**variabili di tipo *int***

```
#include <iostream>
using namespace std;

int main(){
    int *pa;
    pa = new int;
    cout << "pa= " << pa << endl;
    pa++;
    cout << "pa= " << pa << endl;
    pa -= 2;
    cout << "pa= " << pa << endl;

    int *p1, *p2, diff;
    p1 = new int;
    p2 = new int;
    cout << "p1= " << p1 << endl;
    cout << "p2= " << p2 << endl;
    diff = p2 - p1;
    cout << "p2-p1= "<< diff <<endl;

    delete pa, p1, p2;
}
```

**Output | commenti**

**Esecuzione 1**

```
pa= 0x1da068
pa= 0x1da06c
pa= 0x1da064
p1= 0x1d9fb8
p2= 0x1da078
p2-p1= 48
```

**pa++**
**pa -= 2**
$9FB8_h$ = 40888
$A078_h$ = 41080
interi tra p2 e p1 : **48** = (41080-40888)/4 = 192/4

Le aree di memoria allocate dinamicamente NON sono consecutive.

Ad ogni **esecuzione** le aree di memoria possono essere diverse (**0x1da068** e poi **0x849fe8**)

**Esecuzione 2**

```
pa= 0x849fe8
pa= 0x849fec
pa= 0x849fe4
p1= 0x849ff8
p2= 0x849f88
p2-p1= -28
```

28 interi tra p1 e p2

**variabili di tipo *int***

```
#include <iostream>
using namespace std;
int main(){
    int *pi, n1, n2;
    pi = new int;
    *pi = 33;
    n1  = 66;
    n2 = *pi + n1;
    cout  << "pi = "  << pi  << endl;
    cout << "&n1= " << &n1 << endl;
    cout << "&n2= " << &n2 << endl;
    cout << "*pi= " << *pi << endl;
    cout  << " n1= "  << n1  << endl;
    cout  << " n2= "  << n2  << endl;
    delete pi;
}
```

**Output | commenti**

```
pi = 0xcb9fd0
&n1= 0x6dfed0
&n2= 0x6dfecc
*pi= 33
 n1= 66
 n2= 99
```

n1 ed n2 sono allocate staticamente e sono consecutive (n2 indirizzo più basso, n2 più alto)

D0 è 4byte dopo CC

← 33 valore inserito nella zona di memoria di tipo intero e puntato da **pi**

**_Array_ di tipo _int_**

```
#include <iostream>
using namespace std;

int main(){

    int *p5i;
    p5i  = new int[5];

    p5i[0] = 1;      p5i[1] = 3;
    p5i[2] = 5;      p5i[3] = 7;
    *(p5i+4) = -11;

    cout << "p5i = " << p5i << endl;
    cout << "*p5i = "<< *p5i<< endl;

    for (int x=0; x<5; x++) {
        cout << "pi["<< x <<"]= ";
        cout << *(p5i+x) << endl;
    }
    cout << "p5i= " << p5i  << endl;
    cout<< "*p5i = "<< *p5i << endl;

    delete p5i;
}
```

**Output | commenti**

**Esecuzione 1**

```
p5i = 0xcb6d10
*p5i = 1
pi[0]= 1
pi[1]= 3
pi[2]= 5
pi[3]= 7
pi[4]= -11
p5i= 0xcb6d10
*p5i= 1
```

per indicare un elemento di un array allocato dinamicamente mediante il puntatore **p5i** si usa la notazione vettoriale **p5i[4]** o l'equivalente:

***(p5i+4) = -11;**

**output**

**Esecuzione 2**

```
p5i= 0x1d6d10
*p5i = 1
pi[0]= 1
pi[1]= 3
pi[2]= 5
pi[3]= 7
pi[4]= -11
p5i= 0x1d6d10
*p5i= 1
```

***Array* di tipo *int* a dimensione variabile dim**

```
#define TAB '\t'
#define ASK "digita dimensione array > 2: "
#define INT "\nV[k]  vett[k]   pVi[k]  *(pVi+k)"
int main(){
    int V [5] = { -2 , -1, 0, 1, 2};
    int dim;
    do { cout << ASK;          cin  >> dim;
       } while (dim <= 2);
    int vett [dim];
    int *pVi;
    pVi  = new int[dim];
    cout << "&V    = "    << V    << endl;
    cout << "&vett= "    << vett << endl;
    cout << "pVi   = "    << pVi  << endl;
    for (int i=0; i<dim-2; i++) {
        vett[i]  =   i;
        pVi [i]  = 7-i;
    }
    for (int k=dim-2; k<dim; k++) {
        vett[k]    = 7-k;
        *(pVi+k) =   k;
    }
    cout << INT << endl;
    for (int k=0; k <= dim; k++) {
     cout << V[k]     << TAB << vett[k]    << TAB;
     cout << pVi [k]  << TAB << *(pVi+k) << endl;
    }   //
    cout << "\npVi    = " << pVi     << endl;
    cout << "++pVi = "    << ++pVi  << endl;
    cout << "pVi    = "    << pVi     << endl;
    cout << "pVi++ = "    << pVi++  << endl;
    cout << "pVi    = "    << pVi     << endl;
    cout << "pVi+1 = "    << pVi+1   << endl;
    cout << "pVi    = "    << pVi     << endl;
    delete pVi;
}
```

**output**

Il programma C++ utilizza 3 vettori:

- **V** [5] allocato staticamente al caricamento in Memoria Centrale
- **vett** [dim] allocato dinamicamente, a *runtime,* dopo lettura di **dim**
- ***(pVi+k)** allocato dinamicamente mediante puntatore **pVi**

```
digita dimensione array > 2: 6
&V    = 0x6dfeb4
&vett= 0x6dfe88
pVi   = 0x776d18

V[k]   vett[k]    pVi[k]      *(pVi+k)
-2        0         7         7
-1        1         6         6
0         2         5         5
1         3         4         4
2         3         4         4
7826712   2         5         5
7208584   7208624   -17288384          -17288384

pVi    = 0x776d18
++pVi = 0x776d1c
pVi    = 0x776d1c
pVi++ = 0x776d1c
pVi    = 0x776d20
pVi+1 = 0x776d24
pVi    = 0x776d20
```

Il programma C++ utilizza:

una **struttura X** allocata staticamente al caricamento del programma in Memoria Centrale e un'altra allocata dinamicamente mediante puntatore **pTipi**

| *Struct* | output |
|---|---|

```
#include <iostream>
using namespace std;
#define ENDL2 "\n\n"

struct Tipi {
  unsigned short int  USI;  //--  2 bytes
  short int           SI;   //--  2 bytes
  unsigned int        UI;   //--  4 bytes
  int                 I;    //--  4 bytes
  float               F;    //--  4 bytes
};                          //-- 16 bytes totali

int main(){
  int  ultimo = 9999;

  unsigned short int*   pUSI;
  short int*            pSI;
  unsigned int*         pUI;
  int*                  pI;
  float*                pF;
  Tipi*                 pTipi;

  Tipi X;
  int  primo = 1111;

    cout << "sizeof :\n";
    cout << "--USI  : " << sizeof(unsigned short int) <<endl;
    cout << "--SI   : " << sizeof(short int)          <<endl;
    cout << "--UI   : " << sizeof(unsigned int)       <<endl;
    cout << "--int  : " << sizeof(int)                <<endl;
    cout << "--float: " << sizeof(float)              <<endl;
    cout << "--Tipi : " << sizeof(Tipi)               <<ENDL2;
    cout << "--pSI  : " << sizeof(pSI)                <<endl;
    cout << "--pTipi: " << sizeof(pTipi)              <<endl;
```

```
sizeof :
--USI  : 2
--SI   : 2
--UI   : 4
--int  : 4
--float: 4
--Tipi : 16

--pSI  : 4
--pTipi: 4

indirizzo di:
- primo: 0x6dfec0
- X    : 0x6dfec4
- pTipi: 0x6dfed4
- pF   : 0x6dfed8
- pI   : 0x6dfedc
- pUI  : 0x6dfee0
- pSI  : 0x6dfee4
- pUSI : 0x6dfee8
-ultimo: 0x6dfeec

- pTipi: 0x6dfec4
X.USI  : 0
X.SI   : -32768
X.UI   : 0
X.int  : -2147483648
X.float: 2.5

- pTipi: 0xc16d10
D.USI  : 65535
D.SI   : 32767
D.UI   : 4294967295
D.int  : 2147483647
D.float: 1.5
```

***Struct***

```
    cout << "\nindirizzo di:\n";
    cout << "- primo: "  << &primo   <<endl;
    cout << "- X    : "  << &X       <<endl;
    cout << "- pTipi: "  << &pTipi   <<endl;
    cout << "- pF   : "  << &pF      <<endl;
    cout << "- pI   : "  << &pI      <<endl;
    cout << "- pUI  : "  << &pUI     <<endl;
    cout << "- pSI  : "  << &pSI     <<endl;
    cout << "- pUSI : "  << &pUSI    <<endl;
    cout << "-ultimo: "  << &ultimo  <<ENDL2;

    X.USI = 65536;          // da            0  a  65535
    X.SI  = 32768;          // da       -32768  a +32767
    X.UI  = 4294967296;     // da            0  a  4294967295
    X.I   = 2147483648;     // da -2147483648  a +2147483647
    X.F   = 2.5;

    pTipi = &X;
    cout << "- pTipi: " << pTipi   <<endl;
    cout << "X.USI  : " << pTipi -> USI <<endl;
    cout << "X.SI   : " << pTipi -> SI  <<endl;
    cout << "X.UI   : " << pTipi -> UI  <<endl;
    cout << "X.int  : " << pTipi -> I   <<endl;
    cout << "X.float: " << pTipi -> F   <<ENDL2;

    pTipi = new Tipi;
    cout << "- pTipi: "  << pTipi   <<endl;

    pTipi -> USI = X.USI - 1;
    pTipi -> SI  = X.SI  - 1;
    pTipi -> UI  = X.UI  - 1;
    pTipi -> I   = X.I   - 1;
    pTipi -> F   = X.F   - 1;

    cout << "D.USI  : " << pTipi -> USI <<endl;
    cout << "D.SI   : " << pTipi -> SI  <<endl;
    cout << "D.UI   : " << pTipi -> UI  <<endl;
    cout << "D.int  : " << pTipi -> I   <<endl;
    cout << "D.float: " << pTipi -> F   <<ENDL2;

    delete pTipi;
}
```

**Output**

Tutti i puntatori occupano 4 byte indipendentemente dal tipo cui fanno riferimento.

La struttura Tipi è composta da campi numerici di vario tipo; la dimensione complessiva è di 16 byte.

Per far riferimento ad un campo di un dato strutturato allocato **staticamente** la sintassi è:

**nomeDato .** nomeCampo

**← valori minimi e massimi per i tipi utilizzati**

ATTENZIONE: assegnando alle variabili il numero successivo al massimo rappresentabile, si torna al minimo valore, circolarmente

Per far riferimento ad un campo di un dato strutturato utilizzando un puntatore oppure allocandolo **dinamicamente** la sintassi è:

**nomePuntatore ->** nomeCampo