PIEMONTE
FONDI STRUTTURALI E DI INVESTIMENTO EUROPEI 2014/2020
fondo sociale europeo
ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

FONDAMENTI DI PROGRAMMAZIONE

C / C++

Docente: Armando Valentino

in collaborazione con:
ISTITUTI TECNICI SUPERIORI
Ministero dell'Istruzione
UNIONE EUROPEA
REGIONE PIEMONTE
per una crescita intelligente, sostenibile ed inclusiva
www.regione.piemonte.it/europa2020
INIZIATIVA CO-FINANZIATA CON FSE

LINGUAGGIO C++

PUNTATORI

ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

Prof. Armando Valentino

POR Piemonte FSE 2014-2020

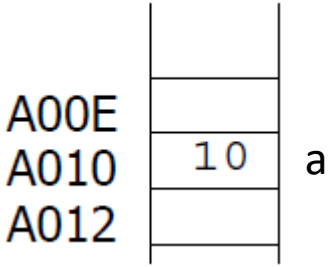
CC BY NC ND

Variabile

Esempio: int **a** = 10;

Proprietà della variabile **a** :

Nome:	a
Tipo:	int
Valore:	10
Indirizzo:	A010



Di solito usiamo di una variabile solo le prime tre proprietà, come utilizzare l'indirizzo??
Si deve usare l'operatore **&** (**operatore indirizzo**)
&a è l'operatore indirizzo applicato alla variabile **a** e **permette di prelevare il valore dell'indirizzo** della variabile **a**

Gli **indirizzi** possono essere **memorizzati nelle variabili puntatore**

Puntatore

Un **puntatore** è una variabile che **memorizza** l'*indirizzo* di una locazione di memoria, cioè **l'indirizzo di una variabile**.

Un puntatore deve essere dichiarato come qualsiasi altra variabile, in quanto anch'esso è una variabile. Per esempio:

int *p;

indica la dichiarazione di una variabile di tipo: *puntatore ad un intero*.

L'introduzione del carattere * davanti al nome della variabile indica che si tratta di un puntatore del tipo dichiarato.

int *p; Proprietà della variabile p

Nome:	p
Tipo:	puntatore ad int (indirizzo di intero)
Valore:	in principio nullo
Indirizzo:	fissato

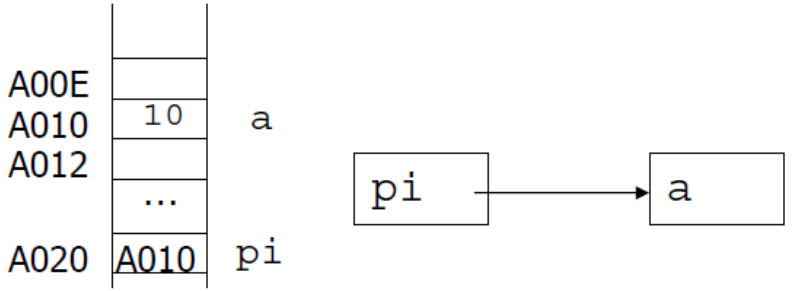
Esempi di dichiarazioni

```
int *p1, *p2;          // puntatori ad interi
float *pf;             // puntatore a float
```

Puntatori

Inizializzazione di un puntatore utilizzando l'operatore di indirizzo &.

```
int *pi, a;
a=10;
pi = &a;
```



Operatore di **dereferenziamento** "*"

Applicato ad una variabile puntatore fa riferimento all'oggetto puntato

```
int *pi, a = 10, b;
pi = &a;          /* pi = indirizzo di a*/
b = *pi;          /* inserisce in b il valore della variabile puntata da pi */
```

Nota: se **pi** è di tipo *** int** allora ***pi** è **int**

Operatori e priorità

Non confondere:
- “*” in **una dichiarazione**: serve per **dichiarare una var. di tipo puntatore** (es. int *pi;)
- “*” in **una espressione**: è **l’operatore di dereferenziamento** (es. a = *pi;)

Priorità degli operatori di dereferenziamento ed indirizzo:
- **Hanno priorità più elevata degli operatori binari**
- “*” associativo a destra: **p equivale a *(*p)
- “&” applicabile solo a variabile: es. &a
- “*” “&” uno inverso dell’altro:

int a;

&a ⇒ = a equivale ad a (*&a ed a sono due variabili) (a è un intero)

int *pi;

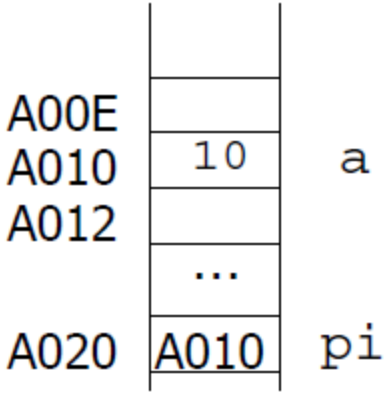
&*pi ⇒ ha valore = pi equivale al valore di pi (pi è una variabile, &*pi non è una variabile, ma un valore)

Stampa dei puntatori

I puntatori si possono stampare con **printf** e specificatore di formato %p (stampa in esadecimale

Esempio

```
int a = 10;
int *pi;
pi = &a;
printf(“indirizzo di a: %p”, &a);
printf(“valore di pi: %p”, pi);
printf(“valore di  &*pi: %p” , &*pi);
printf(“valore di a: %d”, a);
printf(“valore di *pi: %d”, *pi);
printf(“valore di *&a: %d”,*&a);
```



Inizializzazione dei puntatori

Come tutte le variabili, i puntatori devono essere inizializzati prima di essere usati.

Errore: dereferenziare una variabile puntatore non inizializzata

```
int a;
int *pi;          // puntatore non inizializzato
a = *pi;          //errore: dereferenziazione del puntatore non inizializzato
*pi = 500;        /* errore: assegnazione di valore a variabile puntata dal puntatore (che
                     non ha una variabile a cui puntare (puntatore non inizializzato)*/
```

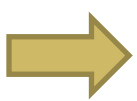
Funzione sizeof

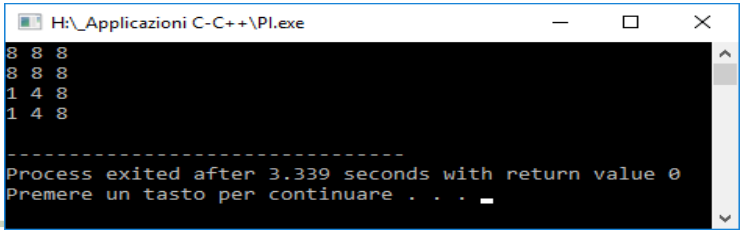
La funzione **sizeof()** restituisce l'occupazione di memoria in byte di una variabile.
Può essere anche applicata ad un tipo.

spazio di memoria di un indirizzo:

```
char *pc;
int *pi;
double *pd;
printf("%d %d %d \n", sizeof(pc), sizeof(pi), sizeof(pd));
printf("%d %d %d \n", sizeof(char *), sizeof(int *), sizeof(double *));
printf("%d %d %d \n", sizeof(*pc), sizeof(*pi), sizeof(*pd));
printf("%d %d %d \n", sizeof(char), sizeof(int), sizeof(double));
```

L'oggetto puntato ha dimensione del tipo puntato





Puntatore a puntatore

Una variabile di tipo puntatore è una variabile, per cui è una zona di memoria, per cui si può trovare il suo indirizzo usando l'operatore &.

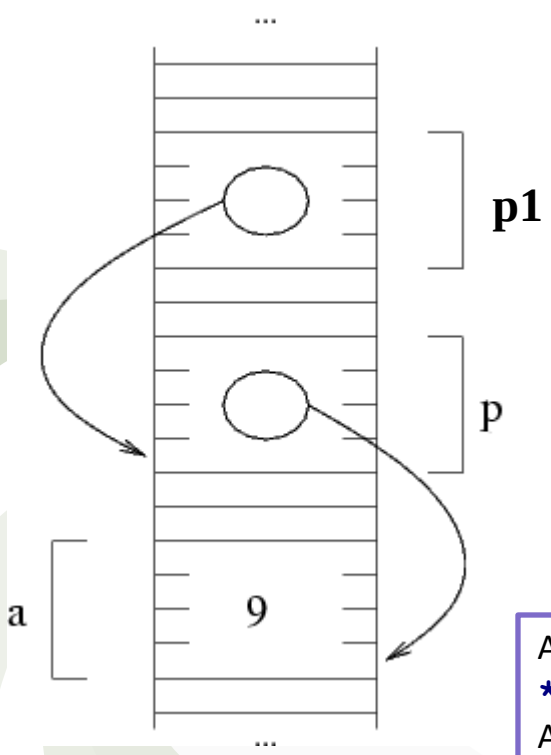
L'indirizzo di una variabile puntatore può essere conservato in un'altra variabile puntatore, questa ultima deve essere una variabile puntatore a puntatore

Esempio:

```
int a;        // dichiarazione variabile int
int *p;       // dichiarazione puntatore a int
int **p1;     // dichiarazione di puntatore a puntatore a int
a=9;
p=&a;         // assegno alla variabile p l'indrizzo di a
p1=&p;        // assegno alla variabile p1 l'indirizzo di p (puntatore)
printf("Indirizzo di p1=%x, valore=%x\n", &p1, p1);
printf("Indirizzo di  p=%x, valore=%x\n", &p, p);
printf("Indirizzo di  a=%x, valore=%x\n", &a, a);
```

Se si esegue questo programma su diversi computer, l'unico valore che rimane lo stesso è il valore di a che è 9.
Tutti gli altri valori sono indirizzi che variano su ogni computer, perché l'indirizzo può assumere un valore diverso su ogni computer.
L'unica cosa che non cambia è che *p1* punta a *p* e *p* punta ad *a* ***cioè p1->p->a***

Puntatore a puntatore



La figura qui accanto mostra una rappresentazione della memoria:
`p` contiene l'indirizzo di `a`, e a sua volta `p1` contiene l'indirizzo di `p`.

```
int a=9;
int *p= &a;
Int *p1 =&p;
```

Dato il valore di `p1` è chiaro che è possibile accedere al valore di `a`: basta seguire i puntatori, ossia prima trovare il valore di `*p1`, che è l'indirizzo di `a`,
e questo permette di trovare il valore di `a` usando ancora l'operatore `*`.
Quindi, dato `p1`, il valore di `a` si può trovare con `**p1`.

Assegno un valore ad `a` usando `p1`

```
**p1=10
```

Assegno un valore ad `a` usando `p`

```
*p=10
```

Prelevo il valore di `a` con i puntatori e lo metto in una variabile `b`

```
int b;
b = *p;
b = **p1;
```

Licenza

Quest'opera è distribuita con Licenza Creative Commons Attribuzione - Non commerciale - Non opere derivate 4.0 Internazionale.