I PUNTATORI

Assegnazione di valori a puntatori

E' bene inizializzare un puntatore al momento della sua dichiarazione.

Per inizializzare un puntatore gli si può assegnare il valore di un indirizzo di memoria oppure il valore NULL (il puntatore non punta a nulla ma non è vuoto)

I puntatori

I puntatori sono variabili, il cui contenuto è un indirizzo di memoria (di variabili, di funzioni, etc).

Una variabile fa riferimento diretto al suo contenuto, mentre un puntatore fa riferimento al contenuto della variabile attraverso il suo indirizzo, e quindi in modo indiretto.

Assegnazione di valori a puntatori

Consideriamo puntatori a variabili.

```
int a = 6;
int *pa;
pa = &a;
```

pa è un puntatore ad una variabile intera, che conterrà l'indirizzo di memoria della variabile intera a.

Si dice che pa punta ad a.

Dichiarazione dei puntatori

Per poter essere usati i puntatori devono essere precedentemente dichiarati.

La dichiarazione del puntatore prevede che si conosca a priori a quale tipo di oggetto esso punta.

Esempio: Un puntatore p a variabile intera si dichiarerà come int *p.

p è un puntatore che conterrà l'indirizzo di memoria di una variabile intera

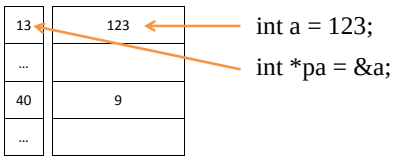
Assegnazione di valori a puntatori

```
float a = 6.5;
float *pa;
pa = &a;
```

```
char a = 'A';
char *pa;
pa = &a;
```

L'operatore di indirizzo &

L'operatore di indirizzo & è un operatore unario che agisce restituendo l'indirizzo di memoria dell'operando su cui agisce.



I sottoprogrammi – passaggio per riferimento

Esempio: Scrivere una funzione di tipo void che restituisca la somma di due numeri ricevuti dal main.

```
#include<stdio.h>
void fun(int, int, int *);
main()
{int a,b, somma;
scanf(‘‘%d%d’’,&a,&b);
fun(a,b,&somma);
printf(‘‘%d’’,somma);}
```

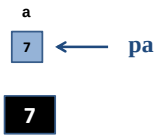
```
void fun(int x, int y, int *z)
{*z=x+y;}
```

L'operatore di deriferimento *

Dichiarando e inizializzando un puntatore lo si riempie con l'indirizzo di memoria della variabile a cui il puntatore punta.

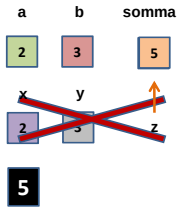
Per conoscere il contenuto della variabile puntata è possibile usare l'operatore di deriferimento *.

```
int a = 7;
int *pa = &a;
printf (‘‘%d’’,*pa);
```



I sottoprogrammi – passaggio per riferimento

```
void fun(int, int, int *);
main()
{int a=2, b=3, somma;
fun(a,b,&somma);
printf(‘‘%d’’,somma);}
```




```
void fun(int x, int y, int *z)
{*z=x+y;}
```

I sottoprogrammi – passaggio per riferimento

Ricordiamo che i parametri possono essere passati alla funzione **per riferimento.**

In C il passaggio per riferimento viene simulato attraverso l'uso dei puntatori.

I sottoprogrammi – passaggio per riferimento

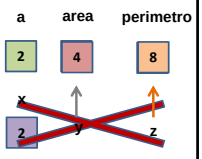
Esempio: Scrivere una funzione che restituisca l'area ed il perimetro di un quadrato, il cui lato viene letto nel main.

```
#include<stdio.h>
void fun(int, int *, int *);
main()
{int l,area, perimetro;
scanf(‘‘%d’’,&l);
fun(l,&area,&perimetro);
printf(‘‘%d %d’’,area, perimetro);}
```

```
void fun(int x, int *y, int *z)
{*y=pow{x,2};
*z=4*x;}
```

I sottoprogrammi – passaggio per riferimento

```
main()
{int l=2,area, perimetro;
fun(l,&area,&perimetro);
printf(“%d %d”,area, perimetro);}
```



4 8

```
void fun(int x, int *y, int *z)
{*y=pow{x,2};
*z=4*x;}
```

I sottoprogrammi – passaggio per riferimento

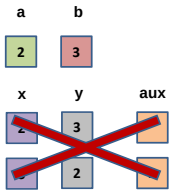
Il passaggio per riferimento si rende necessario quando

- la funzione deve modificare il valore del parametro (esempio: scanf),
- la funzione dovrebbe restituire più di una valore (IMPOSSIBILE, lo simuliamo col passaggio per riferimento) o agisce su strutture dati.
- il passaggio per valore (e quindi il fatto che viene fatta una copia del contenuto della variabile) implica un grande dispendio di risorse.

I sottoprogrammi – passaggio per riferimento

scambio valori di due variabili

```
void fun(int, int );
main()
{int a=2, b=3;
fun(a,b);
printf(“%d%d”,a,b);}
```



2 3

```
void fun(int x, int y)
{int aux;
aux=x;      x=y;      y=aux;}
```

I sottoprogrammi – passaggio per riferimento

C

```
void fun(int *y)
{*y = *y * 2;}
```

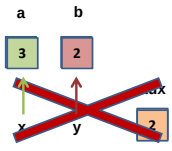
Visual Basic

```
Public Sub fun(ByRef y as integer)
y = y*2
```

I sottoprogrammi – passaggio per riferimento

scambio valori di due variabili

```
void fun(int *, int *);
main()
{int a=2, b=3;
fun(&a,&b);
printf(“%d%d”,a,b);}
```



3 2

```
void fun(int *x, int *y)
{int aux;
aux=*x;      *x=*y;      *y=aux;}
```