



UNIVERSITÀ
degli STUDI
di CATANIA

Puntatori in C++

Corso di programmazione I AA 2019/20

Corso di Laurea Triennale in Informatica

Prof. Giovanni Maria Farinella

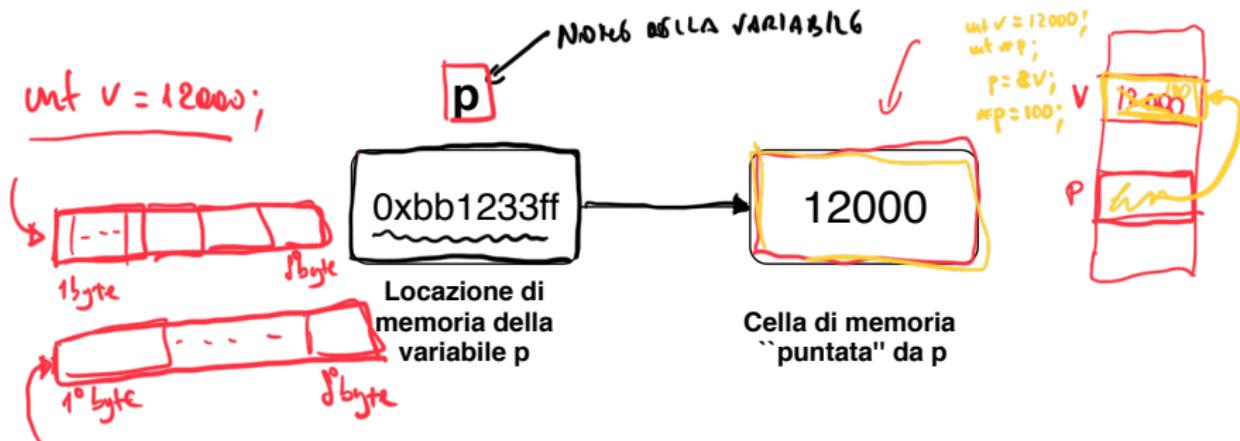
Web: <http://www.dmi.unict.it/farinella>

Email: gfarinella@dmi.unict.it

Dipartimento di Matematica e Informatica

Cosa è una variabile puntatore

Variabile Puntatore: variabile che contiene un indirizzo di memoria.



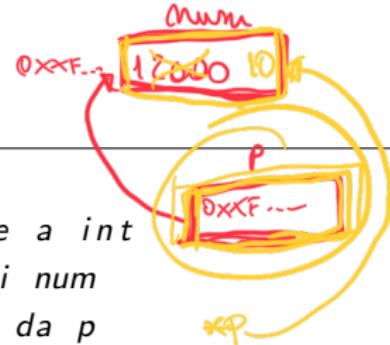
Si dice che una variabile puntatore “punta” ad un certo dato, in quanto contiene il suo **indirizzo in memoria**.

Cosa è una variabile puntatore

Un puntatore permette una differente modalità di accesso alle locazioni di memoria del calcolatore.

Dichiarazione di una variabile puntatore

```
1 int num = 12000;  
→ 2 int *p; // dichiara p come puntatore a int  
3 p = &num; // assegna a p indirizzo di num  
→ 4 *p = 10; // modifica il dato puntato da p
```



num = 10;

Il carattere * anteposto al nome della variabile è denominato operatore di **dereferenziazione** o **indirezione**. Viene usato per

- definire variabili puntatore (linea 2);
- modificare il dato puntato dal puntatore stesso (linea 4)

Cosa è una variabile puntatore

```
1 int num;  
2 int *p; // dichiara p come puntatore a int  
3 p = &num; // assegna a p indirizzo di num  
4 *p = 10; // modifica il dato puntato da p
```

Il carattere **&** è denominato operatore di **referenziazione** o “**address of**”.

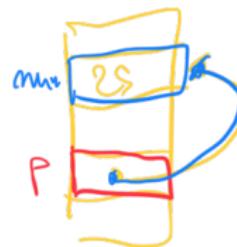
Esso viene anteposto al nome di una variabile per “estrarre” l’indirizzo di memoria di una certa variabile.

Cosa è una variabile puntatore

Dichiarazione puntatore e contestuale inizializzazione

Si dichiara la variabile p come puntatore ad un certo tipo (ES: int) e assegna ad esso l'indirizzo di una certa variabile (ES: num).

```
1 int num = 25;      ←  
2 int *p = &num;  
3 cout << p << endl; ←  
4 cout << *p << endl; ←
```

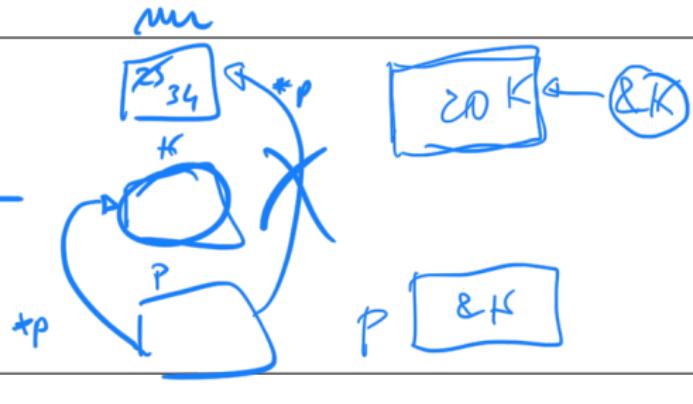


La linea 3 stampa a video un numero in **formato esadecimale** (INDIRIZZO della cella di memoria, ES: 0x112233aa).

La linea 4 stampa a video il **valore contenuto nella variabile num** (il DATO), ovvero 25.

Cosa è una variabile puntatore

```
1 int num = 25; ←  
2 int k = 20; ←  
3 int *p = &num; ←  
4 *p = 34; ←  
5 p = &k ←
```



\cancel{p}

Il puntatore p, in quanto variabile non costante, può essere modificato mediante riassegnamento di altri indirizzi di memoria (linea 5).

Cosa è una variabile puntatore

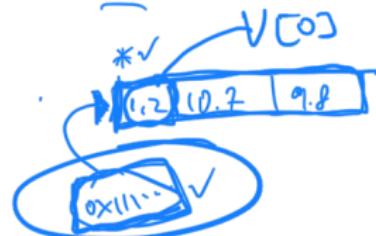
Esempi svolti

17_01.cpp



Array vs puntatori

```
1 double v[] = {1.2, 10.7, 9.8};  
2 cout << v; //stampa un indirizzo, ES: 0x11223344  
3 cout << *v; //stampa 1.2  
4 cout << v[0]; //stampa 1.2  
5  
6 double w[] = {3.4, 6.7, 9.8};  
7 v = w; //Errore di compilazione!
```



Il nome di una variabile array è un **puntatore costante** al primo elemento dello array.

v può essere usata in **espressioni che fanno uso di aritmetica dei puntatori**, ma indirizzo contenuto in `v` non è modificabile.

Array vs puntatori

~~Diff~~

ptr[1]

```
1 double v [] = {1.2, 10.7, 9.8};  
2 double *ptr = v;  
3 cout << ptr [1]; //stampa 10.7  
4 cout << ptr [2]; //stampa 9.8
```

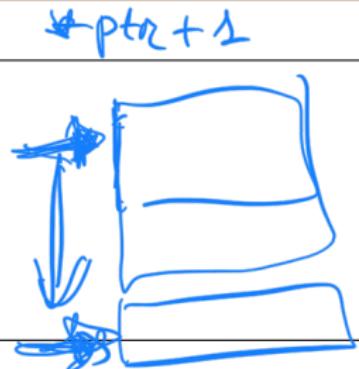
Viceversa, le variabili che sono **puntatori si possono usare mediante indici come gli array.**

Aritmetica dei puntatori

```
1 double v [] = {1.2, 10.7, 9.8};  
2 double *ptr = v; //ptr + 1  
3 cout << *(ptr + 1); //stampa 10.7  
4 cout << *(ptr + 2); //stampa 9.8  
5 cout << *(v + 2); //stampa 9.8
```

VC2

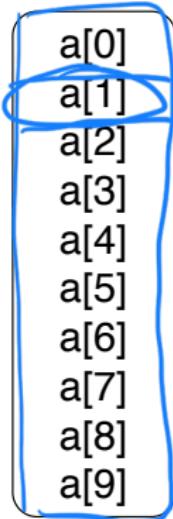
V+2



Se ptr è un puntatore, mediante la espressione $(\text{ptr} + x)$ si ottiene l'indirizzo della locazione di memoria **distante x posizioni rispetto alla locazione puntata da ptr .**

L'incremento è operato dal compilatore, e dipende dalla dimensione in byte del tipo di ptr .

Aritmetica dei puntatori



Supponendo $\text{sizeof}(\text{int})=4..$

```
1 int a[10];
2 cout << a; // 0x23aaff40
3 cout << (a+1); // 0xaa23ff44
4 cout << &a[1]; // 0xaa23ff44
```

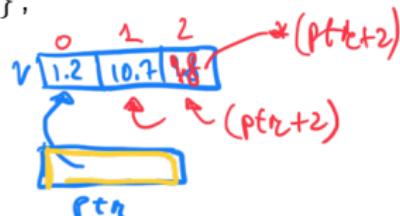
$$0x23aaff44 = 0x23aaff40 + 4$$

$(a+2)$ $0x23aaff44+8$ $(a+1) \leftarrow$ $a[1]$

Aritmetica dei puntatori

Si provi

```
1 double v [] = {1.2, 10.7, 9.8};  
2 double *ptr = v;  
3 cout << ptr;  
4 cout << (ptr + 1);  
5 cout << (ptr + 2);  
  
cout << *(ptr+2);
```



Alla linea 3 sarà stampato l'indirizzo contenuto in *ptr* in formato esadecimale.

Alla linea 4 sarà stampato l'indirizzo contenuto in *ptr + 1* valore restituito dall'espressione `sizeof(double)` in formato esadecimale.

Alla linea 5 ...

Esempi svolti

17_02.cpp

Aritmetica dei puntatori

Accesso ai valori di un array.

Notazione mediante indici vs aritmetica puntatori.

```
int v [] = {1,2,3};  
int *ptr = v;
```

Metodo di accesso	Esempio
Nome array e []	<u>v[2]</u>
Puntatore e []	<u>ptr[2]</u>
Nome array e aritmetica dei puntatori	<u>*(v+2)</u>
Puntatore e aritmetica dei puntatori	<u>*(ptr+2)</u>

Aritmetica dei puntatori

```
1 int v [] = {1,2,3};  
2 int *ptr = v;  
3  
4 *(ptr+7) = 90; // errore a run-time  
5 v[4] = 100; // errore a run-time
```

Le linee di codice 4 e 5 saranno **compilate**, senza alcun warning.

Tuttavia quel codice rappresenta tentativi di accesso (e di modifica) a zone di memoria non allocate per l'applicazione.

Aritmetica dei puntatori

```
1 int v[] = {1,2,3};  
2 int *ptr = v;  
3  
4 *(ptr+7) = 90; // errore a run-time  
5 v[4] = 100; // errore a run-time
```

Nella maggior parte dei casi il **Sistema Operativo** invierà un segnale di kill all'applicazione a causa del **tentativo di accesso a locazioni di memoria non assegnate al processo**.

Operatori consentiti per aritmetica dei puntatori.

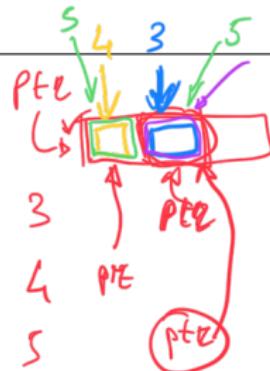
- **Operatori unari di incremento** $++$ / $--$ applicati ad una variabile puntatore.
- **Operatori binari di addizione e sottrazione** $+$ / $-$ e di **assegnamento** $+=$, $-=$, $+$, $-$, in cui un membro è un intero e l'altro membro è un puntatore.
- **Operatore di sottrazione** $-$ applicato a due puntatori.
Ovvero il valore di un puntatore può essere sottratto al valore di un altro puntatore.

Aritmetica dei puntatori



Incremento e decremento unario

```
1 int v [] = {1, 2, 3, 4, 5};  
2 int *ptr = v;  
3 cout << *(++ptr); //stampa 2  
4 cout << *(--ptr); //stampa 1  
5 cout << *(ptr++); //stampa 1  
6 cout << *(ptr); //stampa 2
```



Aritmetica dei puntatori

Addizione/sottrazione e assegnamento.

```
1 - int v [] = {1, 2, 3, 4, 5};  
2 - int *ptr1 = v; // punta al dato '1'  
3 - int *ptr2 = &v [4]; // punta al dato '5'  
4 cout << *(ptr1+1); // stampa il dato '2'  
5 cout << *(ptr2-1); // stampa il dato '4'  
6 ptr2 -= 2;  
7 cout << *ptr2; // stampa il dato '3'  
8 ptr1 += 1; //  $ptr1 = ptr1 + 1$   
9 cout << *ptr1; // stampa il dato '2'  
10 cout << ptr2 - ptr1; // stampa 1 (non e' un dato...)
```

$$ptr1 = \underline{ptr1 - 2};$$

Inizializzazione di puntatori

```
1 int *ptr = NULL; //macro C/C++  
2 int *ptr1 = nullptr; // C++11  
3  
4 float f;  
5 int *ptr3 = &f; // Errore di compilazione!  
6  
7 if(!ptr){ // test if ptr is valid  
8     //.. do something  
9 }
```

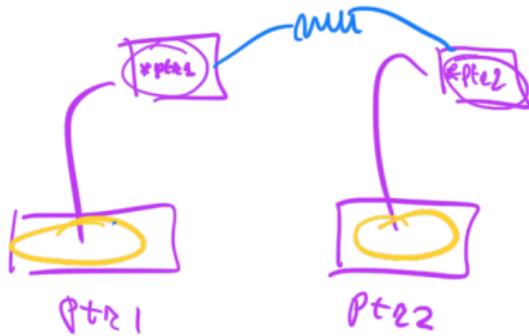


Linea 7, ptr all'interno di un if per verificare che il puntatore **non sia null**.

Operatore di confronto

Valore vs indirizzo!

```
1 int num;  
2 int *ptr1 = &num;  
3 int *ptr2 = &num;  
4 //confronta indirizzi  
5 if(ptr1==ptr2){  
6 /...  
7 }  
8 //confronta valori  
9 if(*ptr1==*ptr2){  
10 /...  
11 }
```



Puntatori costanti e puntatori a costanti

```
1 double d1 = 10.9;  
2 double d2 = 4.5;  
3  
4 const double *ptr1 = &d1;  
5 *ptr1 = 56.9; //errore!  
6 ptr1 = &d2; //OK
```

ptr1 è un puntatore a costante di tipo double.

Ciò significa che il **valore alla locazione di memoria puntata da ptr1 non è modificabile** mediante ptr1. La variabile d1 può essere const o non const.

Puntatori costanti e puntatori a costanti

```
1 double d1 = 10.9;  
2 double d2 = 4.5;  
3  
4 double * const ptr2 = &d2;  
5 ptr2 = &d1; //errore!  
6 *ptr2 = 10.5; //OK
```

ptr2 è un puntatore costante ad un tipo double.

Ciò significa che il **puntatore è una variabile costante**: va inizializzato contestualmente alla sua dichiarazione e non potrà subire riassegnamenti, come una qualunque variabile costante.

Puntatori costanti e puntatori a costanti

```
1 double d1 = 10.9;  
2 double d2 = 4.5;  
3  
4 const double *const ptr3 = &d2;  
5 ptr3 = &d1; //errore!  
6 *ptr3 = 10.5; /errore!
```

ptr3 è un puntatore costante ad una costante di tipo double.

Ciò significa che il **puntatore è una variabile costante** e che tramite il puntatore stesso non è possibile modificare il valore alla locazione di memoria alla quale esso punta.

Puntatori costanti e puntatori a costanti

Riassumendo: Const vs puntatori.

Dichiarazione	Istruzione	Corretta?
const double *ptr	*ptr = 45.9	NO
	ptr = &x	OK
double * const ptr	*ptr = 45.9	OK
	ptr = &x	NO
const double * const ptr	*ptr = 45.9	NO
	ptr = &x	NO

FINE