Strutture di controllo

ASSEGNAMENTO

Ad una variabile può essere assegnato un valore nel corso del programma e non solo all'atto della inizializzazione.

Assegnamento di una variabile: SINTASSI

<identificatore> = <espr> ;

L'assegnamento e' l'astrazione della modifica distruttiva del contenuto della cella di memoria denotata dalla variabile.


```
 int    x;
...
x = 5
```

ASSEGNAMENTO

L'assegnamento è un *particolare tipo di espressione* come t ale *denota comunque un valore* **con un effetto collaterale: quello di *cambiare il valore della variabile.***

Esempi di *espressioni di assegnamento*:

j = 0 k = j + 1

Se k valeva 2, l'espressione **k = j + 1**

– denota il valore 1 (risultato della valutazione dell'espressione)

– *e cambia il valore di k,* che d'ora in poi vale 1 (non più 2)

L'assegnamento è distruttivo

ASSEGNAMENTO

Se x valeva 2, l'espressione

x = x + 1

denota il valore 3

e cambia in 3 il valore di x

il simbolo x **a destra** dell'operatore = denota *il valore attuale (R-value) di x*, cioè 2

il simbolo x **a sinistra** dell'operatore = denota *la cella di memoria associata a x (L-value)*, a cui viene assegnato il valore dell'espressione di destra (3)

l'espressione nel suo complesso denota il valore della variabile dopo la modifica, cioè 3.

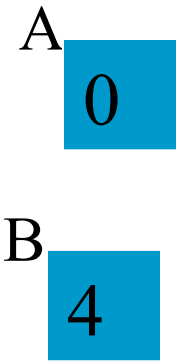
ASSEGNAMENTO

Supponiamo di avere due variabili.
A = 0 e B = 4
e vogliamo scambiare i loro valori.
Come fare ?

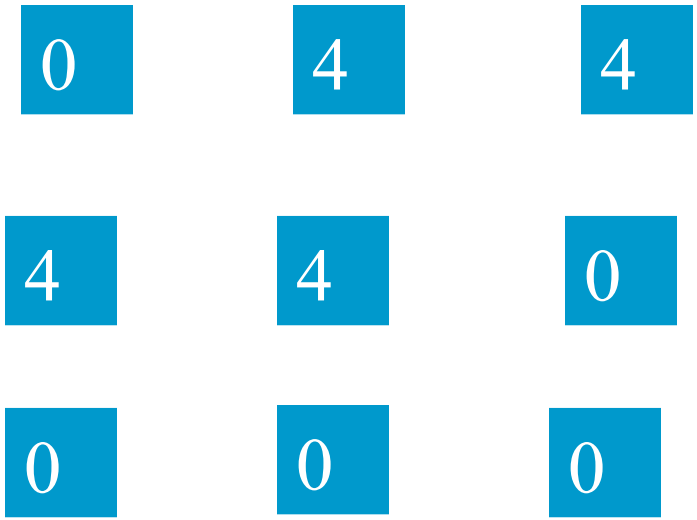
A = B;
B = A;

ASSEGNAMENTO

Supponiamo di avere due variabili.
A = 0 e B = 4
e vogliamo scambiare i loro valori.
Come fare ?



C= A
A = B;
B = C;



Compatibilità di tipo e conversioni

- In un assegnamento, l'identificatore di variabile e l’espressione devono essere dello stesso tipo.
- Conversioni di tipo hanno luogo quando sono impiegate variabili di tipo differente all’interno di un’unica espressione
- La conversione è implicita ma può causare perdita di informazione

CONVERSIONI DI TIPO

- In C e' possibile combinare tra di loro operandi di tipo diverso:

  espressioni **omogenee**: tutti gli operandi sono dello stesso tipo

  espressioni **eterogenee**: gli operandi sono di tipi diversi.

- **Regola adottata in C:**

  sono eseguibili le espressioni eterogenee in cui tutti i tipi referenziati risultano **compatibili** (cioe': dopo l'applicazione della regola automatica di conversione implicita di tipo del C risultano omogenei).

CONVERSIONI DI TIPO

- Data una espressione x op y.

- Ogni variabile di tipo **char** o **short** viene convertita nel tipo **int**;

- Se dopo l'esecuzione del passo 1 l'espressione e' ancora eterogenea, rispetto alla seguente gerarchia

  int < long < float < double < long double

  si converte temporaneamente l'operando di tipo inferiore al tipo superiore (**promotion**);
- A questo punto l'espressione e' **omogenea** e viene eseguita l'operazione specificata. Il risultato e' di tipo uguale a quello prodotto dall'operatore effettivamente eseguito. (In caso di overloading, quello più alto gerarchicamente).

Istruzioni di selezione

Selezione a due vie

if (<espressione>**)** <istruzione> else
<istruzione>

Selezione a due vie

l'operatore ?

Selettore a piu' vie

switch <variabile> {
case <costante1>: <istruzione>
case <costante2> :<istruzione>
case <costante n>: <istruzione>
default:
}

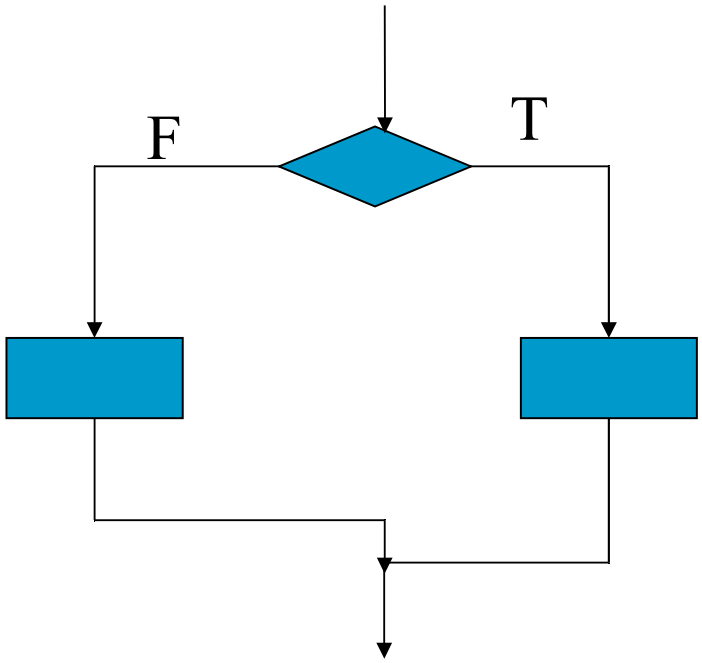
Istruzioni if

Selezione a due vie

if (<espressione>**)** <istruzione> else <istruzione>

if annidati

Lo standard ANSI richiede al compilatore di consentire al minimo 15 livelli di if annidati



Esempio

```
#include <stdio.h>

int main()
{  int a;        /*primo valore a*/
   int b;        /*secondo valore*/
   scanf("%d",&a);
   scanf("%d",&b);
   if (a>b) {
       printf("%d",a);
    printf("%d",b);
   }
   else{
       printf("%d",b);
    printf("%d",a);
   };
}
```

Istruzioni ?

Selezione a due vie

l'operatore ?

<espressione1> **?** <istruzione> **:** <istruzione>

Esempio

```
x = 10;
y = x>0 ? 12 : 22;
```

Esempio

```
x = 10;
if  (x>0)
      y = 12;
else
      y = 22;
```

Istruzione switch

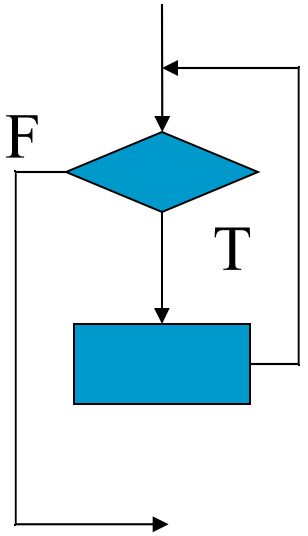
Selettore a piu' vie

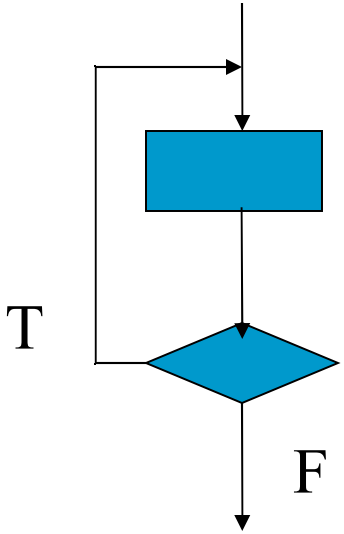
switch <variabile> {
case <costante1>: <istruzione>
case <costante2>: <istruzione>
case <costante n>: <istruzione>
default:
}

Istruzioni iterative

Ciclo a condizione iniziale

while (<espressione>**)** <istruzione>





Ciclo a condizione finale

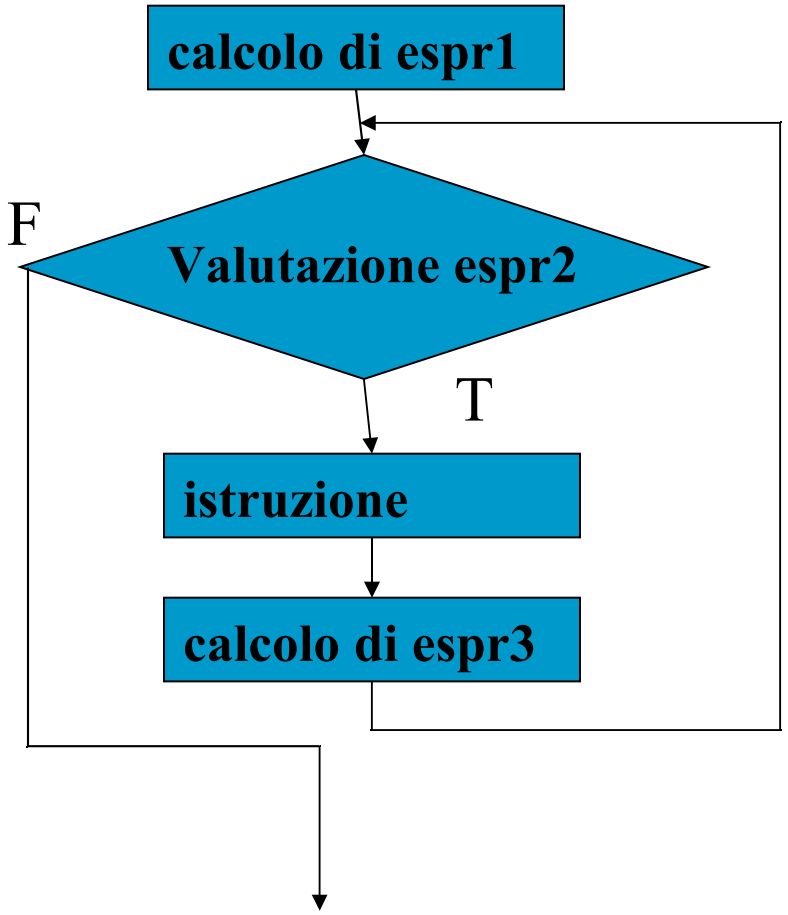
do <istruzione> **while (**<espressione>**)**

esempio

```
/* un menu */
int  main()
{    char car;
     printf(“1 per prima scelta\n”);
     printf(“2 per seconda scenta\n”);
     printf(“3 per terza scelta\n”);
     do {
          printf(“inserisci la tua scelta\n”);
          car = getchar();
          switch(car)
          case “1”:   prima(); break;
          case “2”:   seconda(); break;
          case “3”:   terza(); break;
     }
     while(car ==‘1’ || car ==‘2’ || car ==‘3’)
}
```

Ciclo for

for (<esp1>; <esp2>; <esp3>;**)** <istruzione>



<esp1>;

while (<esp2>**) do**

{ <istruzione>;

<esp3>;

}

- L'inizializzazione e’ una espressione, nella sua forma piu’ semplice e’ una assegnazione

- condizione e’ una espressione relazionale usata per valutare la variabile o le variaibili di controllo del loop; l’esecuzione del ciclo prosegue fino a quando questa condizione e’ vera, quando diviene falsa lesecuzione prosegue con l’istruzione successiva al for

- incremento e’ una espressione che definisce il modo in cui la variabile di controllo cambia il suo valore ad ogni ripetizione del loop

```
#include <stdio.h>
int main() {
      int x;
      for(x=1; x<=100; x++) printf(“%d”,x);
}
```

il test della consizione viene eseguto al momento del’entrata nel loop, prima delle istruzioni

```
#include <stdio.h>
int main(void) {
      int x;
      for(x=100; x>=0; x--) printf(“%d”,x);
}
```

```
#include <stdio.h>

int main() {
      int x;
      for(x=1; x<=100; x+=5) printf(“%d”,x);
}
```

```
int main()  {
            int x;
      for (x=1, y=1; x+y<=100; x++, y+=6)
             printf(“%d”,x+y);
}
```

loop senza fine

```
      for (;;) printf(“questo loop non finisce mai”)
```

Istruzioni di salto

- return
- goto
- break
- continue
- exit

return

return [<espressione>]

Consente di riprendere l'esecuzione del codice dal punto in cui si e' chiamata una funzione

Si puo utilizzare piu' di un return in una funzione

Una funzione void puo non avere return

goto

goto <ethichetta>;

<ethichetta>: <istruzione>;

Consente di alterare l'ordine di esecuzione del codice.

break

break;

Consente di alterare l'ordine di esecuzione del codice.

Termina l'esecuzione del blocco di istruzioni che si sta eseguendo

continue

continue;

Consente di alterare l'ordine di esecuzione del codice.

Termina l'esecuzione corrente di un ciclo

exit

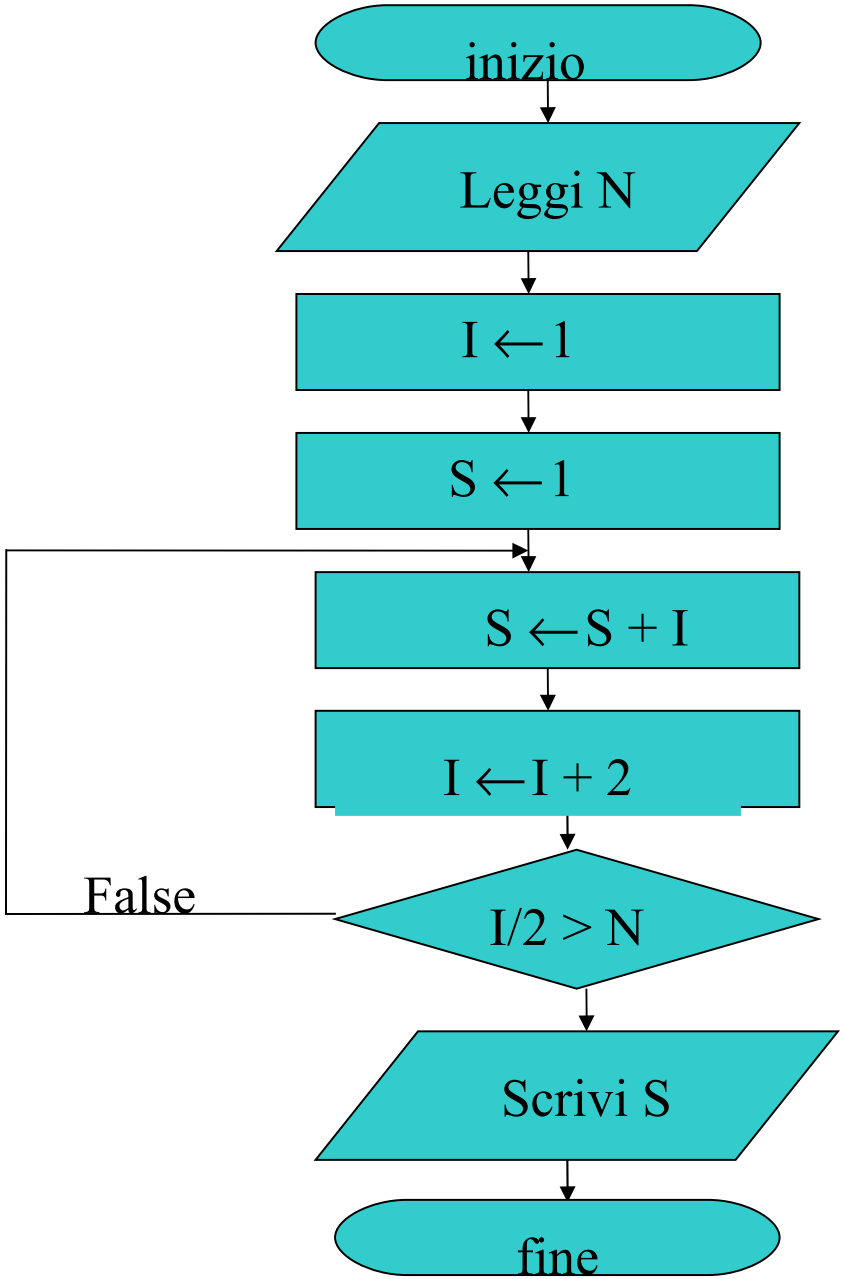
exit();

Consente di alterare l'ordine di esecuzione del codice.

Termina l'esecuzione del programma

In realtà e' una funzione della libreria standard

Sommare i primi N numeri dispari




```
# include <stdio.h>

int  main ()
     {int N, I, S;
      scanf("%d" &N);
      I =1;
      S=1;
      do
         {  S =S + I;
            I = I +2;
         } while (N<I*2);
      printf("somma =&d",S);
     }
```

Calcolare la media dei numeri pari presenti in una sequenza di N numeri

```
#include <stdio.h>
int main()
{
    int N, i, numero, somma;
    float media;
    somma = i = 0;
    scanf("%d", &N);
    while (N != 0)
    {
        scanf("%d", &numero);
        N = N -1
        if (numero%2) continue;
        somma = somma + numero;
        i = i +1
    }
    media = ((float)somma)/i;
    printf("La somma dei numeri digitati è: %d\n", somma);

}
```

Calcolare la media dei numeri pari presenti in una sequenza di N numeri

```
#include <stdio.h>
int main()
{
	int N, i, numero, somma;
	float media;
	somma = i = 0;
	scanf("%d", &N);
	for (;N != 0;N--)
	{
		scanf("%d", &numero);
		if (numero%2) continue;
		somma = somma + numero;
		i = i +1
	}
	media = ((float)somma)/i;
	printf("La somma dei numeri digitati è: %d\n", somma);

}
```

Calcolare la media dei numeri pari presenti in una sequenza chiusa da 0 di al più N numeri

```
#include <stdio.h>
int main()
{
	int N, i, numero, somma;
	float media;
	somma = i = 0;
	scanf("%d", &N);
	while (N != 0)
	{
		scanf("%d", &numero);
		N - - ;
		if (numero==0) break;
		if (numero%2) continue;
		somma = somma + numero;
		i + +;
	}
	media = ((float)somma)/i;
	printf("La somma dei numeri digitati è: %d\n", somma);

}
```

Calcolare del fattoriale di N

```
include <stdio.h>
int main()
{
      int N,Fatt;
      do{
            scanf("%d", &N);
      }
      while (N <0);
      if (N !=0)
            for (fatt = 1; N !=0 ; N- - ) fatt = fatt * N;
      else
            fatt = 1;
      printf(”Fattoriale =%d\n", fatt);

}
```