UNIVERSITÀ
DI PARMA

C++ dati strutturati

dati strutturati

- strutture per trattare in modo unitario dati fra loro correlati
  - ***array*** : ripetizione di elementi omogenei
    - l'array è il meccanismo utilizzato per definire vettori e matrici
  - ***record***: giustapposizione di elementi (anche non omogenei)
    - il record è il meccanismo di base utilizzato negli archivi di dati (basi di dati):
      - ad esempio, per rappresentare le informazioni relative ad uno studente, quali la matricola, il nome, l'anno di corso, si usa una struttura record con i campi matricola, nome, …

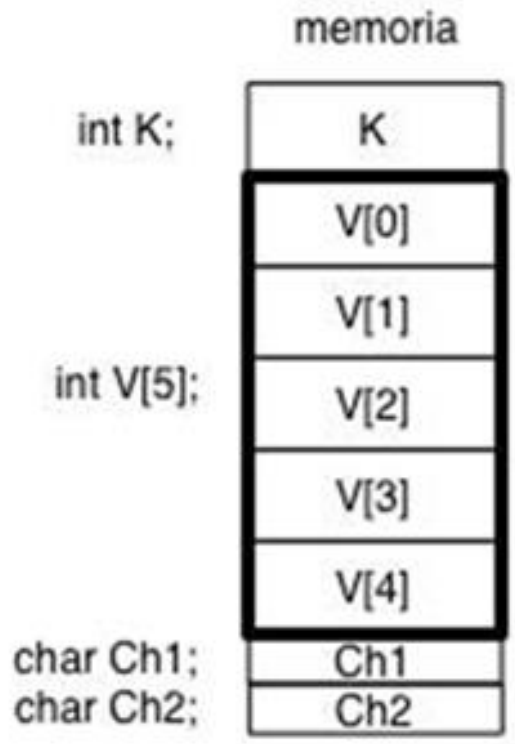
- ***vettori*** e ***matrici*** si dichiarano tramite il costruttore di tipo **[ ]** (costruttore array)

```
<var-array> ::= <tipo-elementi> <identif> <array>;
<array> ::= <costr-array> | <costr-array> <array>
<costr-array> ::= [<espres_costante_intera>]
```

- il valore **<espres_costante_intera>** stabilisce il numero degli elementi del vettore (dimensione del vettore)
- esempi

```
int V[10];              /* variabile V come vettore di 10 int */
int V1[10],V2[5];       /* variabili array V1 e V2 */
float M[10][10];        /* variabile M, matrice 10X10 float*/
```

vettore in memoria



`V[I]`: valore dell'elemento *I-esimo* di V, ovvero
per *accedere* agli elementi del vettore si utilizza **l'operatore [ ]**

`V` : nome del vettore

`I` : indice del vettore

indice: accesso agli elementi

- l'indice ***I*** è un'espressione con valore intero
- l'indice assume valori interi compresi tra 0 e dimensione-1
- ***attenzione*** : nell'esempio in figura l'espressione ***V[5]*** non comporta errore (né durante la compilazione né durante l'esecuzione) però V[5] utilizza erroneamente la stessa area di memoria riservata ad altre variabili (*Ch1 e Ch2 nell'esempio*)
- è buona pratica di programmazione utilizzare una costante simbolica per definire la lunghezza di un array:

```
#define N 100
int a[N];
```

inizializzazione

- inizializzazione di un array

```
int V[5] = {1,2,3,4,6};
int V[5] = {1,6}; /* solo i primi due elementi, gli altri hanno valore indefinito */
int V[5] = {1,2,3,4,6,8}; /* errore */
int V[] = {1,2,121 }; /* array di tre elementi */
```

- **`V[I]`** identifica una «variabile» che denota l'elemento I-esimo di **`V`**
- Assegnamento del valore K all'elemento I-esimo di V : **`V[I] = K;`**
- Lettura dell'elemento I-esimo di V : **`cin >> V[I];`**

vettori: memorizzazione

- i singoli elementi di un array (monodimensionale) sono memorizzati consecutivamente in memoria:

```
int A[100]={3,4,8};
```

- l'accesso ad un elemento avviene specificando l'indice

A[0]	3	0FFC
A[1]	4	1000
A[2]	8	1004
A[3]	0	1008
A[4]	0	100C
A[5]	0	1010
A[6]	0	1014
A[7]	0	1018
A[8]	0	101C

vettori: esercizi di esempio

- acquisire da input un vettore ***v1*** di ***n1*** numeri interi ***pari*** compresi tra ***a*** e ***b*** (estremi inclusi) [controllare l'input]
- determinare e stampare a video il valore massimo del vettore ***v1***; quindi determinare e stampare a video le posizioni del vettore ***v1*** che contengono tale valore
- acquisire da input un vettore ***v2*** di ***n2*** numeri interi ***dispari non decrescenti*** [controllare l'input]
- stampare a video il contenuto dell'array ***v2***

soluzione (1)

```
#include <iostream>
using namespace std;
int main() {
        const int n1 = 5; const int n2 = 8;
        int a,b,max; int v1[n1], v2[n2];
        cout << "valore inferiore: "; cin >> a;
        cout << "valore superiore: "; cin >> b;
        for (int i=0;i<n1;i++)
                do {
                        cout << "elemento di indice " << I << " : ";
                        cin >> v1[i];
                } while while (v1[i] < a || v1[i] > b || v1[i]%2 != 0);
…
```

soluzione (2)

```
…
max = v1[0];
for (int i=1;i<n1;i++)
   if (v1[i] > max)
      max = v1[i];
cout << "valore massimo: " << max << endl;
for (int i=0;i<n1;i++)
   if (v1[i] == max)
      cout << "l'elemento di indice " << i << " ha valore massimo" << endl;
```

soluzione (3)

```
do {
   cout << "Inserisci l'elemento di indice 0: ";
   cin >> v2[0];
} while (v2[0] % 2 == 0);
for (int i=1;i<n2;i++)
   do {
      cout << "Inserisci l'elemento di indice " << i << " : ";
      cin >> v2[i];
   } while (v2[i] % 2 == 0 || v2[i] < v2[i-1]);
for (int i=0;i<n2;i++)
   cout << "v2[" << i << "] = " << v2[i] << endl;
```

array multidimensionali

- gli array multidimensionali possono essere considerati array di array di array …

```
int a[100];          /* array monodimensionale */
int b[2][7];         /* array bidimensionale */
int c[5][3][2];      /* array tridimensionale */
```

- gli array multidimensionali sono memorizzati in sequenza lineare ma nel caso di array bidimensionali (matrici) risulta utile pensarli organizzati in righe e colonne

```
int a[3][5];
```

	col.1	col.2	col.3	col.4	col.5
riga 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
riga 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
riga 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]

inizializzazione

- esistono vari modi, tra loro equivalenti:

```
int a[2][3] = {1,2,3,4,5,6};
int a[2][3] = {{1,2,3},{4,5,6}};
int a[ ][3] = {{1,2,3},{4,5,6}};
```

- se non vengono inizializzati i valori presenti nell'array sono impredicibili, è possibile ottenere velocemente l'azzeramento in questo modo:

```
int a[2][2][3] = {0};
```

matrice in memoria

```
#include <iostream>
using namespace std;
int main() {
	int m[5][4];
	int *pm;
	pm = &m[0][0];
	for (int r=0;r<5;r++)
		for (int c=0;c<4;c++)
			m[r][c] = 10*r+c;
	for (int i=0;i<5*4;i++) {
		cout << "indirizzo " << pm << " valore " << *pm << endl;
		pm++;
	}
}
```

```
indirizzo 0x6dfea0 valore 0
indirizzo 0x6dfea4 valore 1
indirizzo 0x6dfea8 valore 2
indirizzo 0x6dfeac valore 3
indirizzo 0x6dfeb0 valore 10
indirizzo 0x6dfeb4 valore 11
indirizzo 0x6dfeb8 valore 12
indirizzo 0x6dfebc valore 13
indirizzo 0x6dfec0 valore 20
indirizzo 0x6dfec4 valore 21
indirizzo 0x6dfec8 valore 22
indirizzo 0x6dfecc valore 23
indirizzo 0x6dfed0 valore 30
indirizzo 0x6dfed4 valore 31
indirizzo 0x6dfed8 valore 32
indirizzo 0x6dfedc valore 33
indirizzo 0x6dfee0 valore 40
indirizzo 0x6dfee4 valore 41
indirizzo 0x6dfee8 valore 42
indirizzo 0x6dfeec valore 43
```