ARRAY IN LINGUAGGIO C

Cosa sono?

Sono delle **strutture dati** previste dal linguaggio C. Cosa si intende per "**struttura di dati** "oppure "dati strutturati"? Si intendono insiemi di **dati** che sono "in qualche modo" legati fra di loro.

Nel caso dell'array, il legame fra i dati è che essi sono **omogenei**. **Omogenei** significa che sono costituiti da variabili tutte dello stesso tipo (tutti interi o tutti float o tutti double ecc.). L'array è identificato da un unico nome mentre il singolo elemento dell'array è identificato dal nome dell'array con l'aggiunta di un indice.

Esempio 1:

Array "V" formato da 6 elementi:

indice	0	1	2	3	4	5
V	8	2	5	1	6	5

Un generico elemento dell'array viene indicato con V[i] dove

V: è il nome dell'array

[i]: fra parentesi quadre, è l'indice dell'array, identifica la posizione dell'elemento nell'array

OSSERVAZIONE: **gli indici di un array di N elementi vanno sempre da 0 ad N-1 (si parte a contare gli indici sempre da 0), quindi l'elemento, ad esempio, in quinta posizione è quello con indice uguale 4, indicato con V[4].**

Esempio: L' elemento con indice 4 dell'array V ha valore 6, se si vuole indicare l'elemento con indice 4 (i=4) si scrive quindi:

V[4] (il cui valore è 6)

Un array ad una dimensione (monodimensionale), come quello visto è chiamato **vettore, vedremo più avanti che ci saranno anche array con più dimensioni.**

DICHIARAZIONE DI UN VETTORE IN C

La dichiarazione prevede sempre: **tipo_di_dato nome_array [dimensione];**

Esempio: **int v[10];**

La dimensione va indicata fra parentesi quadre!

INIZIALIZZAZIONE DI UN ARRAY IN C

Per inizializzare un' array si possono inserire i valori contestualmente alla dichiarazione nel seguente modo:

int V[6]={8,2,5,1,6,1} **(solo contestualmente alla dichiarazione)**

int V[6]={0} in questo caso tutti i 6 elementi valgono 0 **(vale solo con 0)**

Questa inizializzazione (solo l'inizializzazione intendo) può essere fatta solo se la dimensione fra parentesi è specificata direttamente con un numero oppure con la direttiva al preprocessore #define. Esempio:

#define dimensione 6

int V[6]={8,2,5,1,6,1}; OK

invece

const int dimensione=6;

int V[6]={8,2,5,1,6,1}; **//ERRORE, NON VIENE COMPILATO**

Per modificare un elemento dell'array si individua l'elemento e si assegna il valore:

V[3]=10; in seguito a questa istruzione l'elemento con indice 3 vale 10 e non più 1

Gli array possono avere più di una dimensione (noi vedremo quelli ad 1 e 2 dimensioni), gli array a due dimensioni sono dette **matrici e li vedremo dopo.**

ATTENZIONE: se si "sfora" l'array assegnando un valore ad un elemento con indice maggiore della dimensione dell'array viene sovrascritta una zona di memoria RAM che contiene altro, non si sa cosa, magari un processo dell'OS. Gli esiti sono imprevedibili (anche crash del sistema). Nell'esempio precedente:

v[10000]=34; //non si può prevedere cosa accadrà, possibile anche crash del sistema

A COSA SERVONO GLI ARRAY?

Sono utili al il programmatore quando si hanno più variabili che rappresentano dati "simili", ad esempio nel seguente problema:

> si vogliono acquisire i dati relativi ai millimetri di pioggia caduti **per ogni mese** di un determinato anno per poi calcolarne il valor medio ed individuare i mesi in cui le precipitazioni sono state superiori alla media.

In un problema di questo tipo risulta più comodo utilizzare un array di interi "mmPioggia" anziché 12 variabili (una per ciascun mese dell'anno). La comodità sta nel fatto che è possibile utilizzare la struttura iterativa (un ciclo for) per operare sui dati. Inoltre si ha il vantaggio del riutilizzo del codice, infatti, nel caso si volesse

modificare il software acquisendo i millimetri di pioggia caduti ogni giorno anziché ogni mese, basta modificare la dimensione dell'array senza dover creare una variabile per ogni giorno dell'anno.

Esempio per mostrare la dichiarazione e l'inizializzazione di un array: far inserire all'utente i mm di pioggia di ogni mese e poi visualizzare sul monitor i mm di pioggia caduti per ogni mese dell'anno.

```
5   int main(void)
6   {
7       const int dimensione=12;     //dimensione dell' array
8       int pioggia[dimensione];
9
10      //input dati
11      for (int i=0;i<12;i++)
12      {
13          printf("\ninserisci i mm di pioggia del mese %i ",i);
14          scanf("%i",&pioggia[i]);
15      }
16      //output dati
17      for (int i=0;i<12;i++)
18      {
19          printf("\nmese %i --> %i",i+1,pioggia[i]);
20      }
```

La dimensione dell'array deve essere una **costante.** In questo caso la costante **"dimensione"**

L'array si usa sempre con il **ciclo for** perché il ciclo consente di eseguire la stessa operazione su tutti i dati dell'array. In questi due esempi le operazioni sono input e output di valori nell'array

L'indice dell'array generalmente è la una **variabile** ((in questo caso chiamata "i") che si incrementa di 1 per ogni iterazione del ciclo

OSSERVAZIONE: per mostrare il contenuto di un array va mostrato UN ELEMENTO ALLA VOLTA, non è possibile utilizzare un unico comando printf() per visualizzare tutto il contenuto dell'array.

Esercizio provare a fare da soli poi correggere con la soluzione mostrata di seguito: utilizzando l'array dell'esempio precedente mostrare la pioggia complessivamente caduta durante l'anno (somma della pioggia di ogni mese) e la pioggia caduta per ogni trimestre (somma della pioggia di ogni mese del trimestre).

```c
int main()
{
    const int dimensione=12;     //dimensione dell' array
    int pioggia[dimensione];
    int totale=0,trimestre1=0,trimestre2=0,trimestre3=0,trimestre4=0;

    //input dati
    for (int i=0;i<12;i++)
    {
        printf("\ninserisci i mm di pioggia del mese %i ",i+1);
        scanf("%i",&pioggia[i]);
    }
    //sommatorie
    for (int i=0;i<12;i++)
    {
        printf("\nmese %i --> %i",i+1,pioggia[i]);
        if (i>=0 && i<3)
            trimestre1=trimestre1+pioggia[i];
        else if (i>=3 && i<6)
            trimestre2=trimestre2+pioggia[i];
        else if (i>=6 && i<9)
            trimestre3=trimestre3+pioggia[i];
        else
            trimestre4=trimestre4+pioggia[i];
        totale=totale+pioggia[i];
    }
    //output
    printf("\n Trimestre1: %i mm ",trimestre1);
    printf("\n Trimestre2: %i mm ",trimestre2);
    printf("\n Trimestre3: %i mm ",trimestre3);
    printf("\n Trimestre4: %i mm ",trimestre4);
    printf("\n Totale: %i mm ",totale);

    return 0;
}
```

Esercizio: mesi piovosi. Far inserire all'utente i mm di pioggia per ciascun mese, calcolare la media e mostrare quanti sono i mesi in cui la pioggia è stata superiore alla media (mesi piovosi)

Esercizio: come risulta il seguente vettore applicando la seguente sequenza di istruzioni? :

indice	0	1	2	3	4	5
V	8	2	4	1	6	5

……

i=V[2];

j=1;

V[i-j]=V[i]+j;

….

Risultato (attenzione, sul libro è sbagliato, trova l'errore):

indice	0	1	2	3	4	5
V	8	2	5	**7**	6	5

Esempio: Scambio di due elementi dell'array.

Lo scambio di due elementi di un array è un'operazione spesso utilizzata (impararla bene!). Scambiare i due elementi significa scambiare fra loro I CONTENUTI dei due celle dell'array. Lo scambio avviene con la stessa

procedura di scambio fra due variabili, quindi utilizzando una variabile di appoggio che deve essere dello stesso tipo dei dati contenuti nell'array. (Scrivere il codice per scambiare fra loro il primo e l'ultimo valore dell'array dell'esercizio precedente).

```
5   int main(void)
6   {
7
8       int v[]={8,2,4,1,6,5};
9       int x;
10
11      x=v[0];
12      v[0]=v[5];
13      v[5]=x;
14
15
16      //output dati
17      for (int i=0;i<6;i++)
18      {
19          printf("\nv --> %i",v[i]);
20      }
21  }
```

OSSERVAZIONE: Attenzione a non confondere gli indici con i contenuti di un array!

OSSERVAZIONE: Gli array possono avere anche più di due dimensioni (3,4…) in tal caso per indicare un elemento sarà necessario un indice per ogni dimensione. Noi useremo solo vettori (array ad una dimensione) e matrici (array a due dimensioni).

Esercizi:

ESERCIZIO 1.

È dato il seguente vettore denominato `alfa`:

2	1	5	3	6	5	1	3	4	6	2

Indicare come esso si modifica in seguito a ognuno dei gruppi di istruzioni a partire dalla situazione iniziale illustrata:

A.
```
int i = alfa[3];
int j = 3;
alfa[i-j] = alfa[i] + j;
```

B.
```
int i = alfa[5];
int j = alfa[i-1]+ 1;
alfa[j] = alfa[i]+ alfa[j];
```

C.
```
for (int i = 1; i <= 9; i++)
    {
      int s = 0;
      for (int j = 0; j < i; j++)
        s += alfa[j];
      alfa[i] = s;
    }
```

Riporto il vettore con gli indici:

indice	0	1	2	3	4	5 ….					
alfa	2	1	5	3	6	5	1	3	4	6	2

Soluzione:

A)

i=3

j=3

alfa [0] = alfa[3]+3= 6

indice	0	1	2	3	4	5					
alfa	**6**	1	5	3	6	5	1	3	4	6	2

B)

i=5

j=alfa[4] +1 = 7

alfa [7] = alfa[5]+alfa[7] = 5+3 = 8

indice	0	1	2	3	4	5					
alfa	2	1	5	3	6	5	1	**8**	4	6	2

C)

indice	0	1	2	3	4	5					
alfa	2	2	4	8	16	32	64	128	256	512	2

Esercizio 2 max min media. Scrivere un programma C che acquisisca da tastiera un vettore di interi di dimensione n (n è una costante che vale, ad esempio, 5) e calcoli minimo, massimo e media degli elementi.

Esercizio 3 media su tre punti: uno strumento di misura fornisce un dato ogni minuto nell'arco di un'ora. Per ovviare a possibili errori si vogliono elaborare i valori rilevati sostituendoli con una media a tre punti: ogni elemento viene sostituito dalla media di se stesso, dell'elemento che lo precede e di quello che lo segue; per i due elementi estremi viene considerato due volte il valore dell'elemento stesso e il successivo o il precedente nel caso si tratti rispettivamente del primo o dell'ultimo elemento. Realizzare un programma C che implementi l'elaborazione descritta acquisendo i dati da tastiera. Considerare l'inserimento di 10 valori da tastiera.

Array monodimensionali (vettori) come parametri di funzioni

Gli array possono essere passati come parametri alle funzioni ed il loro passaggio avviene sempre per **indirizzo** (detto anche passaggio per **riferimento)**. Questo avviene **automaticamente senza che sia necessario specificarlo con il simbolo * nella definizione dei parametri formali della funzione e senza la necessità del simbolo & nei parametri attuali**. Questo avviene perché nel caso di un array ciò che viene passato come parametro è sempre l'indirizzo (in memoria) **del primo elemento dell'array**.

Importante è che quando si utilizza un array come parametro venga sempre passato anche un altro **parametro indicante la dimensione della array,** la dimensione dell'array è necessaria poter operare sull'array all'interno della funzione senza "sforare" con gli indici. Poiché il C non effettua alcun controllo sulla dimensione dell'array, è compito del programmatore porre attenzione a non scrivere valori al di fuori del limite massimo dell'array per non ottenere effetti imprevedibili.

- Il parametro **formale**, quando è un array, va sempre indicato seguito dalle parentesi quadre, ad esempio:

  Void cercaMinimo (**int vettore[ ],** int n)

  n è la dimensione dell'array (ribadisco **è opportuno che il programmatore lo passi sempre come parametro**)

- Al momento dell'invocazione della funzione all'interno del main, invece, il parametro **attuale**, anche se è un array, non prevede le parentesi quadre ma solamente il nome

  Minimo= cercaMinimo(**mmPioggia**,12);

OSSERVAZIONE IMPORTANTE

Non è possibile specificare un array come valore di ritorno di una funzione.

Esercizio: scrivere e testare una funzione “elementoMinimo” che individua e restituisce il valore minimo all’interno di un vettore di 10 elementi numeri interi. La dimensione del vettore va definita nel main utilizzando una costante. I valori del vettore vanno inseriti dall’utente (realizzare la richiesta di inserimento con un ciclo for nel main)

Esercizio codice a barre (corretto) (Vedi la soluzione nella seguente videolezione, il codice nella videolezione è C++ ma può comunque essere utile, utilizzare printf e scanf anziché cin e cout per le operazioni di input e output): https://www.youtube.com/watch?v=wPAqxt2Eghg&list=PL-NrWrNHrd0p8ColPTDZqb-LDUMy6pgFz&index=21)

26 I codici a barre di tipo GS1 sono composti da 13 cifre di cui l’ultima è la cifra di controllo che si determina a partire dalle prime 12 (il codice vero e proprio) con le seguenti regole:

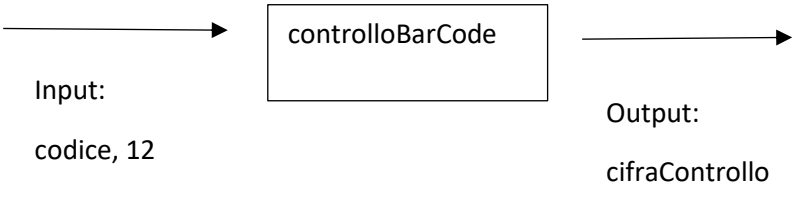
- moltiplicare per 1 tutte le cifre in posizione «dispari» (la prima, la terza, ... fino all’undicesima);
- moltiplicare per 3 tutte le cifre in posizione «pari» (la seconda, la quarta, ... fino alla dodicesima)[1];
- sommare i 12 valori così ottenuti;
- prendere il resto della divisione per 10 della somma ottenuta.
- **Da 10 sottrarre la cifra ottenuta al punto precedente**

Scrivere una funzione C/C++ che, a partire da un vettore di 12 numeri corrispondenti alle singole cifre di un codice a barre, restituisca la cifra di controllo calcolata con le regole illustrate. Scrivere un programma C++ che richieda all’utente l’inserimento delle singole cifre di un codice a barre e visualizzi la corrispondente cifra di controllo calcolata con la funzione realizzata in precedenza.

Vediamo due esempi:

9+8+8+7+9+0	41
(7+8+9+1+2+0)*3	27*3= 81
somma	122
x= resto divisione per 10	2
10 - x	10-2=8

8+3+0+9+0+0	20
(0+2+8+0+0+1)*3	33
somma	53
x= resto divisione per 10	3
10-x	7



Esercizio selection sort: ordinare gli elementi di un vettore con selection sort (vedi testo, spiegazione e soluzione nella videolezione: https://www.youtube.com/watch?v=c9gPwVkjDFo&list=PL-NrWrNHrd0p8ColPTDZqb-LDUMy6pgFz&index=19)