086180 - INFORMATICA APPLICATA

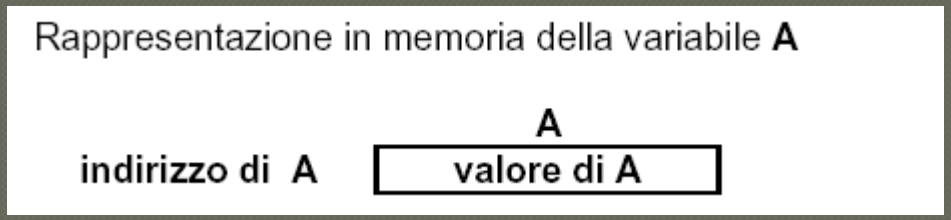
A.A. 2011-12

2° semestre

M07a – arrays

I dati semplici

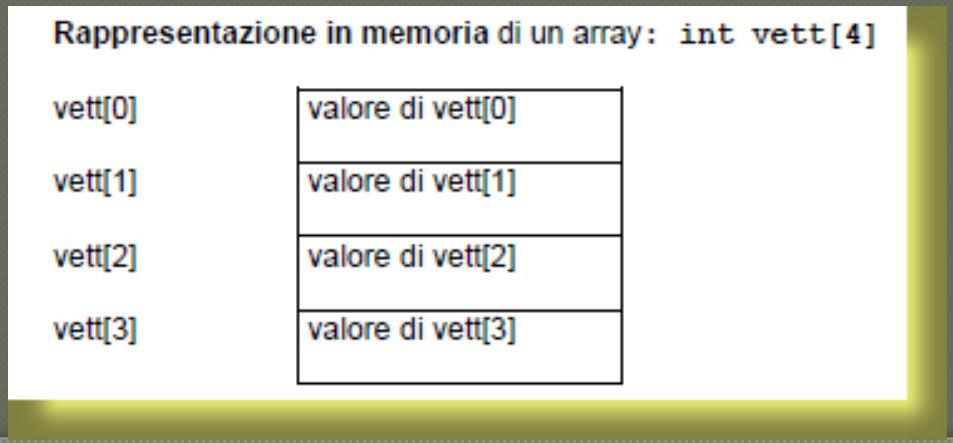
- I dati semplici sono gli elementi dove si opera singolarmente nelle espressioni.
- Ogni dato semplice occupa una singola posizione di memoria (più o meno ampia) ed è dotato di valore e indirizzo



- A indica di solito il valore della variabile
- Con &A si indica l'indirizzo di A

Aggregati di variabili

- È utile poter dare un nome a un insieme di variabili in modo da poterle elaborare collettivamente
- L'insieme di variabili può essere omogeneo o disomogeneo
- Un insieme di variabili ***omogeneo*** è chiamato array o vettore (o matrice)



Dettagli sugli array

- L'array si definisce ponendo le parentesi quadre dopo il nome di una variabile
- ```
  int  H; /*è una variabile semplice  */
  int K[10]; /* è un insieme ordinato
                di 10 elementi interi  */
  ```
- In definitiva K è l'indirizzo della prima cella di memoria
- L'indirizzo della seconda è K+1, della terza K+2 e così via
  ```

Dettagli sugli array segue

- `int K[10];`
- I dieci elementi che costituiscono l'array sono quindi i seguenti
- `K[0]  K[1] …….  K[9]`

- I dieci elementi che costituiscono l'array hanno quindi indirizzo
- `&K[0]  &K[1] …….  &K[9]`

- Ovvero
- `K+0  K+1  ……. K+9`

- Mentre l'indirizzo di tutto l'array è semplicemente `K`

Uso degli array

- Una volta dichiarato un array è possibile usare il valore alla posizione voluta, richiamandola tramite l'indice,
- ad esempio se volessi inserire il valore 87.43 nell'array di float alla quinta posizione, basta scrivere:
  - `float float_array[10];`
  - `float_array[4] = 87.43;`

Inizializzazione degli array

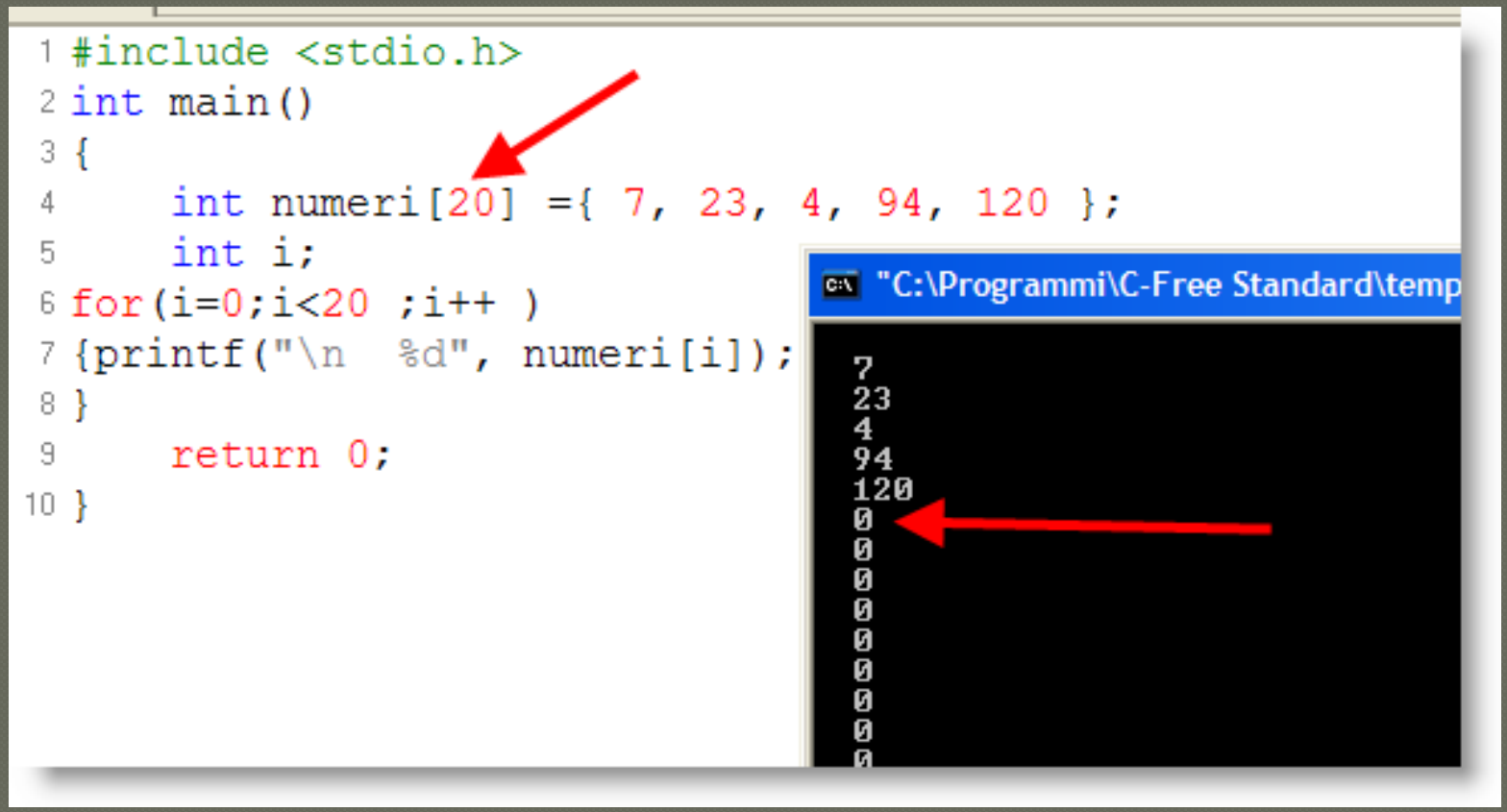
- Mediante un ciclo

```
#include <stdio.h>
int main(void)
{
int myarray[100];
int i;
for(i = 0; i < 100; i++)
{
myarray[i] = 1;
}
return(0);
}
```

Inizializzazione degli array

- Mediante una inizializzazione statica
  - `int numeri[] ={ 7, 23, 4, 94, 120 };`
- In questo modo si dichiara l'array, si inizializza e (N.B.) si determina anche il numero degli elementi
- Sarebbe diverso scrivere
  - `int numeri[20] ={ 7, 23, 4, 94, 120 };`

Inizializzazione incompleta



Uso degli array

- Non esistono operatori specifici per gli array
- Si può operare solo sugli elementi singoli
- **Non è possibile l'assegnamento**

```
int V[10], W[10];
V=W;    /* e` scorretto! */
```

Uso degli array

- Non è possibile leggere o scrivere un intero array (si opera sugli elementi)
- Per esempio per leggere un array

```
int V[100];
int i;
for(i=0; i<100; i++)
{ printf("\n valore %d-simo? ", i);
 scanf("%d", &V[i]);
};
```

Uso degli array

- **Per il resto è possibile applicare agli elementi di un array tutte le operazioni che si applicano sulle variabili semplici**

```
A[i] =n%i;
A[10]=A[0]+7;
scanf("%d", &A[i]);
If (A[i]<= 34) A[i] = A[i+3]*2;
```

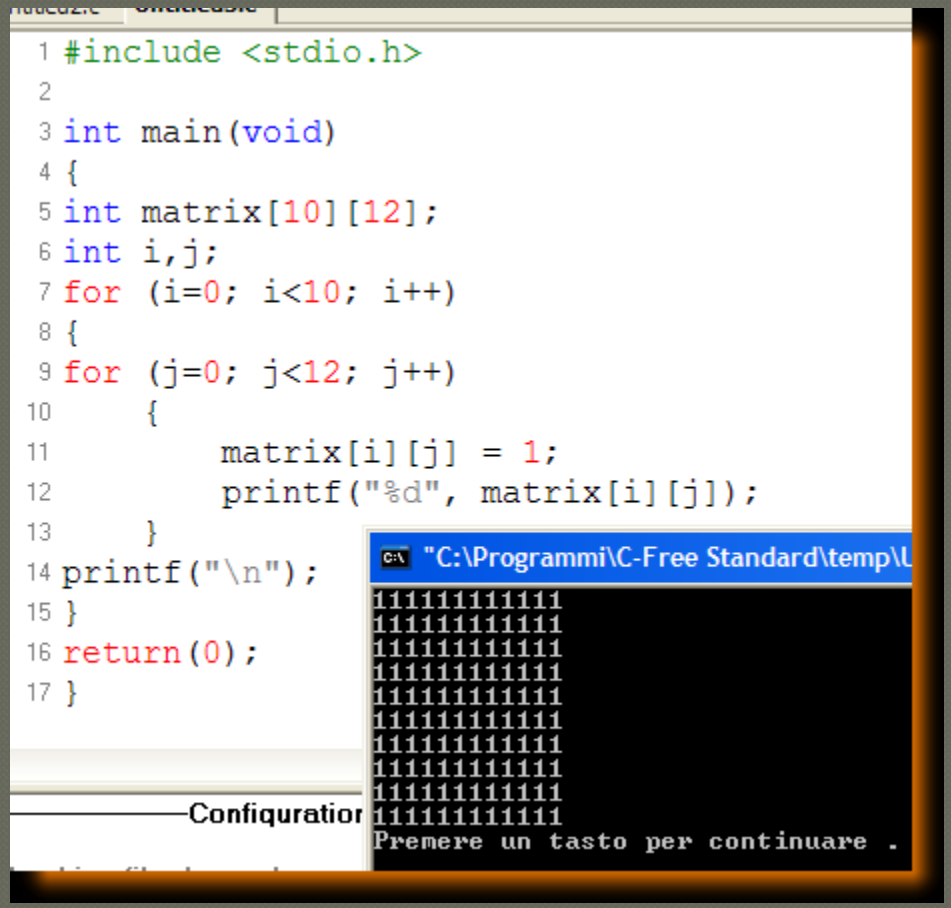
Costruttore di tipo

- È comodo poter assegnare un sinonimo ad una variabile strutturata in modo da non dover ripetere la definizione più volte
- Es
  - `typedef int Vettori[30];`
  - `Vettori V1,V2;`
- V1 e V2 sono dello stesso tipo come se si fosse scritto
  - `int V1[30], V2[30];`

Array multidimensionale

- Il costruttore array …. Ovvero [ ] … può essere applicato ad ogni tipo
- Quindi anche ad un array
- ***Int vet[10]*** [10]
- Ovvero la variabile in giallo è multiplata 10 volte
- In questo modo si costruiscono matrici (anche) n-dimensionali

Inizializzazione di una matrice bidimensionale



```c
#include <stdio.h>

int main(void)
{
int matrix[10][12];
int i,j;
for (i=0; i<10; i++)
{
for (j=0; j<12; j++)
    {
        matrix[i][j] = 1;
        printf("%d", matrix[i][j]);
    }
printf("\n");
}
return(0);
}
```

```
"C:\Programmi\C-Free Standard\temp\U
111111111111
111111111111
111111111111
111111111111
111111111111
111111111111
111111111111
111111111111
111111111111
111111111111
Premere un tasto per continuare .
```

Configuration

Somma di due vettori

```
6  int i;
7  /* lettura dei dati */
8  for(i=0; i<10; i++)
9  { printf("valore di A[%d] ", i);
10 scanf("%d", &A[i]);
11 }
12 for(i=0; i<10; i++)
13 { printf("valore di B[%d] ", i);
14 scanf("%d", &B[i]);
15 }
16 /* calcolo del risultato */
17 for(i=0; i<10; i++)
18 C[i]=A[i]+B[i];
19 /* stampa del risultato */
20 for(i=0; i<10; i++)
21 printf("C[%d]=%d\n",i, C[i]);
22 }
```

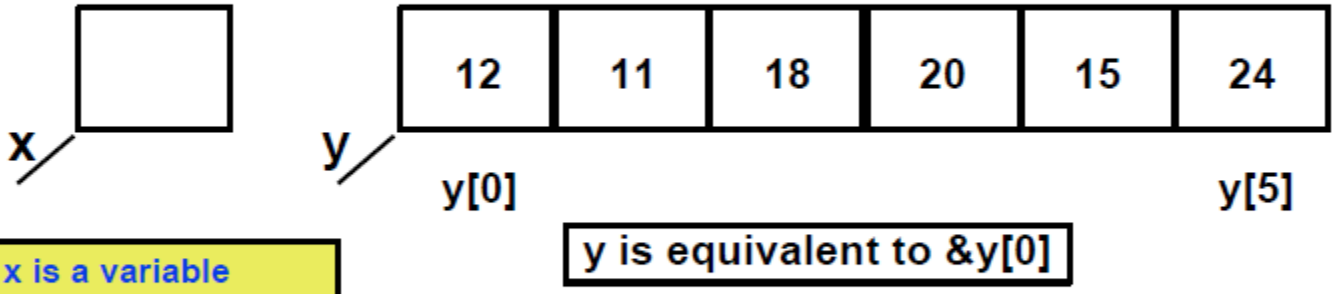
Configuration: mingw2.95 - CUI Debug, Build

```
"C:\Programmi\C-Free Sta
valore di A[0] 2
valore di A[1] 3
valore di A[2] 4
valore di A[3] 5
valore di A[4] 1
valore di A[5] 2
valore di A[6] 3
valore di A[7] 4
valore di A[8] 5
valore di A[9] 8
valore di B[0] 4
valore di B[1] 2
valore di B[2] 8
valore di B[3] 7
valore di B[4] 6
valore di B[5] 33
valore di B[6] 7
valore di B[7] 6
valore di B[8] 5
valore di B[9] 4
C[0]=6
C[1]=5
C[2]=12
C[3]=12
C[4]=7
C[5]=35
C[6]=10
C[7]=10
C[8]=10
C[9]=12
Premere un tasto per co
```

Riassunto

- **Array name is a constant**

  represents address of the 1st element; which is all you need to get to the other elements!

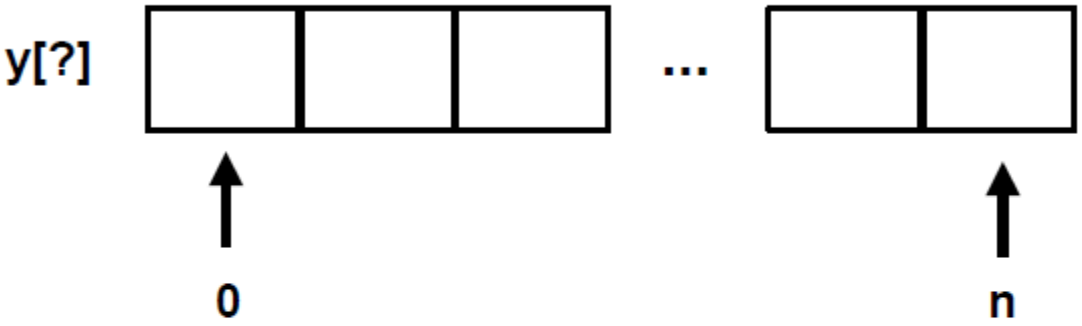


x is a variable
y is a constant
y is an address
&y[0] is an address
&y[0] is a constant
y[0] is a variable

```
x = 5      /* allowed; variable, not address*/
y[0] = 5   /* allowed; variable, not address*/
y = 18     /* address constant; not allowed */
```

L'uso di sizeof()

Usage with sizeof()



sizeof( ) returns number of BYTES in ENTIRE array

because array is a data structure

sizeof(array)/sizeof (char)	gives # of elements in array, if array of char (equal to sizeof(array), which is # of bytes)
sizeof(array)/sizeof(int)	gives # of elements in array, if array of int
sizeof(array)/sizeof(array[0])	gives # of elements in array, regardless of type!

inghe.c | matpermat.c | matpervet.c*

```c
#include <stdio.h>
#define N  10
#define M  10
int main()
{/*CALCOLO PRODOTTO MATRICE PER VETTORE (mat*vett=vris)*/
int vett[M], mat[N][M], vris[N];
int i,j;

for(i=0;i<=N-1;i++)
{
vris[i]=0;
for (j=0; j<=M-1;j++)
vris[i]=vris[i]+mat[i][j]*vett[j];
}
    return 0;
}
```

```c
#include <stdio.h>
#define N  10
#define M  10
#define K   5
int main()
{/*CALCOLO PRODOTTO MATRICE PER MATRICE (mat1*mat2=ris)*/
int mat1[N][M], mat2[M][K], ris[N][K];
int i,j,k;

for(i=0; i<=N-1; i++)
{
    for (k=0;k<=K; k++)
    {
    ris[i][k]=0;
    for (j=0; j<=M-1;j++)
    ris[i][k]=ris[i][k]+mat1[i][j]*mat2[j][k];
    }
};
    return 0;
}
```