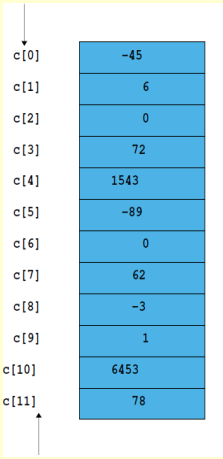
I vettori

- **Vettore** (Array monodimensionale)
  - Sequenza di **posizioni consecutive** (o locazioni di memoria) che vengono chiamate **elementi** del vettore
  - Gli elementi hanno tutti lo **stesso nome** e sono tutti dell **stesso tipo di dato** (**omogenei**)

- Per fare riferimento ad un elemento, specificare
  - il **nome** del vettore
  - numero di posizione (o ***indice***)

- Formato: ***nomevettore*[*numeroposizione*]**
  - il primo elemento è alla posizione **0** e l'ultimo alla **n-1**
  - se **c** è un vettore di ***n*** elementi: **c[0], c[1]…c[n-1]**
    (nell'esempio **c** ha **12 elementi** quindi **l'indice** va da **0** a **11**)

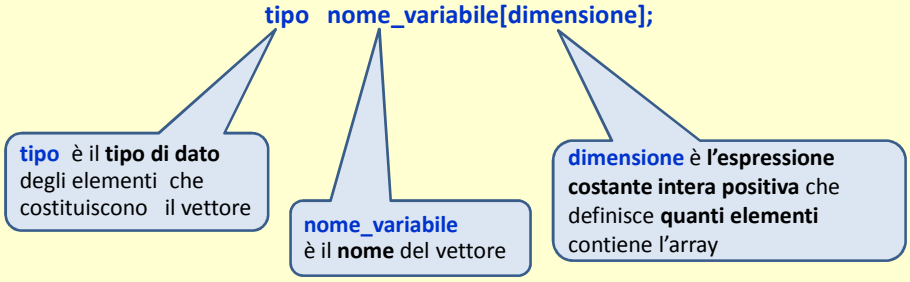
Nome del vettore (tutti gli elementi hanno lo stesso nome, **c**, e un indice che li identifica)



numero di posizione (*indice*) all'interno del vettore **c**

Dichiarazione di un vettore in C++:

tipo nome_variabile[dimensione];



Quando l'array è dichiarato, il **compilatore alloca un blocco** di memoria per contenere l'array.
Per un array monodimensionale la dimensione totale in byte occupata viene calcolata in questo modo:

Byte totali = **n. di byte** (associati al tipo) **x** **n. di elementi**

Esempi di dichiarazioni

```
int A[10];     // Variabile di nome A, di tipo vettore di valori interi, di dimensione 10
                  (10 elementi)
float B[20];   // variabile di nome B, di tipo vettore di valori reali, di dimensione 20
                  (20 elementi)
char S[32];    // variabile di nome S, di tipo vettore di caratteri, di dimensione 32
                  (32 elementi)

const int NELEM = 30;          // dichiarazione di una costante di tipo intero
float X[NELEM ];               // variabile di nome X, di tipo vettore di valori reali, di
                                  dimensione NElem (espressione costante)
int Y[NELEM * 2];              // variabile di nome Y, di tipo vettore di valori interi, di
                                  dimensione NElem*2 (espressione costante)
```

In alternativa la **costante** si può dichiarare con **#define**:

```
#define NELEM 30;              // dichiarazione di una costante di tipo intero #define in
                                  genere viene scritta subito dopo le istruzioni #include
float X[NELEM];                // variabile di nome X, di tipo vettore di valori reali, di
                                  dimensione NElem (espressione costante)
int Y[NELEM * 2];              // variabile di nome Y, di tipo vettore di valori interi, di
                                  dimensione NElem*2 (espressione costante)
```

Dichiarazione delle costanti

Per convenzione, le costanti sono indicate da nomi TUTTI_MAIUSCOLI

- **Il costrutto** **#define**
  - Metodo originario, in tutte le versioni del C
  - Definisce costanti valide su tutto il file
  - **Non specifica il tipo** della costante e tra il nome e il valore non compare il segno =
  - Le istruzioni #define devono essere una per riga

- **Il modificatore** **const**
  - Metodo più moderno, nelle versioni recenti del C
  - Usa la stessa sintassi di definizione delle variabili
  - Specifica il **tipo** della costante e la definizione è terminata da **;**
  - Il valore di NELEM non si può più cambiare

Il compilatore si occuperà di sostituire, ad ogni occorrenza del nome della costante, il suo valore numerico.

Esempi di dichiarazioni

Per dichiarare più vettori dello stesso tipo si scrivono i nomi sulla stessa riga separati da virgola, come si fa per le variabili:

```
int b[ 100 ], x[ 27 ];
```

Gli elementi di un vettore possono essere utilizzati come le altre variabili

```
c[0] = 3;          // si assegna il valore 3 al primo elemento del vettore c
cout << c[0];      // si visualizza il contenuto del primo elemento del vettore c
```

E' possibile eseguire operazioni sull'indice

```
j=8;
c[j-2] = 3;        // si assegna il valore 3 all'elemento c[8-2] cioè a c[6]
```

Inizializzazione di un vettore

Ci sono 2 forme per inizializzare un vettore come per tutte le altre variabili:

- **In fase di dichiarazione:**

  tipo nome_variabile[ ]={e1,…,en};

- **tramite indicizzazione diretta all'interno del programma quando occorre**, ad es:

```
int a[2];       /* dichiarazione vettore */
...
a[0]=23;        /* inizializzazione vettore */
a[1]=100;
```


Attenzione!

- **Inizializzare un vettore:** **int Vet[8] = {15, 2, 13, 4, 9 };**

15	2	13	4	9	0	0	0

Se gli *inizializzatori non sono sufficienti, gli* elementi rimanenti sono **azzerati**

Se sono troppi → **errore di sintassi** **int Vet[3] = {15, 2, 13, 4, 9 };**

Scrivendo: **int Vet[5] = {0};** // **si azzerano tutti gli elementi**

In C++ non ci sono controlli sul superamento a run time della dimensione di un vettore per esempio in un vettore di 20 elementi l'assegnamento :

Vet[25] =12; // **non viene segnalata come un errore**

Se la dimensione non è dichiarata gli inizializzatori la determinano

int Vet[ ] = { 15, 2, 13, 4, 9 }; **5 inizializzatori, quindi il vettore ha 5 elementi**

In **C++ non è possibile assegnare un vettore ad un altro.**

Pertanto scrivere così è ILLEGALE:

```
int A[10], B[10];
A = B;                    /* ERRORE!!! */
```

Per trasferire il contenuto di un array in un altro è necessario **assegnare individualmente ogni valore** per esempio utilizzando un ciclo **for.**

```
for (int i=0; i<10; i++)
    A[i]= B[i];
```


Uso di un elemento di un vettore

L'elemento di un vettore è utilizzabile come una qualsiasi variabile:

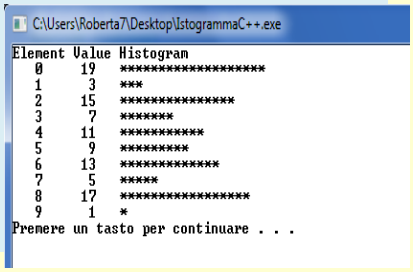
- utilizzabile all'interno di un'espressione: **Tot = Tot + V[i] ;**
- utilizzabile in istruzioni di assegnazione: **V[0] = 0 ;**
- utilizzabile per stampare il valore: **cout << Vet[k] ;**
- utilizzabile per leggere un valore: **cin >> Vet[j] ;**

Altri esempi:

```
if (dato[i]==0)          // se l'elemento contiene zero
if (dato[i]==dato[i+1])  // due elementi consecutivi uguali
dato[i] = dato[i] + 1 ;  // incrementa l'elemento i-esimo
dato[i] = dato[i+1] ;    // copia un dato dalla cella successiva
```

Esempio

```
/* Histogram printing program */
#include <iostream>
#include <iomanip>
using namespace std;
#define SIZE 10
int main()
{ int A[ SIZE ] = { 19, 3, 15, 7, 11, 9, 13, 5, 17, 1 };
  int i, j;
  cout<< "Element\t"<< "Value "<< "Histogram\n" ;
  for ( i = 0; i <= SIZE - 1; i++ )
    { cout<<setw(4)<< i<< setw(7)<<n[ i ]<<"   " ;
      for ( j = 1; j <= A[ i ]; j++ )     // print one bar
         cout<< '*';
     cout<<endl;
    }
  system("pause");
  return 0;
}
```



Operazioni di base sui vettori

- **Stampa di un vettore**

  Occorre stampare un elemento per volta, all'interno di un ciclo **for**

  Ricordare che:

  - gli indici del vettore variano tra 0 e N-1, se N è il numero degli elementi
  - V[i] è l'elemento (i+1)-esimo

```
// stampa degli elementi di un vettore uno sotto l'altro
for  (int i=0; i<N ; i++)
      cout << V[i]<<endl;

// stampa degli elementi di un vettore sulla stessa riga distanziati
for  (int i=0; i<N ; i++)
      cout << V[i]<<"\t";
```

Operazioni di base sui vettori

- **Lettura di un vettore da tastiera**

  Occorre leggere un elemento per volta, all'interno di un ciclo **for**

```
// lettura degli elementi di un vettore
 for  (int i=0; i<N ; i++)
      {cout <<"Inserisci l'elemento di posizione  "<<i<<" :";
       cin >> V[i];
      }
```

- **Copia di un vettore in un altro**

  Si tratta si copiare il contenuto di un vettore in un altro vettore.

  Occorre copiare un elemento per volta dal vettore "sorgente" al vettore "destinazione", all'interno di un ciclo **for**

  I due vettori devono avere lo stesso numero di elementi, ed essere dello stesso tipo base

```
// copia il contenuto di ciascun elemento  di V nel corrispondente elemento di W
for  (int i=0; i<N ; i++)
     W[i] = V[i] ;
```