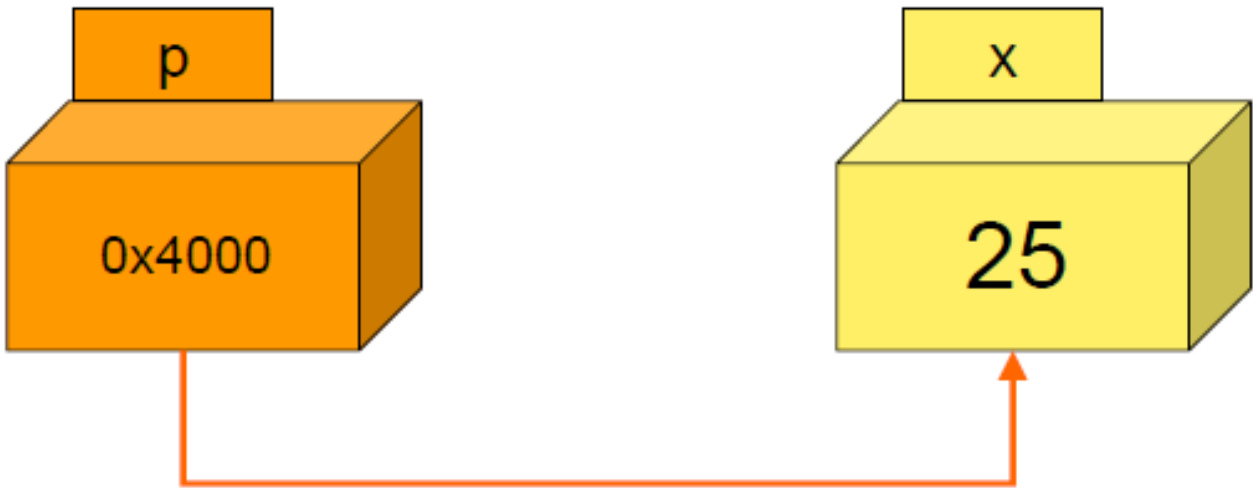
Introduzione

- Una variabile è caratterizzata da:
  - Nome
  - Tipo
  - Valore
  - Indirizzo in Memoria Centrale (MC)

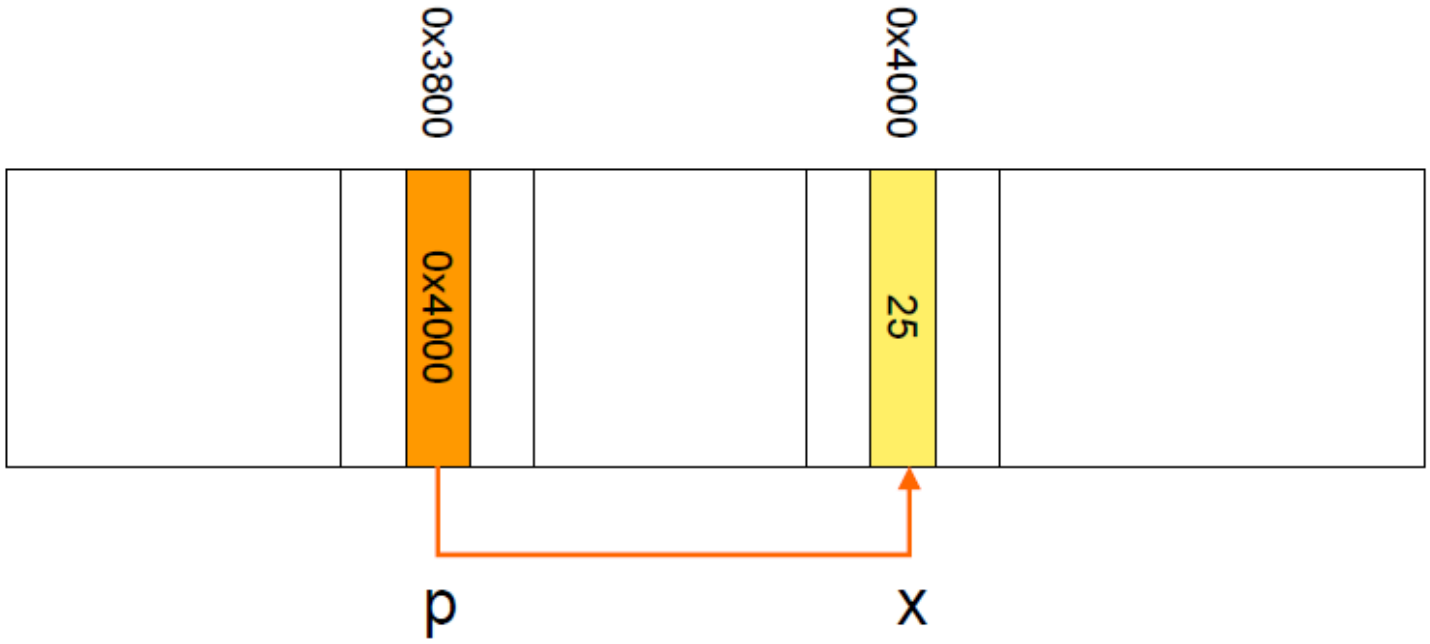
  int x = 25;
- Come possiamo conoscere l'indirizzo di x?
- In realtà lo abbiamo già visto ed utilizzato nell'istruzione scanf, è l'operatore &:
- **&x restituisce l'indirizzo in memoria centrale della variabile x**.

Puntatori e Indirizzi

Un puntatore e' una variabile il cui valore e' un indirizzo in memoria centrale.



Puntatori e Indirizzi



Il tipo puntatore

- L'espressione &x può essere assegnata a variabili di tipo puntatore, o più semplicemente puntatori.
- La sintassi per dichiarare un puntatore è la seguente:< *tipobase* > * < *identificatore* >;
- Esempio:
  int* pi; pi un puntatore a variabili di tipo intero.
- Altri esempi:
  int *pi1, *pi2;
  float *pf;
  double *pd;
- Notare che:

  ```
  int* pi;
  int * pi;
  int *pi;
  ```

  Sono tre scritture equivalenti per definire un puntatore ad interi

Abbiamo gia' visto che l'operatore unario & restituisce l'indirizzo della variabile cui viene applicato: &x restituisce l'indirizzo in memoria della variabile x.
Dunque

p= &x;

e' un assegnamento legittimo

La dichiarazione del puntatore avviene attraverso l'operatore unario * di indirezione o deriferimento.

```
int *p;
```

Indica che *p e' un intero!
Mentre p e' un puntatore ad interi.

Puntatori

```
int x=1, y=2, z[10];
int *p;
p=&x;//p punta a x
y=*p;//ad y assegno lo stesso valore
     //dell'intero puntato da p, cioe' 1
*p=0; //ad x assegno 0
p=&z[0]; //p punta al primo elemento di z
```

Puntatori tipati

- I puntatori sono vincolati ad un tipo di dati
  int *p;
  double d=5.0;
  p=&d; ILLEGALE!!
- Esiste pero' un'eccezione: il puntatore a void il quale puo' essere utilizzato per puntare ad un qualsiasi tipo di dato, tale puntatore e' infatti detto generico

Se p punta ad un intero x, allora *p puo' comparire in qualsiasi contesto nel quale puo' comparire x.

```
*p=*p+5;
y=*p+1;
*p+=20;
++*p;
(*p)++;
```

Stampa dei puntatori

I puntatori si possono stampare con printf usando %p come carattere di controllo, l'indirizzo viene visualizzato in formato esadecimale

```
int v = 5;
int *pi;
pi = &v;
printf("indirizzo di v = %p\n", &v);
//stampa 0xbacd7a20
printf("valore di pi = %p\n", pi);
//stampa 0xbacd7a20
printf("valore di &*pi = %p\n", &*pi);
//stampa 0xbacd7a20
printf("valore di v = %d\n", v);//stampa 5
printf("valore di *pi= %d\n", *pi); //stampa 5
printf("valore di *&v = %d\n", *&v); //stampa 5
```

- Attenzione alle parentesi!
  (*p)++; //incremento il valore puntato da p
- E'diverso da:
  *p++; //espressione che restituisce il valore puntato da p e successivamente incrementa il puntatore p
- Infatti, gli operatori * e ++ hanno la stessa priorita' ed entrambi sono associativi da destra verso sinistra.

Passaggio dei Parametri

All'atto dell'invocazione di una funzione:

- si crea una nuova attivazione (istanza) della funzione
- si alloca la memoria per i parametri (e le eventuali variabili locali):[allocazione del record d'attivazione della funzione sullo stack]
- si trasferiscono i parametri alla funzione
- si trasferisce il controllo alla funzione
- si esegue il codice della funzione

Esistono due modi per passare i parametri ad una funzione:

- Per valore(by value): si copia il valore del parametro attuale
- Per riferimento (by reference): si passa un riferimento(indirizzo) del parametro attuale

Passaggio dei Parametri per Valore

In C, i parametri sono trasferiti sempre e solo per valore (by value)

- si trasferisce una copia del parametro attuale, non l'originale!!
- tale copia strettamente privata e locale alla funzione chiamata
- la funzione potrebbe quindi alterare il valore ricevuto, senza che ci abbia alcun impatto sul chiamante!!

Conseguenza:

- è impossibile usare un parametro per trasferire informazioni verso il chiamante
- per trasferire un'informazione al chiamante si sfrutta il valore di ritorno della funzione

Passaggio dei Parametri per Valore

Esempio:

```
int incrementa(int num) {
num++;
return num;
}
main() {
int x = 4;
int valore;
valore = incrementa(x);
}
```

Alla fine dell'esecuzione quanto vale x? E quanto vale valore?

Passaggio dei Parametri per Riferimento

- Il passaggio per Riferimento non trasferisce una copia del valore del parametro attuale ma un riferimento al parametro, in modo da dare alla funzione chiamata accesso diretto al parametro in possesso del chiamante.
- La funzione chiamata accede direttamente al dato del chiamante e può modificarlo.

Passaggio dei Parametri per Riferimento

- Per fare questo occorre essere capaci di:
  - ricavare l'indirizzo di una variabile
  - dereferenziare un indirizzo di variabile, ossia "recuperare" il valore di una variabile dato l'indirizzo della variabile stessa
- Il C offre a tale scopo due operatori, che consentono di:
  - ricavare l'indirizzo di una variabile: operatore estrazione di indirizzo &
  - dereferenziare un indirizzo di variabile: operatore di dereferenziamento *

Passaggio dei Parametri per Riferimento

Il chiamante deve passare esplicitamente gli indirizzi.
La funzione chiamata deve prevedere esplicitamente dei puntatori come parametri formali.

```
void swap(int* primo, int* secondo) {
int tmp;
tmp= *primo;
*primo = *secondo;
*secondo= tmp;
}

main(){
int y = 27;
int x = 36;
swap(&x, &y);
}
```