I puntatori

Associazione Variabile-Indirizzo

- Tutte le volte che in un programma si dichiarano delle variabili, l'esecutore associa alla variabile un indirizzo.

Es. Dato il programma

```
#include <stdio.h>
main()
{ int a,b,somma;
  scanf("%d %d",&a,&b);
  somma=a+b;
  printf("%d",somma);
}
```

Si crea in questo modo una *tabella di associazioni* variabile-indirizzo

variabile	indirizzo
a	Ind_a
b	Ind_b
somma	Ind_somma

Associazione Variabile-Indirizzo

Per esempio, se Ind_a=100, Ind_b=110 e Ind_somma=130,
se in seguito all'esecuzione della scanf a = 7; b = 10, la configurazione della memoria sarà come quella illustrata in Tabella

Per eseguire l'istruzione

```
somma=a+b;
```

l'esecutore valuta l'espressione a destra dell'uguale:

1. identifica **a** considerando l'indirizzo **Ind_a** e legge il contenuto di tale locazione;
2. identifica **b** considerando l'indirizzo **Ind_b** e legge il contenuto di tale locazione;
3. somma **a** e b;
4. assegna alla variabile somma (cioè alla locazione di indirizzo Ind_somma) il risultato della somma

locazione di memoria	Indirizzo
7	Ind_a=100
	…
10	Ind_b=110
	…
	Ind_somma=130

Memoria Riservata per Tipi di Dati

- La quantità di locazioni di memoria riservate a ciascuna variabile (`int`, `char`, `float`, ecc.) dipende dal tipo di compilatore e dal tipo di dato adoperato.
- Quando si dichiara un vettore:

  ```
  int V[N];
  ```

  l'esecutore riserva una quantità di locazioni di memoria tale da contenere N variabili di tipo intero.
- L'indirizzo associato al vettore V sarà quello della prima locazione di memoria riservata al primo elemento del vettore.
- Per determinare l'indirizzo della locazione di memoria del generico elemento di posizione i, l'esecutore applicherà la seguente formula:

*Indirizzo_V[i]=Indirizzo_V[0]+i*Numero_locazioni_per_un_elemento*

In linguaggio C la funzione sizeof(...) restituisce la dimensione in byte del dato passato come parametro.
Quindi l'elemento V[i] avrà indirizzo: *Indirizzo_V[0]+i*sizeof(int)*

I Puntatori

- I puntatori sono variabili atte a contenere l'indirizzo di locazioni di memoria in cui sono memorizzate le variabili di un certo tipo.
- Per esempio, per dichiarare una variabile puntatore che dovrà contenere l'indirizzo di una variabile intera basta fare:

```
int *p;
```

- Generalizzando, per dichiarare una variabile puntatore che dovrà contenere l'indirizzo di una variabile di tipo tipo basta fare:

```
tipo *p;
```

- Per ottenere l'indirizzo di una variabile si usa l'operatore &. Per esempio, si consideri il seguente frammento di codice:

```
int *p;
int a;
p = &a;
```

l'ultima istruzione assegna alla variabile puntatore ad intero p l'indirizzo della variabile intera a. Diremo in questo caso che *p sta puntando ad a*.

Accesso mediante puntatore

- L'accesso al dato contenuto nella locazione di memoria il cui indirizzo è memorizzato in una variabile puntatore, avviene utilizzando l'operatore *.
- Per esempio si consideri il seguente frammento di codice:

```
int *p;
int a;
p = &a;
a = 7;
printf("%d", *p);
*p = 18;
printf("%d", a);
```

- L'esecuzione di questo codice visualizzerà 7 e poi 18.

Accesso mediante puntatore

- **Esempio:** Si consideri la seguente dichiarazione:

```
int *p;
int a, b;
```

che supponiamo dia luogo alla seguente tabella delle associazioni

variabile	indirizzo
a	68
b	80
p	90

Per assegnare il valore *47* alla variabile **a** basta fare:

```
a = 47;
```

Un'altro metodo possibile è fare:

```
p = &a;
*p = 47;
```

in cui la prima istruzione (p = &a;) assegna a p l'indirizzo della variabile **a** e la seconda istruzione (*p = 47;) scrive nella locazione di memoria, il cui indirizzo è memorizzato in p, il valore *47*.

Assegnazioni tra Puntatori

- In C non è possibile attribuire ad un puntatore un dato indirizzo in maniera diretta. Cioè non è possibile fare:

```
int *p;
p = 105;
```

- La forma corretta è:

```
int *p;
int a;
p = &a;
```

dove si suppone che a sia stata allocata all'indirizzo 105.

- Non si confondano le seguenti sintassi:

```
int *p, *q;
p = q;
*p = *q;
```

La prima assegna a ***p*** l'indirizzo contenuto in ***q***. Cioè se **q** punta alla locazione 39, allora a seguito dell'assegnazione p = q; anche p punterà alla locazione 39.

La seconda, invece, assegna alla locazione puntata da p il valore contenuto nella locazione puntata da q.

Visualizzazione del valore di un puntatore

- Per visualizzare il valore di un puntatore (indirizzo) si utilizza il seguente formato %p

- Es.

```
#include <stdio.h>

main() {
        int A,*p;

        p=&A;

        printf("L'indirizzo di A e': %p ", p);

}
```

Aritmetica dei Puntatori

- L'insieme delle operazioni eseguite sugli indirizzi memorizzati nelle variabili puntatore costituisce l'aritmetica dei puntatori.
- Dato:

```
int *p;
int a;
p =&a;
p++;
```

  Incrementare di 1 il valore delpuntatore'non significa sommare 1 all'indirizzo contenuto in p, bensì significa incrementare l'indirizzo di una quantità in byte pari alla dimensione del dato puntato.

- In questo caso, se per esempio indirizzo della variabile a è 30, e dunque p punta all'indirizzo 30, p++ assegna a p l'indirizzo 30 + sizeof(int) e non l'indirizzo 31.

Puntatori e Vettori

Quando si usa un vettore è possibile specificare un suo elemento non solo indicando la sua posizione tramite l'indice tra le parentesi quadre ma anche nel seguente modo. Sia data la seguente dichiarazione:

```
int tab[10];
int *p;
```

Scrivere:

```
tab[3] = 49;
```

equivale a scrivere:

```
p = tab;
p = p + 3;
*p = 49;
```

Quindi

tab[i] o *(p+i) sono equivalenti.

Puntatori e Vettori

Ricordando che tab rappresenta anche il puntatore al primo elemento dell'array, per accedere all'elemento i-esimo si può anche usare *(tab+i). Analogamente si può scrivere p[i].

Es. Leggere e visualizzare gli elementi di un vettore

```
#include <stdio.h>
main()
{int i,V[10],*p;

 p=V;
 for(i=0;i<10;i++)
    scanf("%d,V+i);
 for(i=0;i<10;i++)
    printf("%d",p[i]);
}
```