GLI ARRAY

Gli array

Gli array sono strutture dati statiche, di tipo sequenziale, che consentono la memorizzazione e la gestione di uno o più dati omogenei (dello stesso tipo) raggiungibili per mezzo di un indice.

Gli array

Per definire un array è necessario fornire:

- **Nome**
- **Tipo**

  Come per le altre variabili
- Dimensione

  Dopo la dichiarazione della variabile, non può essere modificata

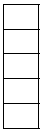
int v [5];

Gli array monodimensionali

Gli array monodimensionali o vettori sono quegli array per la cui definizione è sufficiente fornire un solo valore per la dimensione.

int v [5];

I dati vengono memorizzati in modo sequenziale, per facilitare l'accesso.



Elementi di un vettore

Per raggiungere gli elementi di un vettore è necessario precisare il nome del vettore e la posizione dell'elemento desiderato.

In C il primo elemento di un vettore si trova in posizione 0.

v[i]

↑

Indice (numero o espressione intera positiva)

Dati singoli che possono essere usati in espressioni o assegnazioni

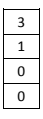
Assegnazione valori a vettore

Per assegnare i valori ad un vettore è possibile:

❖Assegnazione diretta

int v[4]={7,9,10,3};

int v[4]={3,1};



~~int v[2]={3,1,9};~~

int v[]={3,1};

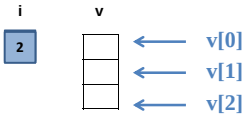
Meglio evitare

Assegnazione valori a vettore

Per assegnare i valori ad un vettore è possibile:

- Assegnazione diretta
- Acquisizione da tastiera
  E' necessario assegnare i valori alle singole componenti del vettore

```
int i, v [3];
for(i=0;i<3;i++)
    scanf("%d",&v[i]);
```

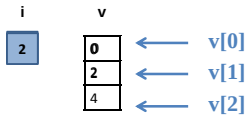


Assegnazione valori a vettore

Per assegnare i valori ad un vettore è possibile:

- Assegnazione diretta
- Acquisito da tastiera
- Successivo ad un calcolo

```
int i, v [3];
for(i=0;i<3;i++)
    v[i]=2*i;
```



Visualizzazione dei valori di un vettore

Per visualizzare i valori memorizzati in un vettore, è necessario scorrere le singole componenti e visualizzarle:

```
for(i=0;i<N;i++)
    printf("%d", v[i]);
```

N è la dimensione del vettore impostata in fase di dichiarazione.

Se nel programma si userà spesso N può essere utile definirla con una #define.

Copia dei valori di un vettore

Per copiare i valori di un vettore dentro un altro vettore è necessario:

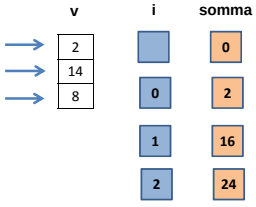
- che i due vettori abbiano la stessa dimensione oppure che si copi il vettore più piccolo in quello più grande.
- copiare le singole componenti in posizione corrispondente

```
for(i=0;i<N;i++)
    w[i]=v[i];
```

~~v=w;~~

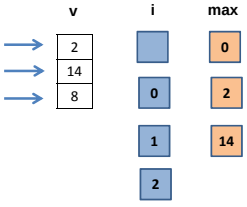
Somma degli elementi di un vettore

```
main()
{int v[3]={2,14,8};
int i, somma=0;
  for(i=0;i<3;i++)
      somma=somma+v[i];
  }
```



Ricerca dell'elemento massimo di un vettore

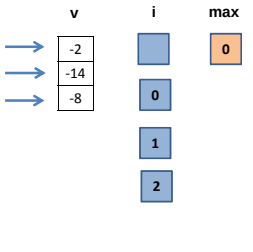
```
main()
{int v[3]={2,14,8};
int i, max=0;
  for(i=0;i<3;i++)
      if(v[i]>max)
          max=v[i];
  }
```



E se il vettore contiene numeri negativi?

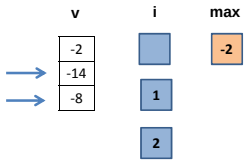
Ricerca dell'elemento massimo di un vettore

```
main()
{int v[3]={-2,-14,-8};
int i, max=0;
   for(i=0;i<3;i++)
       if(v[i]>max)
              max=v[i];
   }
```



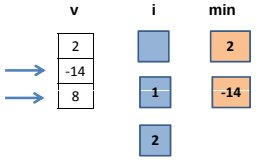
Ricerca dell'elemento massimo di un vettore

```
main()
{int v[3]={-2,-14,-8};
int i, max=v[0];
   for(i=1;i<3;i++)
       if(v[i]>max)
              max=v[i];
   }
```



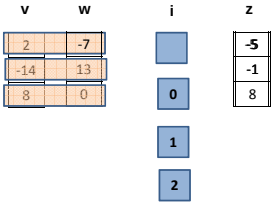
Ricerca dell'elemento minimo di un vettore

```
main()
{int v[3]={2,-14,8};
int i, min=v[0];
   for(i=1;i<3;i++)
       if(v[i]<min)
              min=v[i];
   }
```



Somma di due vettori

```
main()
{int v[3]={2,-14,8};
 int w[3]={-7,13,0};
 int i, z[3];
 for(i=0;i<3;i++)
     z[i]=v[i]+w[i];
 }
```



Questo è un esempio di uso di vettori paralleli. Si lavora su più vettori considerando gli elementi posti nella medesima posizione.

Vettori e puntatori

IL NOME DI UN VETTORE E' UN PUNTATORE ALLA SUA PRIMA POSIZIONE.

Aritmetica dei puntatori

I puntatori contengono indirizzi di memoria.

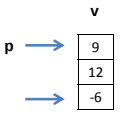
Le uniche operazioni possibili sono l'incremento ed i decremento per spostarsi in posizioni di memoria più alte o più basse.

Aritmetica dei puntatori

Ai puntatori si applicano i seguenti operatori:

+ - ++ --

```
main()
{int v[3]={9,12,-6},*p;
p=v;
p=p+2;
printf("%d",*p);}
```



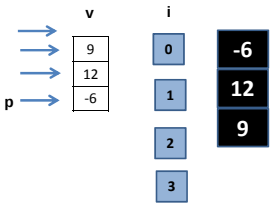
-6

Aritmetica dei puntatori

Ai puntatori si applicano i seguenti operatori:

\+ - ++ --

```
main()
{int v[3]={9,12,-6},*p,i=0;
p=&v[2];
for(;i<3;i++)
      printf(“%d”,*p--);}
```

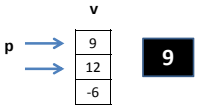


Aritmetica dei puntatori

Gli operatori ++ e – possono essere inclusi dentro altre istruzioni o espressioni.

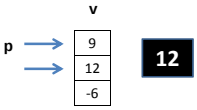
Se posti dopo la variabile agiscono dopo l’istruzione:

printf(“%d”,*p++);



Se posti prima della variabile agiscono prima dell’istruzione:

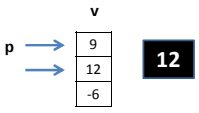
printf(“%d”,*++p);



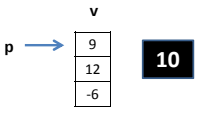
Aritmetica dei puntatori

ATTENZIONE ALL’ORDINE DEGLI OPERATORI

printf(“%d”, *++p);



printf(“%d”,++*p);



ATTENZIONE: Quando si usano gli operatori sui puntatori accertarsi che il risultato non sia una locazione di memoria vuota o non valida.

Passaggio di vettore a funzione

```
void fun(int a[10]);
main()
{int v[10];
…
fun(v);
…}
```

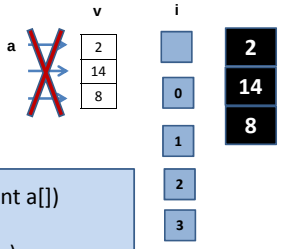
← La dimensione può essere omessa

```
fun(int a[10])
{…}
```

← Il nome di un vettore è un puntatore alla sua prima posizione, quindi un vettore verrà sempre passato per riferimento.

Passaggio di vettore a funzione

```
#define N 3
void stampa(int a[ ]);
main()
{int v[N]={2,14,8};
stampa(v);}
```

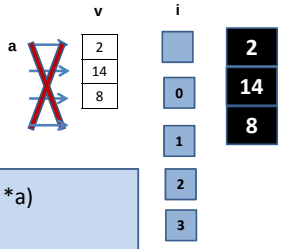



```
void stampa(int a[])
{int i;
for(i=0;i<N;i++)
      printf(“%d\n”,a[i]);}
```

Passaggio di vettore a funzione

Uso dei puntatori

```
#define N 3
void stampa(int *);
main()
{int v[N]={2,14,8};
stampa(v);}
```




```
void stampa(int *a)
{int i;
for(i=0;i<N;i++)
      printf(“%d\n”,*a++);}
```

Passaggio di vettore a funzione

Il passaggio del vettore ad una funzione avviene attraverso il nome del vettore che è un puntatore alla sua prima posizione.

Il vettore viene passato per indirizzo attraverso il passaggio per valore dell'indirizzo della sua prima posizione.

E' possibile passare vettori omettendo [] nel prototipo e nella definizione ma è consigliabile metterli per facilitare la lettura del codice.

Passaggio di vettore a funzione

In C il passaggio del vettore avviene necessariamente per riferimento.

Per evitare accidentali modifiche al contenuto di un vettore è possibile 'bloccarlo' usando il qualificatore **const** da inserire nel prototipo e nella definizione della funzione.

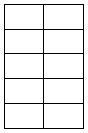
```
fun(const int [])

main()                  fun(const int a[])
{…                      {…}
fun(v);
…}
```

Gli array bidimensionali

Gli array bidimensionali o matrici sono quegli array per la cui definizione è sufficiente fornire due valori per la dimensione.

int m [5][2];

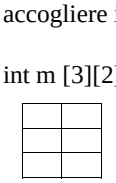


Gli array bidimensionali

Anche nelle matrici i dati vengono memorizzati in modo sequenziale per facilitare l'accesso.

Data una matrice di interi con N righe ed M colonne vengono allocate NxM posizioni di memoria adatte ad accogliere interi.

int m [3][2];



E' indispensabile specificare il numero di colonne per sapere quando stanno iniziando i dati della riga successiva.

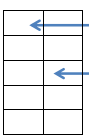
Elementi di una matrice

Per raggiungere gli elementi di una matrice è necessario precisare il nome della matrice e le coordinate dell'elemento desiderato.

In C il primo elemento di una matrice si trova in posizione 0,0.

~~m[i,j]~~ m[i][j]

Indici (numero o espressione intera positiva)



m[0][0]

m[2][1]

Dati singoli che possono essere usati in espressioni o assegnazioni

Assegnazione valori a matrice

Per assegnare i valori ad una matrice è possibile:

- Assegnazione diretta

```
int m[2][2]={{7,9},{10,3}};
int m[2][2]={7,9,10,3};
int m[][2]={{7,9},{10,3}};
```

7	9
10	3

~~int m[][];~~

```
int m[2][2]={{3,1}};
```

3	1
0	0

```
int m[2][2]={{0}};
```

0	0
0	0

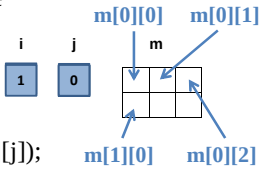
Assegnazione valori a matrice

Per assegnare i valori ad una matrice è possibile:

- Assegnazione diretta
- Acquisizione da tastiera
  E' necessario assegnare i valori alle singole componenti della matrice

```
int i,j, m[2][3];

for(i=0;i<2;i++)
      for(j=0;j<3;j++)
            scanf("%d",&m[i][j]);
```



Assegnazione valori a matrice

Per assegnare i valori ad una matrice è possibile:

- Assegnazione diretta
- Acquisito da tastiera
- Successivo ad un calcolo

```
int i, j, m[3][2];

for(i=0;i<3;i++)
      for(j=0;j<2;j++)
            m[i][j]=2*i+3*j;
```

0	3
2	5
4	7

Visualizzazione dei valori di una matrice

Per visualizzare i valori memorizzati in una matrice è necessario scorrere le singole componenti e visualizzarle:

```
for(i=0;i<N;i++)
      for(j=0;j<M;j++)
            printf("%d", m[i][j]);
```

In questo modo i valori verranno visualizzati in fila. E se volessi visualizzarli in forma tabulare?

Visualizzazione dei valori di una matrice

Per visualizzare i valori memorizzati in una matrice è necessario scorrere le singole componenti e visualizzarle:

```
for(i=0;i<N;i++)
      {for(j=0;j<M;j++)
            printf("%d ", m[i][j]);
      printf("\n");}
```

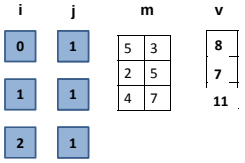
N è il numero di righe ed M il numero di colonne della matrice, impostati in fase di dichiarazione.

Può essere utile definire N ed M con una #define.

Esercizio

Data una matrice 3x2, calcolare e memorizzare la somma degli elementi di ciascuna riga:

```
main()
{int i=0, j=0;
 int m[3][2], v[3]={0};
…
for(i=0;i<3;i++)
      for(j=0;j<2;j++)
            v[i]=v[i]+m[i][j];
```



Matrici e puntatori

L'indirizzo base è quello dell'elemento di posto 0,0.

Data una matrice NxM, l'elemento di posto i,j si trova all'indirizzo &a[0][0]+M*i+j.

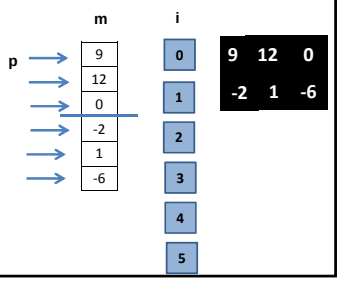
m[3][2]

&m[2][1]= &m[0][0] + 2*2 + 1

Matrici e puntatori

Stampa dei valori di una matrice mediante puntatore.

```
main()
{int m[2][3],*p, i=0;
p=&m[0][0];
for(;i<2*3;i++)
   {printf("%d ",*p++);
    if(i%3==2)
       printf("\n");}
```



Passaggio di matrice a funzione

```
void fun(int a[10][5]);
main()
{int m[10][5];
…
fun(m);
…}
```

La prima dimensione può essere omessa ma non la seconda.

```
fun(int a[10][5])
{…}
```

Anche le matrici vengono sempre passate per riferimento, quindi se non voglio apportare modifiche uso il qualificatore const.

Gli array multidimensionali

Il C consente di definire array con qualunque dimensione.