

C - Strutture di controllo

Sequenza - Blocchi

- Le parentesi graffe vengono usate per racchiudere istruzioni che formano una **istruzione composta** o **blocco**. Un blocco é sintatticamente equivalente ad un'unica istruzione.

Esempio:

```
{  
  {  
    a=10;  
    {  
      b=2;  
    }  
  }  
}
```

Selezione: if

`if (espr) istruzione`

Viene valutata `espr`.

1. Se `espr` é diversa da 0: viene eseguita `istruzione`.
2. Se `espr` é uguale a 0: l'esecuzione riprende dall'istruzione successiva all'`if`.

```
if (a>=0)
    area = a * a;
printf("Area: %d", area);
```

if osservazioni

Solo la prima istruzione fa parte del corpo dell'istruzione `if`. Quindi se vogliamo che l'istruzione `printf()` venga eseguita solo se la condizione é verificata dobbiamo introdurre un'istruzione di blocco.

```
if (a >= 0)
{
    area = a * a;
    printf("Area: %d", area);
}
```

Selezione: `if-else`

```
if (espr) istruzione1  
else istruzione2
```

Viene valutata `espr`.

- Se `espr` é diversa da 0: viene eseguita `istruzione1`.
- Se `espr` é uguale a 0: viene eseguita `istruzione2`.

```
if (a >= 0)  
    area = a * a;  
else  
    printf("Il valore di a é negativo");
```

if-else osservazioni

Se si devono eseguire più istruzioni nel corpo dell'`if`, queste devono essere poste all'interno di un'istruzione di blocco.

Ad esempio:

```
if (a >= 0)
    area = a * a;
    printf("Area: %d", area);
else
    printf("Il valore e' negativo");
```

Si ottiene un errore in compilazione.

Si deve scrivere:

```
if (a >= 0)
{
    area = a * a;
    printf("Area: %d", area);
}
else printf("Il valore e' negativo");
```

if-else innestati

Un `else` è sempre associato all'`if` più vicino

```
if (a >= 0)          /* if1 */
    if (b >= 0)      /* if2 */
        printf("a>= 0 e b>=0");
else printf("Il valore di a e' negativo");
```

Il programma non funziona come si desidera perché `else` é associato a `if2` e non a `if1`.

Occorre scrivere:

```
if (a >= 0)
{
    if (b >= 0)
        printf("a>= 0 e b>=0");
}
else printf("Il valore di a e' negativo");
```

Esercizio: Trovare l'errore

```
int main(void)
{
    int a, b;
    a = 3;
    b = 4;
    if (a = b)
        printf("uguali\n");
    else
        printf("diversi\n");
    return 0;
}
```

Esempio: Radici di un equazione di secondo grado

```
#include <stdio.h>
#include <math.h>
int main(void)
{
    double a, b, c, delta;
    printf("Inserisci i coefficienti");
    scanf("%lf%lf%lf", &a, &b, &c);
    delta = b * b - 4 * a * c;
    if (delta < 0) printf("Non ci sono radici reali\n");
    else
        if (delta == 0) printf("Radici coincidenti: %f\n", (-b) / (2 * a));
        else
        {
            printf("X1 = %f\n", (-b - sqrt(delta)) / (2 * a));
            printf("X2 = %f\n", (-b + sqrt(delta)) / (2 * a));
        }
    return 0;
}
```

Esecuzione

Supponiamo di dare in ingresso i seguenti valori 1, 3 e 2.

Ciclo while

`while (espr) istruzione`

1. Viene valutata `espr`.
 - (a) Se `espr` é diversa da 0 (vera) viene eseguita istruzione e si ritorna al punto 1.
 - (b) Se `espr` é uguale a 0 (falsa) il ciclo termina e si esegue l'istruzione che segue il ciclo `while`.

while esempio

```
#include <stdio.h>
int main(void)
{
    int count;
    count=0;
    while (count < 3)
    {
        printf("Il valore di count e' %d\n", count);
        count = count + 1;
    }
    return 0;
}
```

Ciclo for

```
for (espr1;espr2;espr3)  
  istruzione
```

1. Viene valutata espr₁ (inizializzazione del ciclo);
2. Viene valutata espr₂.
 - (a) Se espr₂ è diversa da 0:
 - viene eseguita istruzione
 - viene valutata espr₃ (tipicamente un incremento)
 - si ritorna al punto 2.
 - (b) Se espr₂ è uguale a 0 il ciclo termina e si esegue l'istruzione che segue il ciclo for.

Si noti che espr₁ viene valutata una sola volta.

for esempio

```
for(i=1; i<= 3; i=i+1)
    printf("%d ",i);
printf("Fine del ciclo for ");
```

while VS for

Il costrutto for equivale a:

```
espr1;  
while (espr2)  
{  
    istruzione  
    espr3;  
}
```

Alcune o tutte le espressioni di un for possono essere omesse, ma in ogni caso i ';' devono essere presenti. Se manca espr_2 si assume che tale condizione sia diversa da 0 (sempre vera).

```
i=1;  
sum=0;  
for(; i<= 10;)  
    sum = sum + i;
```

Tavola di verità dei connettivi logici

```
#include <stdio.h> int main(void)
{
    int b1, b2;
    printf("b1 b2 !b1!b2 b1&&b2 b1||b2\n\n");
    for (b1 = 0; b1 <= 1; b1=b1+1)
        for (b2 = 0; b2 <= 1; b2=b2+1)
            printf("%d %d %d %d %d %d\ n", b1, b2, !b1, !b2, b1&&b2, b1||b2);
    return 0;
}
```

Ciclo do-while

```
do  
    istruzione  
while(espr);
```

1. viene eseguita istruzione
2. viene valutata espr
 - (a) se espr é diversa da 0 (vera) si ritorna al quindi si torna al punto 1.
 - (b) se espr é uguale 0 il controllo passa all'istruzione che segue il ciclo do-while.

Esempio

```
#include <stdio.h>
int main(void)
{
    int i;
    i = 0;
    do
    {
        printf("Il valore di i e' %d\ n", i);
        i=i+1;
    }
    while (i < 3);
    return 0;
}
```

break

L'istruzione `break`; causa l'immediata interruzione del ciclo (`for`, `while` e `do-while`) più interno che la contiene.

Esercizio:

```
A=77;  
for (i=2; i <= A; i=i+1)  
    if (A%i == 0)  
        break;  
printf("Il minimo di A é %d\n", i);
```

L'output del programma é ...

Esempio

```
for (A=2; A<=10; A=A+1)
{
    for (i=2; i <= A; i=i+1)
        if (A%i == 0)
            break;
    printf("Il minimo di %d é %d\n", A, i);
}
```

L'output del programma é ...