

Programmazione e Laboratorio di Programmazione

Manualistica 3 Le strutture di controllo

Le strutture di controllo

- **Strutture di controllo:**

alterano l'ordine sequenziale secondo il quale vengono eseguite le istruzioni del programma

- **Blocco di istruzioni:**

```
{  
istruzione1;  
istruzione2;  
....  
istruzionek;  
}
```

- **istruzione_i :**

- a) singola istruzione

- b) blocco di istruzioni

se il blocco contiene una sola istruzione, le parentesi graffe possono essere omesse

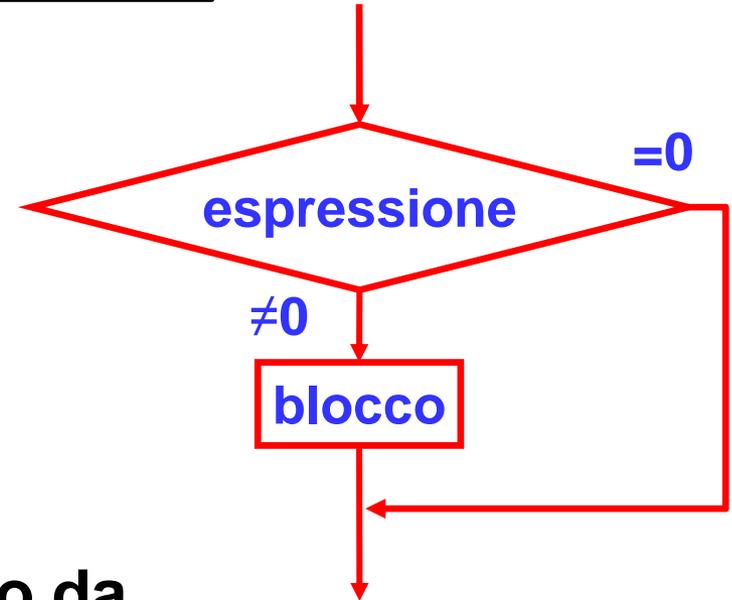
Il costrutto if

- **Sintassi:**

intestazione → if (espressione)
corpo → blocco;

- **Comportamento:**

- valuta il valore di **espressione**
- se tale valore è diverso da **0** esegue il **blocco** di istruzioni
- altrimenti esegue la **istruzione successiva al costrutto if**



Il costrutto if

- **Esempio:**

```
/* sorgente: if.c */  
#include <stdio.h>  
  
/* funzione che calcola il massimo  
** tra 2 numeri interi */  
int massimo (int num1, int num2)  
{  
    if (num1 > num2)  
        return (num1);  
    return (num2);  
}
```

Il costrutto if

```
/* chiamante */
```

```
int main ()
```

```
{
```

```
int A, B;
```

```
/* acquisizione del valore delle variabili */
```

```
printf("\\nDammi il I intero: ");
```

```
scanf ("%d", &A);
```

```
printf("\\nDammi il II intero: ");
```

```
scanf ("%d", &B);
```

```
/* restituzione del massimo tra i 2 valori */
```

```
printf("\\nIl massimo e': %d", massimo(A, B));
```

```
return(0);
```

```
}
```

Il costrutto if-else

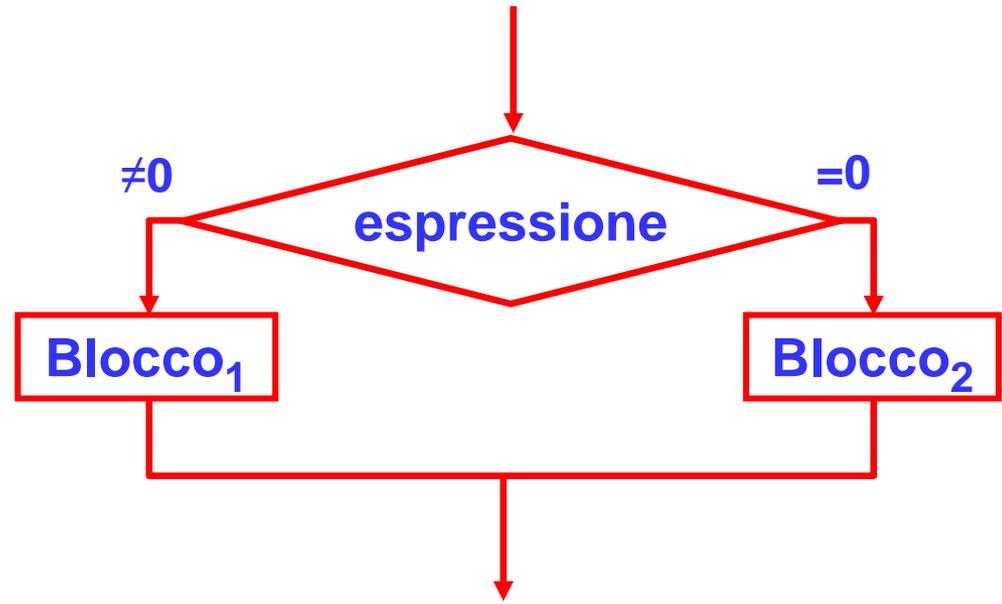
- **Sintassi:**

if(espressione)

 blocco₁

else

 blocco₂;



- **Comportamento:**

a) valuta il valore di **espressione**

b) se tale valore è diverso da **0** esegue **blocco₁**

c) altrimenti esegue **blocco₂**

Il costrutto if-else

- **Esempio:**

```
/* sorgente: if_else.c */  
#include <stdio.h>  
  
/* funzione che calcola il massimo  
** tra 2 numeri interi */  
int massimo (int num1, int num2)  
{  
    if (num1 > num2)  
        return (num1);  
    else  
        return (num2);  
}
```

```
if (num1 > num2)  
    return (num1);  
return (num2);
```

Il costrutto while

- **Sintassi:**

while (espressione)

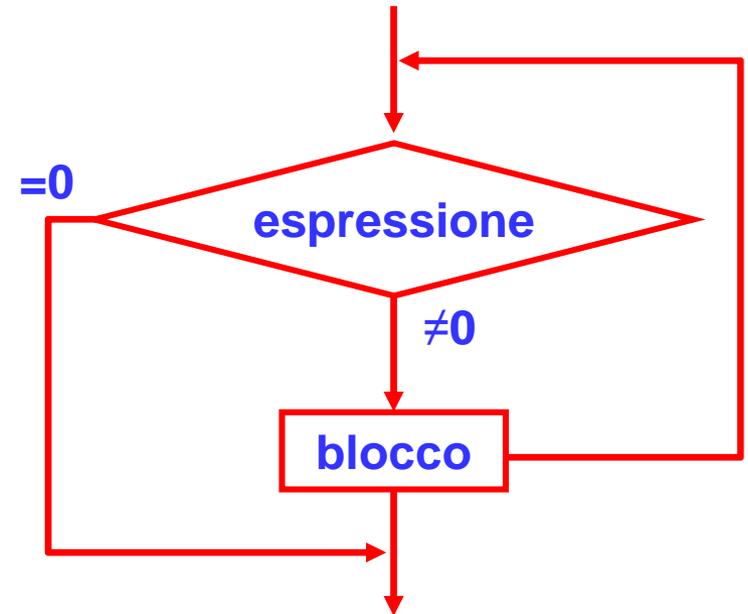
blocco;

- **Comportamento:**

a) valuta il valore di **espressione**

b) se tale valore è diverso da **0** esegue **blocco**
e torna al punto a)

c) altrimenti esegue l'istruzione
immediatamente successiva al costrutto



Il costrutto while

• Esempio:

```
/* sorgente: while.c */
#include <stdio.h>
/* funzione che acquisisce una sequenza di valori interi non negativi
** terminata da un valore negativo e calcola la loro somma */
int somma_pos()
{
    /* definizione e inizializzazione delle variabili */
    int somma, numero;
    somma = numero = 0;
    /* continua ad acquisire valori e ad aggiornare la somma parziale
    ** fino all'acquisizione del primo valore negativo */
    while (numero >= 0)
    {
        printf("\nDammi il prossimo numero intero: ");
        scanf("%d", &numero);
        somma = somma + numero;
    };
    /* restituisce la somma di tutti i valori acquisiti */
    return(somma);
}
```

Il costrutto while

```
/* chiamante */  
int main ()  
{  
  /* visualizza la somma di tutti i numeri acquisiti */  
  printf (“\nLa somma e’: %d”, somma_pos());  
  return(0);  
}
```

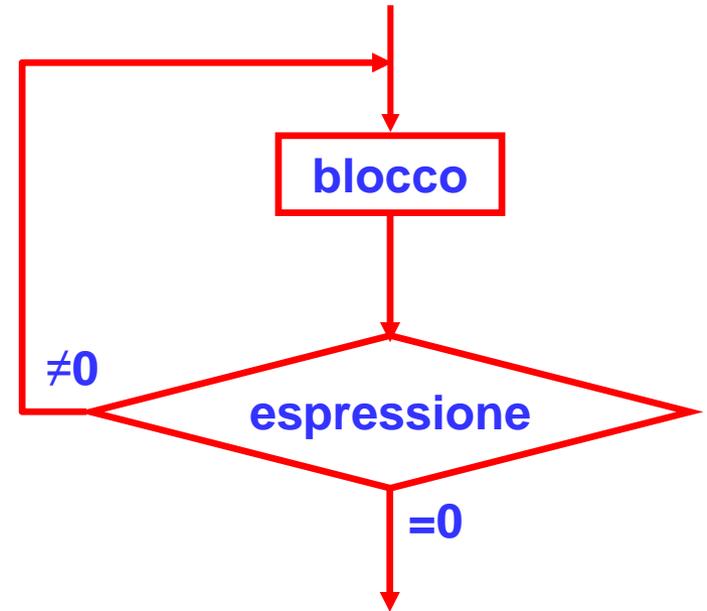
Il costrutto do-while

- **Sintassi:**

do

blocco

while (espressione);



- **Comportamento:**

a) esegue **blocco**

b) valuta il valore di **espressione** e se tale valore è diverso da **0** torna al punto a)

c) altrimenti esegue l'istruzione immediatamente successiva al costrutto

Il costrutto do-while

- **Esempio:**

```
/* sorgente: do_while.c */
#include <stdio.h>
/* funzione che acquisisce una sequenza di valori interi non negativi
** terminata da un valore negativo e calcola la loro somma */
int somma_pos()
{
    /* definizione e inizializzazione delle variabili */
    int somma, numero;
    somma = 0; somma = numero = 0;
    /* continua ad acquisire valori e ad aggiornare la somma parziale
    ** fino all'acquisizione del primo valore negativo */
    do {
        printf("\nDammi il prossimo numero intero: ");
        scanf("%d", &numero);
        somma = somma + numero;
    }
    while (numero >= 0);
    // restituisce la somma
    return(somma);
}
```

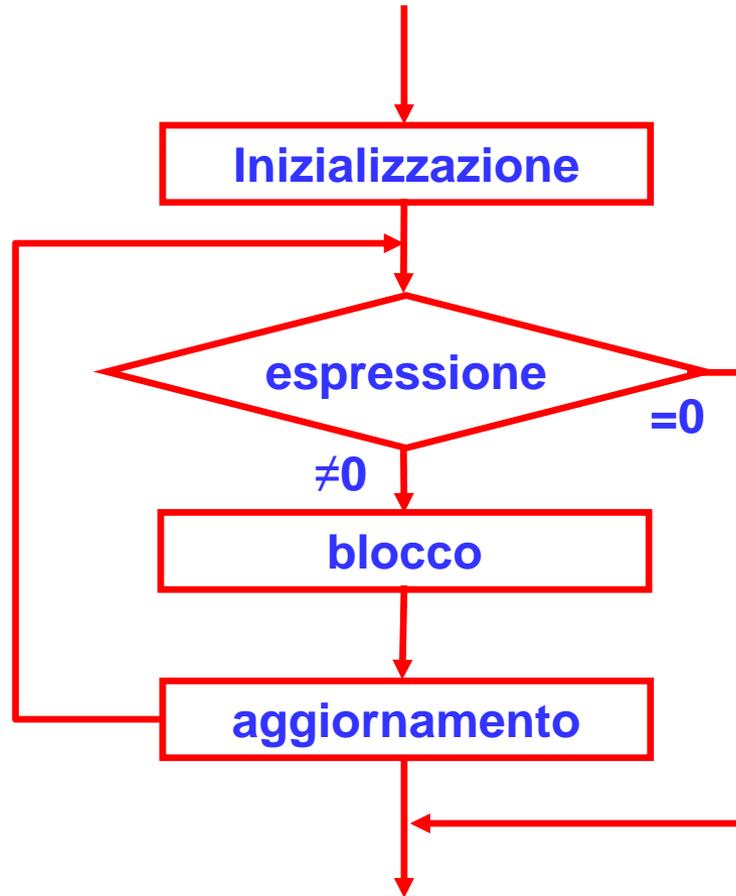
```
while (numero >= 0)
{
    printf("\nDammi il prossimo numero intero: ");
    scanf("%d", &numero);
    somma = somma + numero;
};
```

Il costrutto for

- **Sintassi:**
for (inizializzazione; espressione; aggiornamento)
blocco;
- **Comportamento:**
 - a) esegue **inizializzazione**
 - b) valuta il valore di **espressione**
 - c) se tale valore è diverso da **0**
 - c.1) esegue **blocco**
 - c.2) esegue **aggiornamento**
 - c.3) torna al punto b)
 - d) altrimenti esegue l'istruzione immediatamente successiva al costrutto

Il costrutto for

- **Comportamento:**



Il costrutto for

- **Esempio:**

```
/* sorgente for.c */  
#include <stdio.h>  
/* funzione per il calcolo della somma dei primi n numeri interi */  
int somma_n (int n)  
{  
    /* definizione e inizializzazione della variabili */  
    int somma, num;  
    somma = 0;  
    /* calcola la somma dei primi n numeri interi */  
    for (num = 1; num <= n; ++num)  
        somma += num;  
    /* restituisce tale somma */  
    return(somma);  
}
```

Il costrutto for

```
/* chiamante */  
int main ()  
{  
    /* definizione e acquisizione delle variabili */  
    int numero;  
    printf("\\nSpecificare il valore di n: ");  
    scanf("%d", &numero);  
  
    /* visualizza la somma dei primi n numeri interi */  
    printf ("\\nSomma dei primi %d numeri: %d", numero,  
    somma_n(numero));  
    return(0);  
}
```

Il costrutto switch-case

- **Sintassi:**

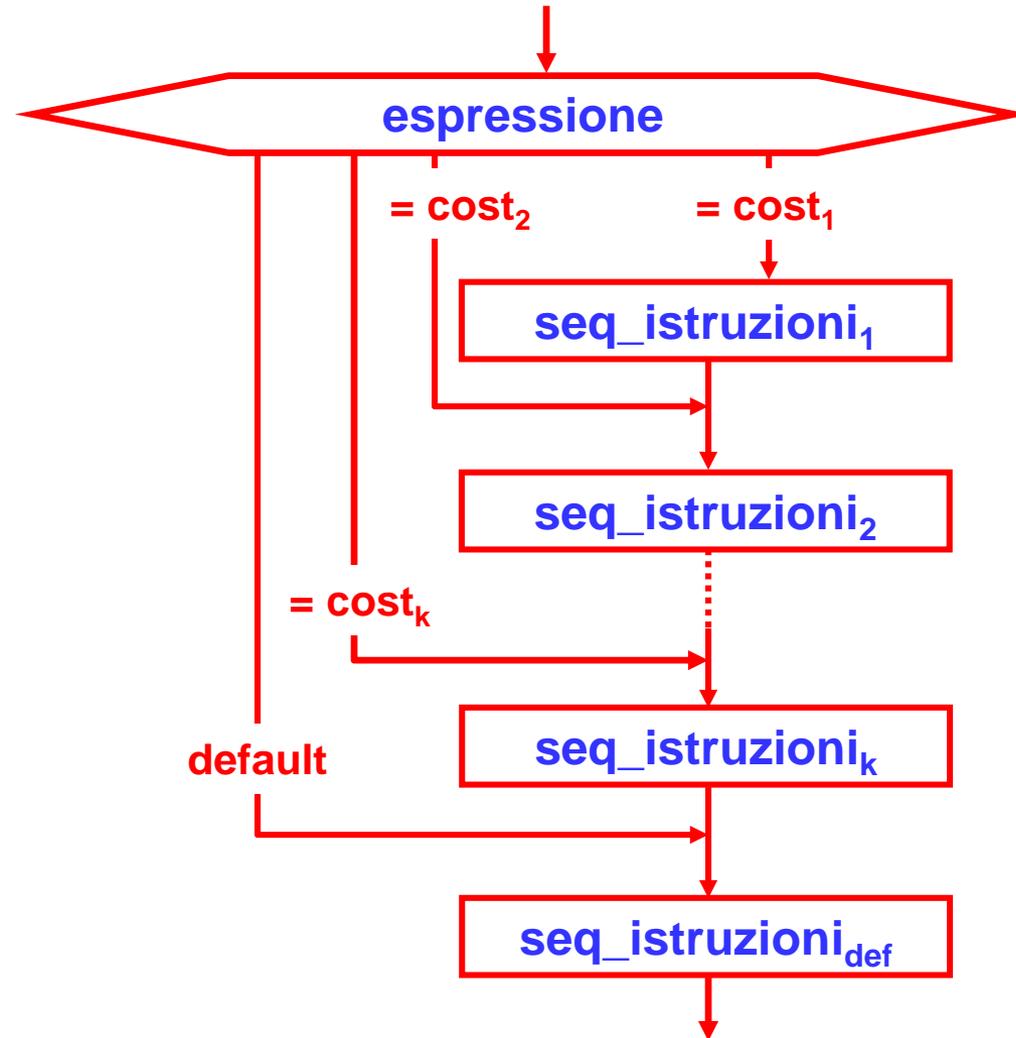
```
switch (espressione)
{
  case cost1:
    seq_istruzioni1;
  case cost2:
    seq_istruzioni2;
  ....
  case costk:
    seq_istruzionik;
  default:
    seq_istruzionidef;
};
```

- **Comportamento:**

- a) valuta il valore di **espressione**
- b) se il valore è **cost_i**, riprende l'esecuzione del corpo dalla prima istruzione di **seq_istruzioni_i**
- c) altrimenti, se il suo valore è diverso da **cost_i**, per ogni $i = 1, 2, \dots, k$, riprende l'esecuzione del corpo dalla prima istruzione di **seq_istruzioni_{def}**

Il Costrutto switch-case

- **Comportamento:**



Il costrutto break

- **Sintassi:**

break;

- **Può comparire all'interno del corpo:**

- del costrutto **while**
- del costrutto **do-while**
- del costrutto **for**
- del costrutto **case**

- **Comportamento:**

forza l'uscita dal corpo del costrutto, o, in altre parole l'esecuzione della prima istruzione a questo successiva

I costrutti switch-case e break

- **Esempio:**

```
/* sorgente: switch_case.c */
#include <stdio.h>
/* funzione che implementa lo scheletro di un menu di scelta */
void menu()
{
    /* definizione e inizializzazione della variabile */
    /** che permette l'uscita dal programma */

    int quit = 0;
    /* rimane nel ciclo fino a quando tale variabile
    /** non viene settata a 1 */

    while(!quit)
    {
        /* variabile che memorizza la selezione */

        int selezione;
        /* visualizza le possibili scelte

        printf("\nSelezionare Funzionalita"); */
        printf("\nFunzione A: 1");
        printf("\nFunzione B: 2");
        printf("\nUscita: 3");
        printf("\nSelezione: ");
    }
}
```

I costrutti switch-case e break

```
/* acquisisce la scelta */  
scanf("%d", &selezione);  
// discrimina tra le diverse scelte  
switch (selezione)  
{  
  case 1:  
    printf("Hai selezionato la funzione A\n");  
    break;  
  case 2:  
    printf("Hai selezionato la funzione B\n");  
    break;  
  case 3:  
    quit = 1;  
    break;  
  default:  
    /* selezione errata */  
    printf("Selezionare 1, 2 o 3");  
    break;  
};  
};  
}
```

I costrutti switch-case e break

```
/* chiamante */  
int main()  
{  
  /* chiama la funzione che implementa lo scheletro di un menu di  
  ** scelta */  
  menu();  
  return(0);  
}
```

Il costrutto continue

- **Sintassi:**
continue;
- **Può comparire all'interno del corpo:**
 - del costrutto **while**
 - del costrutto **do-while**
 - del costrutto **for**
- **Comportamento:**
interrompe l'esecuzione del corpo del costrutto, la cui esecuzione riprende dalla valutazione dell'**espressione**

I costrutti while e continue

- **Esempio:**

```
/* sorgente: continue.c */
#include <stdio.h>
/* funzione che visualizza i primi n numeri pari */
void primi_n_pari(int n)
{
    /* definizione e inizializzazione delle variabili */
    int cont;
    cont = 1;

    /* generazione dei primi 2*n numeri interi e
    ** selezione e visualizzazione dei soli pari */
    while ((cont++) <= (2*n))
    {
        /* tralascia i dispari */
        if ((cont % 2) == 1)
            continue;

        /* visualizza i pari */
        printf ("\n%d pari: %d\n", cont/2, cont);
    }
}
```

I costrutti while e continue

```
/* chiamante */  
int main ()  
{  
    /* definizione e acquisizione della variabile che  
    ** memorizza il numero degli interi pari da visualizzare */  
    int n;  
    printf("\\nSpecificare il valore di n: ");  
    scanf("%d", &n);  
    /* chiamata della funzione che genera i primi  
    ** n numeri pari */  
    primi_n_pari(n);  
    return(1);  
}
```