



**UNIVERSITÀ DEGLI STUDI
DEL SANNIO** Benevento
DING
DIPARTIMENTO DI INGEGNERIA

CORSO DI "PROGRAMMAZIONE I"

Prof. Franco FRATTOLILLO
Dipartimento di Ingegneria
Università degli Studi del Sannio

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 1

Puntatori

- Un puntatore è una variabile che contiene l'indirizzo di memoria di un'altra variabile o costante
- I puntatori sono "type bound", cioè ad ogni puntatore è associato il tipo della variabile che può essere riferita
- Nella dichiarazione di un puntatore bisogna specificare un asterisco * prima del nome della variabile puntatore
- Esempio:


```
int *pointer;    // puntatore a intero
char *punCar;   // puntatore a carattere
float *fltPnt;  // puntatore a float
```

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 2

Inizializzazione di puntatori

- Prima di poter usare un puntatore è necessario inizializzarlo, ovvero deve contenere l'indirizzo di un oggetto
- Per ottenere l'indirizzo di un oggetto si usa l'operatore unario di referenziazione &


```
int x, *p;
p = &x;
```
- Il puntatore p contiene ora l'indirizzo della variabile x
- Per assegnare un valore all'oggetto puntato da p occorre utilizzare l'operatore di dereferenziazione *


```
*p = 15;
```

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 3

Esempio

```
int x=15;
int *p;

p:A20
A55
```

x:A55
15

memoria

| | | |
|-----|-----|---|
| A55 | 15 | x |
| | ... | |
| | ... | |
| | ... | |
| | ... | |
| A20 | A55 | p |

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 4

Assegnazioni e priorità

- Sia un puntatore `p` che la variabile puntata `*p` sono modificabili, ossia ad essi si può assegnare un valore
- L'operatore `*` ha priorità superiore a quella degli operatori matematici


```
int x = 2;
int *p = &x;
x = 6 * *p;  equivale a:  x = 6 * (*p);
```
- Per visualizzare il valore di un puntatore si può utilizzare la specifica di formattazione `%p` in una `printf`

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 5

La necessità del "type bound"

- L'informazione relativa al tipo è necessaria per permettere ai puntatori di conoscere la dimensione di memoria dell'oggetto puntato (necessaria per implementare l'aritmetica dei puntatori)
- Poiché variabili di tipo diverso possono avere dimensione diversa in termini di byte di memoria, l'assegnazione tra puntatori di tipo diverso è in genere errata ed il compilatore dovrebbe segnalare con un "warning"

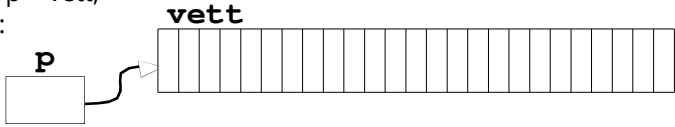

```
int *p, x=2;
long *q, y=6;
p = &x;  OK!
q = &y;  OK!
q = p;   NO! Warning
q = &x;  NO! Warning
```

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 6

Puntatori ed array ...

- Il nome di un array costituisce un puntatore costante alla testa dell'array, cioè al primo elemento dell'array
- Pertanto, i puntatori e gli array sono strettamente correlati, al punto da poter essere interscambiati nel loro uso


```
int vett[25];
int *p = vett;
vett:
```



The diagram illustrates the relationship between a pointer and an array. On the left, a box labeled 'p' represents the pointer variable. An arrow points from this box to the first cell of a horizontal array of 25 cells. The array is labeled 'vett' at its top-left corner.

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 7

Puntatori ed array ...

- Una variabile di tipo puntatore-a-T, assegnata in modo che punti ad un oggetto di tipo array-di-T, può essere utilizzata come se fosse un array-di-T
- Ad esempio


```
int vett[10];
int *p = vett;
qui p[3] equivale a vett[3]
```
- Il compilatore internamente trasforma le espressioni con notazione di array `[]` in espressioni con i puntatori

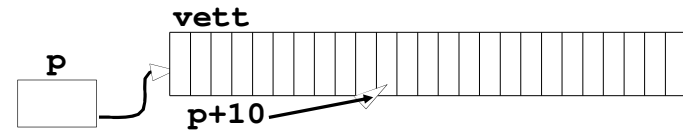
Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 8

Puntatori ed array

- Qualunque operazione eseguita usando un array ed un indice può essere implementata usando i puntatori
- Ad esempio, l'indirizzo dell'elemento di posto zero dell'array è il puntatore alla testa dell'array, per cui:
`int vett[10], *p, num;`
`p = &vett[0] equivale a p = vett`
`num = vett[0] equivale a num = *p`

Riferimenti negli array

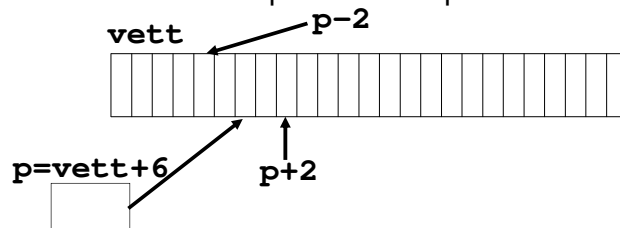
- Sommando un valore intero N ad un puntatore p che punta alla testa di un array, si ottiene l'indirizzo di memoria dell'elemento di posizione N dell'array



- È possibile riferire il valore dell'elemento i-esimo di un array `vett[i]` nella seguente forma:
`*(vett+i)` oppure `*(p+i)`

Spiazzamenti

- Se p punta ad un particolare elemento di un array, allora `p+1` punterà all'elemento successivo dell'array, `p-1` a quello precedente, e `p+i` allo i-esimo elemento successivo all'elemento puntato da p



Differenze tra puntatori ed array

- Esiste una differenza sostanziale tra puntatore ed array:
 - il puntatore è comunque una variabile, mentre il nome di un array è una costante e riferisce una collezione di oggetti di ugual tipo
- Ciò comporta che espressioni del tipo:
 - `p = vett` e `p++` sono legali, mentre espressioni del tipo:
 - `vett = p` ed `vett++` sono illegali !!
 - il nome di un array non può mai figurare alla sinistra di un'assegnazione

Aritmetica dei puntatori ...

- Sui puntatori sono lecite le seguenti operazioni:
 - assegnamento tra puntatori dello stesso tipo


```
int *ptr1, *ptr2, val=1;
ptr1 = &val;
ptr2 = ptr1; /* ptr2 punta al valore 1 */
```
 - addizione e sottrazione tra puntatori ed interi


```
int *ptr, arr[10];
ptr = &arr[0]; /* oppure ptr = arr;
ptr = ptr + 4 /* punta al quinto elemento dell'array (arr[4]) */
ptr = ptr - 2 /* punta al terzo elemento dell'array (arr[2]) */
```

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 13

Aritmetica dei puntatori

- Assegnamento di un puntatore con lo zero


```
int *ptr;
ptr = 0; /*indica che ptr non punta a nulla */
```
- Confronto di un puntatore con lo zero


```
int *ptr;
ptr = 0;
if (ptr == 0) printf("Il puntatore non è inizializzato\n");
```
- Da notare che il C garantisce che lo zero non sia un indirizzo valido per i dati, per cui il riferimento a zero si usa per indicare un puntatore non inizializzato
 - lo zero può essere sostituito dalla costante simbolica NULL definita nella libreria standard

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 14

Operazioni con i puntatori

- Sottrazione tra due puntatori ad elementi di uno stesso array


```
int *ptr1, *ptr2, arr[10];
ptr1 = ptr2 = arr;
ptr1 += 10; /* punta fuori dall'array */
ptr1 - ptr2 /* indica la lunghezza dell'array: 10 */
```
- Confronto tra puntatori ad elementi di uno stesso array


```
int *ptr1, *ptr2, arr[10];
ptr1 = ptr2 = arr;
ptr1 += 5;
ptr1 > ptr2 /* è vero se ptr1 punta ad un elemento che nell'array è successivo all'elemento puntato da ptr2 */
```

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 15

Cosa non è consentito fare con i puntatori

- Non è ASSOLUTAMENTE consentito:
 - sommare, moltiplicare o dividere due puntatori
 - sommare e sottrarre quantità float o double ad un puntatore
 - assegnare ad un puntatore di un tipo un puntatore di un tipo differente, a meno di:
 - utilizzare un puntatore a void
 - utilizzare l'operatore cast


```
void *ptr1; int *ptr2; char *ptr3;
...
ptr1 = ptr3;
ptr2 = (int *) ptr1;
```

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 16

Priorità

- L'operatore di deriferimento `*` ha priorità quasi massima, inferiore solo alle parentesi (e a `'->'` e a `'.'`) e associatività da destra a sinistra
- Considerando che gli operatori `*` e `++` hanno stessa priorità e associatività da D a S:
 - `*p++` equivale a `*(p++)` → incrementa p e ...
 - `*++p` equivale a `*(++p)` → incrementa p e ...
 - `++*p` equivale a `++(*p)` → incrementa `*p`

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 17

Esempi

- `/* Dichiarazioni */`
- `int v1, v2, *v_ptr;`
- `/* moltiplica v2 per il valore puntato da v_ptr */`
- `v1 = v2 * (*v_ptr);`
- `/* somma v1, v2 e il valore puntato da v_ptr */`
- `v1 = v1 + v2 + *v_ptr;`
- `/* assegna a v2 il valore che si trova tre interi dopo v_ptr */`
- `v2 = *(v_ptr + 3);`
- `/* incrementa di uno l'oggetto puntato da v_ptr */`
- `*v_ptr += 1;`

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 18

Esempi

- `/* azzera l'intero puntato da v_ptr */`
- `*v_ptr = 0;`
- `/* incrementa di uno l'oggetto puntato da v_ptr */`
- `++*v_ptr;`
- `/* azzera l'intero puntato da v_ptr e incrementa il puntatore */`
- `*v_ptr++ = 0;`
- `/* incrementa il puntatore e azzera l'intero puntato da v_ptr */`
- `(*v_ptr++) = 0;`

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 19

Puntatore variabile a valore costante ...

- Si può specificare nei due modi equivalenti:
 - `int const *p;`
 - `const int *p;`
- `p` è una variabile di tipo puntatore-a-costante, cioè ad un oggetto costante di tipo `int`
 - `const int x=3, y=5;`
 - `const int *p; /* p è puntatore-a-costante */`
 - `p = &x; ok /* p è una variabile */`
 - `p = &y; ok /* p è una variabile */`
 - `*p = 13; NO perché *p è costante`

Franco FRATTOLILLO - Dipartimento di Ingegneria - Università degli Studi del Sannio Corso di "Programmazione I" - Corso di Laurea in Ingegneria Informatica / ExAT 20



Puntatore variabile a valore costante

- L'assegnazione di un valore di tipo puntatore-a-costante (l'indirizzo di un valore costante) ad una variabile di tipo puntatore-a-variabile genera un "warning" del compilatore, perché permette di by-passare la restrizione const

```
const int x = 12;
int y = 10;
int *p;
const int *q; /* puntatore-a-costante */
p = &x;      /* warning */
*p = 5;      /* warning */
q = &x;      /* ok */
*q = 5;      /* da' errore */
```



Puntatore costante a variabile

- Si specifica nel seguente modo:

```
int * const p;
```
- p è una costante e può puntare ad un oggetto variabile di tipo int
- Le costanti possono essere solo inizializzate

```
int x, y;
int * const p = &x; → inizializzazione all'atto della definizione di p
```
- *p = 13; → OK, *p è una variabile

```
p = &y; → NO, p è una costante
```



Puntatori a void ...

- Sono puntatori generici e non possono essere dereferenziati
 - non si può scrivere *p
- Possono essere utilizzati solo come contenitori temporanei di valori di tipo puntatore (a qualsiasi tipo), e non serve il cast (void *) per copiare un puntatore non-void in un puntatore void

```
void *h;
int *p=&...;
h = p;
```
- Qualsiasi tipo di puntatore può essere confrontato con un puntatore a void



Puntatori a void

- Per dereferenziare il valore di un puntatore a void è necessario prima assegnarlo ad un puntatore al tipo appropriato (non void), per poter conoscere la dimensione dell'oggetto puntato
- Può essere necessario il cast (tipo *) per copiare un puntatore void in un puntatore non-void

```
int *q;
void *h;
..... /* h viene assegnato */
q = (int*) h; oppure q = h;
```