ci sono alcuni caratteri speciali).

2) **ARRAY di caratteri**

Il programma **ARalfaDa.cpp** espone a video il contenuto dell'**ARRAY di caratteri lettera[N]** che viene **inizializzato** con 10 caratteri (racchiusi tra **apici**) oppure con un codice ASCII espresso in base **10**, **16** (il valore deve essere preceduto da **0x**) o **8** (il valore deve essere preceduto da **0**).

```
main() {
    char lettera[N]={72, 0x65, 'l', 0154, 'o',
                   33, '! ', SP, 0x20, 040};
    char salva = lettera[0];
    for (int i=0; i < N; i++)
    { cout << lettera[i];
      lettera[i] = salva + 2*i;
    }
    cout<<"\n\n";
    for (int i=0; i < N; i++)
    { cout << lettera[i];
      lettera[i] = salva + 32 + i;
    }
    cout<<"\n\n";
    for (int i=N-1; i >= 0; i--)
        cout << lettera[i];
}
```



(Le costanti N ed SP come nel primo programma)

Nella prima riga i valori iniziali di **lettera[N]**

Nella seconda riga i nuovi valori dell'array (calcolati nel primo ciclo).

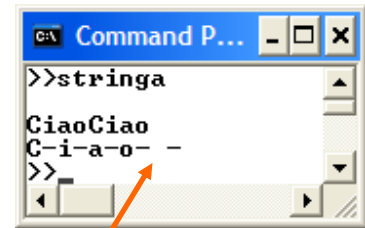
Nella terza riga i nuovi valori dell'array (calcolati nel secondo ciclo), ma esposti in ordine inverso (dall'ultima posizione alla prima).

La costante esadecimale **0x20** equivale a SP e alla costante ottale **040**: rappresentano uno spazio ' '.

3) ARRAY di caratteri / STRINGHE

Il programma **stringa.cpp** scrive a video il contenuto dell'ARRAY di caratteri (*parola[N]*) dopo averlo **inizializzato** con una **stringa** delimitata da **virgolette**, poi lo riscrive separando ogni carattere con un trattino.

```
#define TR '-'
#define N 5
main() {
    char parola[N]="Ciao"; //--- 4 caratteri (posizioni 0-3)
    cout << endl << parola << parola << endl;
    for (int i=0; i < N; i++)
        cout << parola[i] << TR ;
}
```



L'ultimo elemento dell'array *parola* contiene il **delimitatore di stringa**, **NULL** (codice ASCII: 8 bit a zero), carattere **non** rappresentabile a video (appare come un carattere *spazio*).

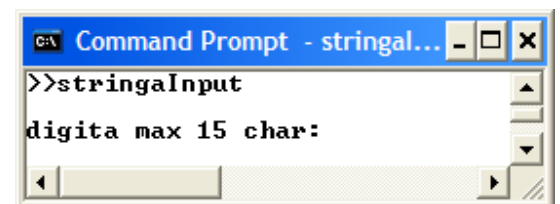
Il NULL è stato accodato dall'inizializzazione come quinto elemento del vettore *parola* (NULL è in posizione 4). Il NULL viene accodato automaticamente anche acquisendo una stringa da tastiera mediante **cin**.

4) ARRAY di caratteri / STRINGHE

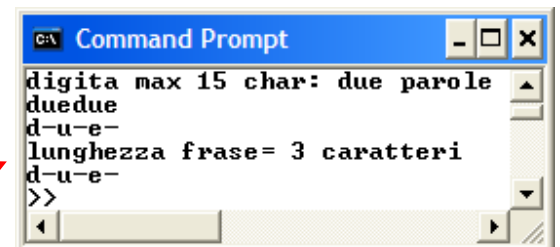
Il programma **stringaInput.cpp** legge una **stringa** (mediante **cin**) memorizzandola in un **ARRAY** di 15 **caratteri** di nome *frase[N]* che poi riscrive a video due volte (mediante **cout**) e ancora una terza volta separando ogni carattere con un trattino (mediante **cout**).

ATTENZIONE: solo per gli array di caratteri **cin** e **cout** operano su tutti gli elementi contemporaneamente se seguiti dal nome dell'array senza [dimensione/indice]

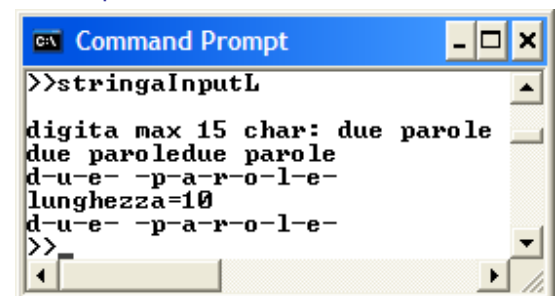
```
main() {
    const char TR = '-';
    const int N = 15;
    char frase[N]; int i, lung;
    cout << "\ndigita max 15 char: ";
    cin >> frase;
    cout << frase << frase << endl;
    for (i=0; frase[i] != '\0'; i++)
        cout << frase[i] << TR ;
    lung = strlen(frase);
    cout << "\nlunghezza frase= " << lung;
    cout << " caratteri" << endl;
    for (int i=0; i < lung; i++)
        cout << frase[i] << TR ;
}
```



In basso l'output digitando: due parole



ATTENZIONE: **cout << frase** non espone il NUL a video



in alto l'output a video, sostituendo:

cin >> frase;
con:
cin.getline(frase, N);

ATTENZIONE: il carattere **spazio**, la **tabulazione** e l'**invio** vengono interpretati da **cin** come **fine stringa** e i caratteri successivi vanno persi (o considerati da un successivo **cin**).

Nel vettore *frase[N]* dopo i caratteri della prima parola troviamo il carattere NULL accodato automaticamente in fase di lettura.

La funzione **strlen** fornisce il numero di caratteri che compongono la stringa che può essere scorsa, carattere per carattere, fino al NULL (rappresentabile nel programma con la sequenza di escape **'\0'**) oppure fino all'elemento di posizione **strlen(frase)-1**

Per acquisire una stringa di lunghezza N formata da parole separate da spazi (un testo) va utilizzato il metodo **getline** di **cin**, indicando il numero di caratteri **N** da considerare:

5) ARRAY di stringhe

Un **ARRAY di stringhe** è una **matrice** (array bidimensionale) di caratteri in cui ogni riga è una **stringa**.

Il programma **ARstringhe.cpp** utilizza un **ARRAY di 7 stringhe** di **4** caratteri ognuna `giorniSet[N][4]` per rappresentare i giorni della settimana (il quarto carattere, per ogni stringa, è il NULL di fine stringa).

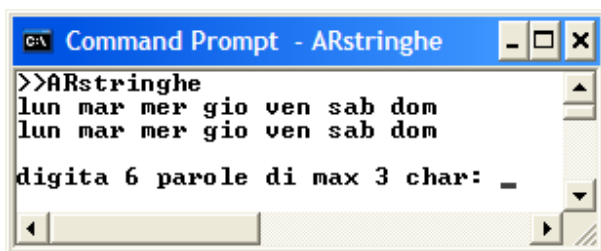
Per indicare ogni stringa occorre specificare solo il valore del primo indice (quello di riga); quindi, per scrivere a video *sab* (sabato) basta codificare `cout<<giorniSet[6];`

Il programma scrive a video i giorni della settimana per due volte a partire dalla posizione 1 (*lun*) fino alla posizione 0 (*dom*); poi attende la digitazione di 6 stringhe e le riscrive a video.

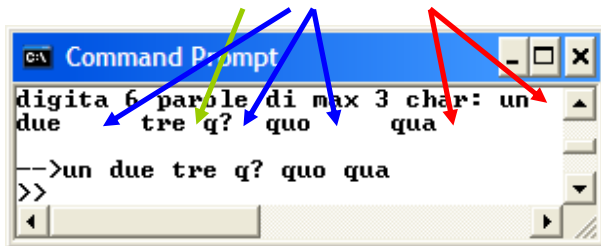
```
#define SP ' '
#define N 7
main() {
    char giorniSet[N][4] = {"dom" , "lun", "mar",
                          "mer", "gio" , "ven" , "sab" };

    for (int i=1; i <= 2*N; i++)
    {
        cout << giorniSet[(i+N)%N] << SP;
        if (i%N == 0)
            cout<<endl;          //--- a capo ogni 7 stringhe
    }                             //--- per i=7 e per i=14(ultimo)
    cout<<endl<< "digita 6 parole di max 3 char: ";
    cin >>giorniSet[0]>> giorniSet[1]>> giorniSet[2];
    cin >>giorniSet[3]>> giorniSet[4]>> giorniSet[5];
    cout<< endl << "-->";
    for (int i=0; i <= 5; i++)//--- scrittura 6 stringhe lette
        cout << giorniSet[i] << SP;
}
```

L'esito a video del programma è il seguente: scritto 2 volte a video l'array `giorniSet[N][4]`, il programma resta in attesa della digitazione di 6 stringhe:



Le 6 stringhe vengono lette con **cin**, pertanto spazio, tabulazione e invio vengono interpretati come fine stringa. Nell'eseguire il programma, le 6 stringhe sono state separate con **spazio, tab e invio**:



Le sei stringhe lette e salvate nell'array `giorniSet[N][4]` vengono riscritte a video separate da un carattere spazio e precedute da una feccia (-->).

La formula $(i+N)\%N$ utilizzata per scorrere gli elementi dell'array `giorniSet` consente di elaborarlo riga per riga (una stringa dopo l'altra; giorno dopo giorno), **circolarmente (modulo 7=N)**.

In questo modo ogni elemento ha un successivo, anche l'ultimo elemento dell'array, `giorniSet[6]`, ha un successivo: `giorniSet[0]` raggiunto quando i vale 7 (e poi 14) perché, in questo caso, la formula diventa:

$$(i+N)\%N = (7+7)\%7 = 14\%7 = 0$$

Analogamente, dopo `giorniSet[0]` verrà scritto `giorniSet[1]` perché per $i=8$:

$$(i+N)\%N = (8+7)\%7 = 15\%7 = 1$$

L'array di stringhe `giorniSet[N][4]` è una **matrice** di **7x4=28** caratteri, costituita da **7 righe** e **4 colonne**

	COLONNE			
	0	1	2	3
0	d	o	m	NUL
1	l	u	n	NUL
2	m	a	r	NUL
3	m	e	r	NUL
4	g	i	o	NUL
5	v	e	n	NUL
6	s	a	b	NUL

`giorniSet[0][0]` contiene **d**

`giorniSet[4][0]` contiene **g**

`giorniSet[6][2]` contiene **b**

`giorniSet[i][3]` contiene NULL

per ogni valore di i tra 0 e 6 ($i \geq 0$ e $i < 7$).

Quindi `giorniSet[i][3]=='\0'` è una condizione VERA per i nell'intervallo 0-6

A fine elaborazione:

`giorniSet[0][0]` vale 'u'

`giorniSet[3][1]` vale '?'

`giorniSet[3][2]` vale '\0' (NUL)