## Operatori di incremento e decremento

- ++ e --
- Esempi
  - y=x++

    il contenuto della variabile x viene inizialmente usata per l'assegnazione y, e solo dopo incrementata
  - y=--x

    il contenuto della variabile x viene inizialmente decrementato di uno e solo dopo viene effettuata l'assegnazione di x a y



# Strutture dati

Vettori e matrici

## Strutture di dati

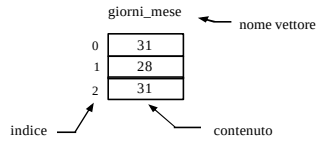
Una struttura di dati ha le seguenti proprietà:

- è un insieme di dati
  - *Es. una struttura contenente il totale delle vendite effettuate da un'azienda in ognuno dei dodici mesi di un anno*
- ogni dato dell'insieme può essere singolarmente identificato rispetto agli altri
  - *Es. si deve poter conoscere il totale venduto in un certo mese*
- la modalità di identificazione dei singoli dati, consente che una stessa istruzione, eseguita in momenti diversi, possa operare su diversi dati della struttura
  - *Es. deve essere possibile leggere, con un ciclo, tutte le fatture (emesse da un'azienda, contenute in un archivio)*
- possono esistere operazioni che agiscono a livello dell'intera struttura



## Vettori

- un vettore (*array unidimensionale*) è un insieme di elementi dello stesso tipo
- ha un nome che lo identifica
- gli elementi del vettore vengono identificati, oltre che con il nome del vettore di appartenenza, con il valore di un indice numerico






## Vettori

- per identificare un elemento si utilizza il nome del vettore seguito dal valore dell'indice racchiuso tra parentesi.

  *es. giorni_mese [3]*
- il valore dell'indice può essere una qualsiasi espressione, ad esempio

  *es. giorni_mese [n+3]*
- occorre dichiarare il tipo degli elementi e la dimensione del vettore (in C i vettori hanno dimensione prefissata)

  *<tipo> <nome> '[' <dimensione> ']'*
- il valore dell'indice parte da zero
- un vettore occupa locazioni contigue di memoria
- il C non effettua alcun controllo (né in fase di compilazione, né in fase di esecuzione) sul superamento dei limiti di un vettore



## Vettori

```
/* Programma che legge n numeri in un vettore e li visualizza in ordine
   inverso */
#include <stdio.h>
#define MAX  100
int main (){
   int n, i, numeri[MAX ];
   /* Lettura dimensione vettore */
   printf ("Inserire il numero di elementi:");
   scanf ("%d", &n);
   if (n>MAX) printf ("Valore troppo elevato\n");
   else{
        /* lettura dei numeri */
        for (i=0; i<n; i++){
                printf ("Inserire un numero:");
                scanf ("%d", &numeri[i]);
        }
        /* visualizzazione dei numeri in ordine inverso */
        for(i=n-1; i>=0; i--) printf ("%d\n", numeri[i]);
   }
   return 0;}
```

## Array Multidimensionale

- hanno due o più indici (o *dimensioni*)
- in C, occorre dichiarare la dimensione di ogni indice

<tipo> <nome> ‘[‘<dimensione>‘]’ { ‘[‘<dimensione>‘]’ }

- i vettori bidimensionali sono denominati *matrici*



## Esempio

```
/*legge, per righe, gli elementi di una matrice 2 3 e stampa la somma di ogni riga*/
#include <stdio.h>
#define N_RIGHE  2
#define N_COLONNE  3
int main(){
   int i, j, somma, numeri[N_RIGHE][N_COLONNE];
   for (i=0; i<N_RIGHE; i++)    {
      printf (“Inserisci riga n.%d\n”, i+1);
      for (j=0; j<N_COLONNE; j++) {
          printf (“Inserisci elemento n.%d:”, j+1);
          scanf (“%d”, &numeri[i][j]);
      }
   }
   for (i=0; i<N_RIGHE; i++)    {
      somma=0;
      for (j=0; j<N_COLONNE; j++)
          somma+=numeri[i][j];
      printf (“La somma della riga n.%d vale %d\n”, i+1, somma);
   }
    return 0;}
```



## Caratteri e stringhe

<string.h>

## Caratteri e Stringhe

- *Lettura di un carattere da tastiera* (dispositivo standard di input):

  **`getchar()`**

  accetta caratteri fino alla pressione del tasto invio;
  solo allora restituisce il primo carattere inserito; gli altri restano in memoria.

- *Scrittura di un carattere sullo schermo* (dispositivo standard di output)

  **`putchar (`**<**`espressione carattere`**>**`)`**



## Caratteri e stringhe

- Esempi

  char c;
  c=getchar();
  putchar(c)



## Stringa

- È una sequenza di caratteri
- In C non esiste il dato di tipo stringa; una stringa viene memorizzata in un array di caratteri

  **`char nome[30]`**

- Il C usa la tecnica di contrassegnare la fine effettiva di una stringa con il carattere avente codice 0 ( ‘\0’)
- Inizializzazione di una stringa:
  **`char nome[30]=”Luca”`** (inserisce automaticamente il carattere ‘\0’)

## Lettura di una stringa da tastiera

**gets (<array>)** //termina l'immissione con il tasto "invio"

**scanf ("%s", <array>)** //termina l'immissione con il tasto "invio" (il codice del tasto invio non viene letto)

- in entrambi i casi, l'inserimento di un numero di caratteri maggiore della lunghezza dell'array, ha effetti imprevedibili



## Scrittura di una stringa su schermo

```
puts(<espressione stringa>)  //aggiunge '\n'
printf ("%s", <espressione stringa>)

Es. puts (stringa);
    printf ("%s %s\n", stri1, stri2);
```



## Esempio

```
//legge una stringa e conta le eventuali cifre presenti al suo
   interno
#include <stdio.h>
#define MAX_CAR  128
int main() {
   int i, n_cifre=0;
   char stringa [MAX_CAR];
   puts ("Inserire una stringa:");
   gets (stringa);
   for (i=0; i<MAX_CAR; i++) {
       if (stringa[i]=='\0') break;
       if (stringa[i]>='0' && stringa[i]<='9')
       n_cifre++;
   }
   printf ("Il numero di cifre è %d\n", n_cifre);
  return 0;
}
```



## Funzioni di manipolazione delle stringhe

- **Per il loro utilizzo occorre includere il file** string.h

**strcpy (s1, s2)** **copia la stringa s2 in s1**

**strcat (s1, s2)** **concatena s2 alla fine di s1**

**strlen (s)** **restituisce la lunghezza della stringa s**

**strcmp (s1, s2)** **confronta s1 con s2**