



Fondamenti di Informatica
Ingegneria Clinica
Lezione 20/11/2009



Prof. Raffaele Nicolussi
FUB - Fondazione Ugo Bordoni
Via B. Castiglione 59 - 00142 Roma



Docente	Raffaele Nicolussi	rnicolussi@fub.it 0654803323
Lezioni Aula 54 (ex aula 4) Via del Castro Laurenziano, 7	Lunedì, Giovedì, Venerdì	12:00 – 13:30
Esercitazioni Aula 15 Via Tiburtina, 205	Giovedì	14:00 – 16.30
Ricevimento:	Per appuntamento	in FUB, per email, per telefono
Sito web:	http://w3.uniroma1.it/IngClinFondinf	



Strutture di controllo

- FOR
- WHILE
- ...
- Esercizi

Strutture di controllo

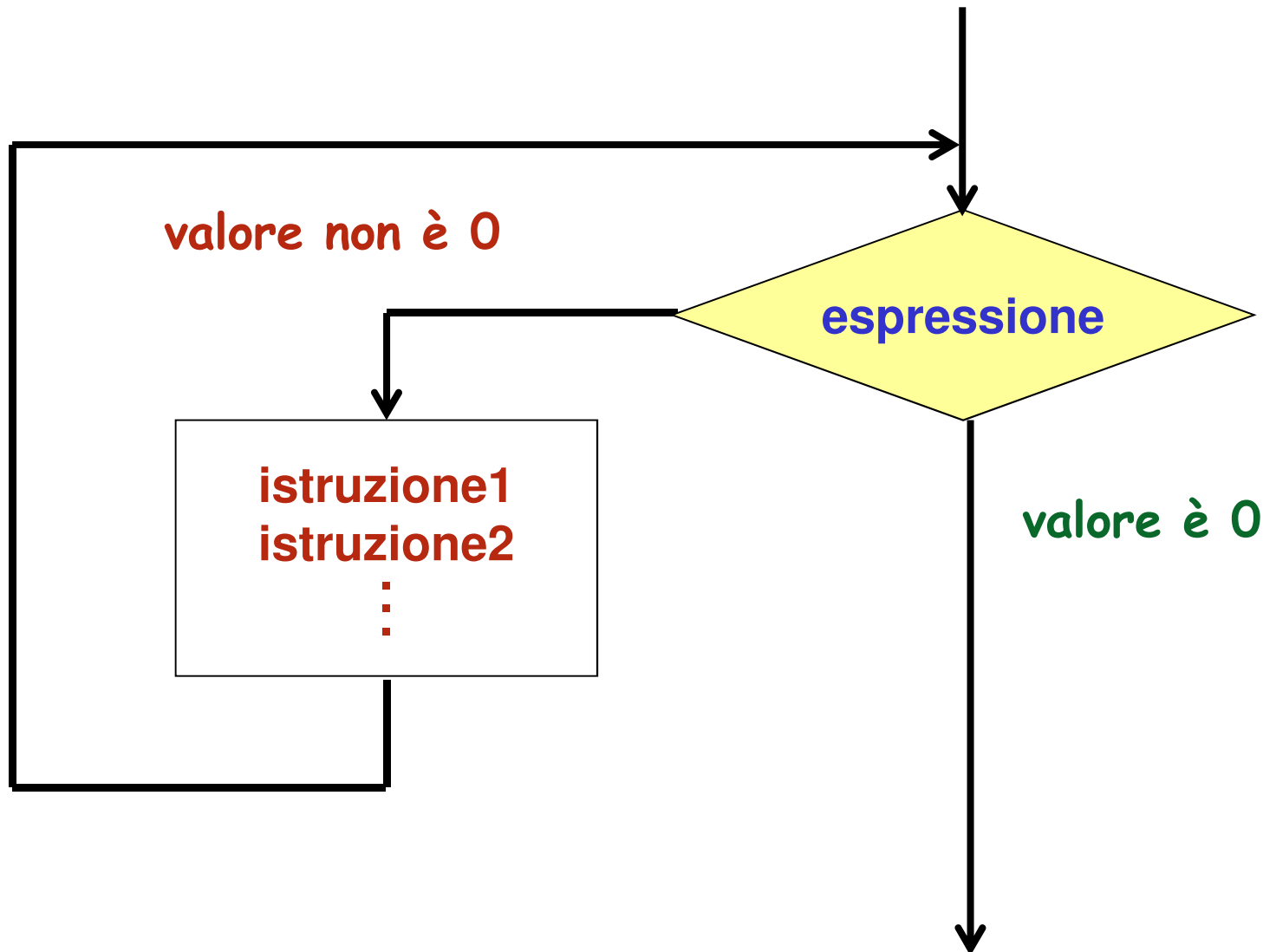


- ❑ Modifiche condizionate del flusso di esecuzione delle istruzioni
- ❑ Strutture di **selezione**: **ramificazione condizionata** del flusso (istruzioni **if** – selezione singola, **if else** – selezione doppia, **switch** – selezione multipla)
- ❑ Strutture di **iterazione**: **ripetizione condizionata** di un blocco di istruzioni (istruzioni **while** e **for**)



C: Istruzione **while**

```
while ( espressione ) { istruzione1 ; istruzione2 ; ... ; }
```



C: istruzione for



```
for ( espressione1; espressione2; espressione3 )  
    { istruzione1 ; istruzione2 ; ... ; }
```

- **espressione1** viene preliminarmente valutata
- se **espressione2** ha valore **diverso da 0**, si esegue il corpo del ciclo **istruzione1 ; istruzione2 ; ... ;**
- **espressione3** è valutata ogni volta che viene eseguito il corpo del ciclo, dopo di questo e prima di rivalutare **espressione2**
- Il ciclo termina quando **espressione2** ha valore **0**

Esempio uso ciclo for



- **espressione1** inizializza una variabile contatore
- **espressione3** aggiorna il valore della variabile contatore
- **espressione2** verifica se la variabile contatore ha raggiunto il valore finale
- Il **corpo del ciclo** viene così eseguito un numero di volte che è determinabile da **espressione1**, **espressione2** ed **espressione3**

Esempio: Stampa numeri da 1 a 100

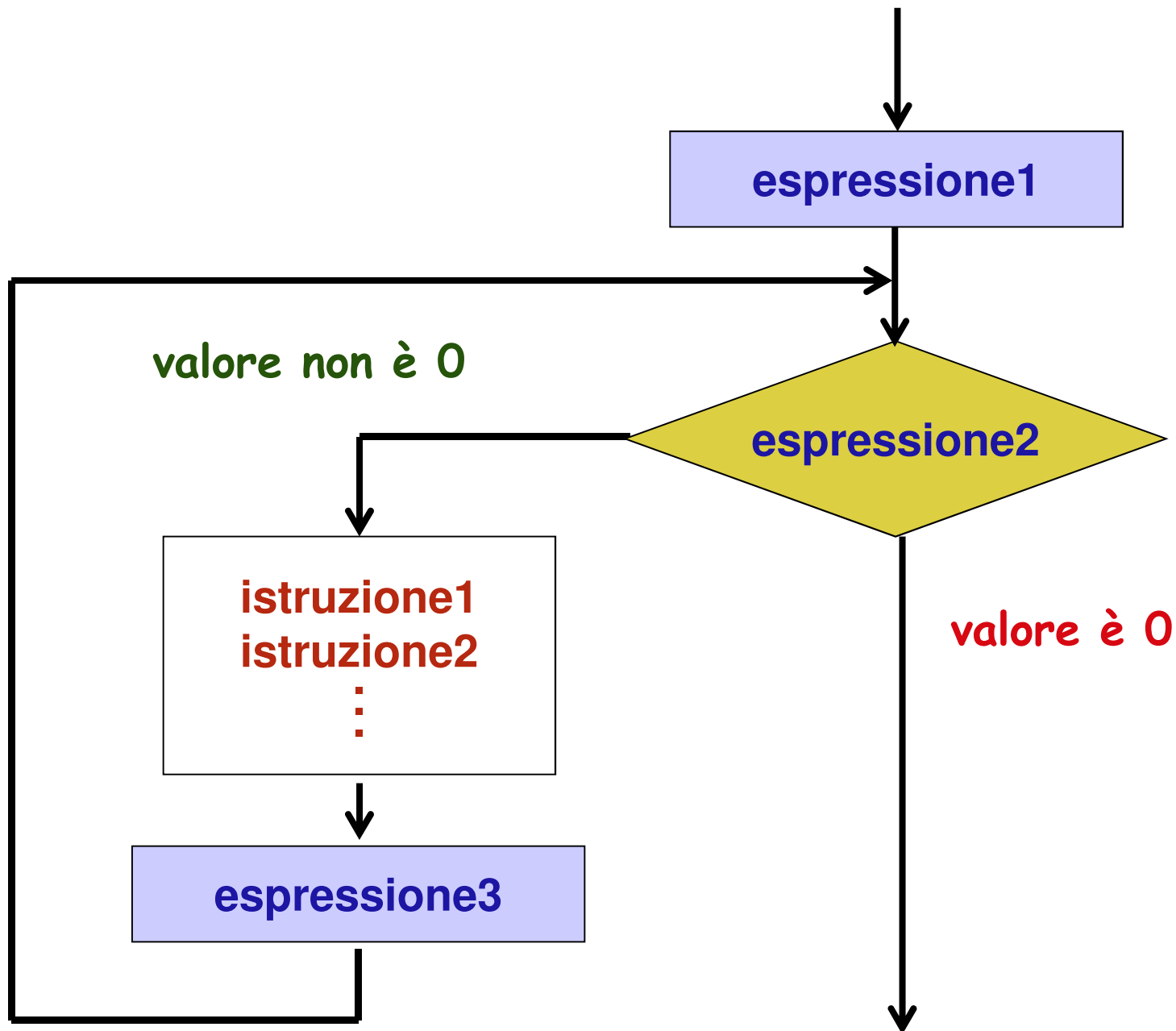


```
int i ;  
for ( i = 1 ; i <= 100 ; i = i + 1 )  
    printf( "%d", i ) ;
```

equivale a

```
int i ;  
i = 1 ;  
while ( i <= 100 ) {  
    printf( "%d", i ) ;  
    i = i + 1 ;  
}
```


C: Confronto for-while



Istruzione for



Sintassi:

for (*inizializzazione; condizione; incremento*)

istruzione

Semantica: è equivalente a

{

inizializzazione;

while (*condizione*) {

istruzione

incremento;

}

}



Istruzione for (2)

- ❑ Ciascuna delle tre parti del **for** (*inizializzazione*, *condizione* e *incremento*) può essere omessa
- ❑ In questo caso i ";" vanno messi lo stesso
- ❑ Se manca *condizione*, viene assunta pari a **true**



Istruzione for (3)

- La sintassi del **for** permette che le tre parti siano delle espressioni qualsiasi, purché *inizializzazione*; e *incremento*; siano delle istruzioni
- Nell'uso del ciclo **for** è buona norma:
 - usare le tre parti del for con riferimento ad una *variabile di controllo*
 - non modificare la variabile di controllo nel corpo del ciclo
- In generale, inizializzazione e/o incremento possono essere una sequenza di espressioni separate da ,
 - Questo permette di inizializzare e/o incrementare più variabili contemporaneamente



Tanti modi di scrivere il for

- ❑ Calcola e stampa le prime 10 potenze di 2
- ❑ Possiamo scrivere il for in diversi modi, tutti esatti

```
int i, potDi2;
```

```
for (i = 0, potDi2 = 1; i < 10; i++, potDi2 *= 2)
```

```
    printf ("2 alla %d = %d", i, potDi2);
```



Tanti modi di scrivere il for

Inizializzazioni tutte dentro la for

```
for (int i = 0, potDi2 = 1; i < 10; i++, potDi2 *= 2)
    printf ("2 alla %d = %d", i, potDi2);
```



Tanti modi di scrivere il for

Inizializzazioni tutte fuori la for

```
int i = 0, potDi2 = 1;  
for (; i < 10; i++, potDi2 *= 2)  
    printf ("2 alla %d = %d", i, potDi2);
```

Operazioni dentro il corpo della for

```
for (int i = 0, potDi2 = 1; i < 10; i++) {  
    potDi2 *= 2;  
    printf ("2 alla %d = %d", i, potDi2);  
}
```



Ciclo for: esempi

for (int i = 1; i <= 10; i++) ...

valori assunti da i: 1, 2, 3, ..., 10

for (int i = 10; i > 0; i--) ...

valori assunti da i: 10, 9, 8, ..., 2, 1

for (int i = -4; i <= 4; i = i+2) ...

valori assunti da i: -4, -2, 0, 2, 4

for (int i = 0; i >= -10; i = i-3) ...

valori assunti da i: 0, -3, -6, -9



Esempio

- Stampare il codice numerico dei caratteri compresi tra 33 e 85

```
for (int i = 33; i <= 85; i++)  
    printf("i = %d → c = %c", i , i);
```

- $i = 33 \rightarrow c = !$
- $i = 34 \rightarrow c = "$
- $i = 35 \rightarrow c = \#$
- $i = 36 \rightarrow c = \$$
- $i = 37 \rightarrow c = \%$
- $i = 38 \rightarrow c = \&$
- $i = 39 \rightarrow c = '$
- $i = 40 \rightarrow c = ($
- $i = 41 \rightarrow c =)$
- $i = 42 \rightarrow c = *$
- $i = 43 \rightarrow c = +$
- $i = 44 \rightarrow c = ,$
- $i = 45 \rightarrow c = - \backslash$
- $i = 46 \rightarrow c = .$
- $i = 47 \rightarrow c = /$
- $i = 48 \rightarrow c = 0$
- $i = 49 \rightarrow c = 1$

- $i = 51 \rightarrow c = 3$
- $i = 52 \rightarrow c = 4$
- $i = 53 \rightarrow c = 5$
- $i = 54 \rightarrow c = 6$
- $i = 55 \rightarrow c = 7$
- $i = 56 \rightarrow c = 8$
- $i = 57 \rightarrow c = 9$
- $i = 58 \rightarrow c = :$
- $i = 59 \rightarrow c = ;$
- $i = 60 \rightarrow c = <$
- $i = 61 \rightarrow c = =$
- $i = 62 \rightarrow c = >$
- $i = 63 \rightarrow c = ?$
- $i = 64 \rightarrow c = @$
- $i = 65 \rightarrow c = A$
- $i = 66 \rightarrow c = B$
- $i = 67 \rightarrow c = C$

- $i = 68 \rightarrow c = D$
- $i = 69 \rightarrow c = E$
- $i = 70 \rightarrow c = F$
- $i = 71 \rightarrow c = G$
- $i = 72 \rightarrow c = H$
- $i = 73 \rightarrow c = I$
- $i = 74 \rightarrow c = J$
- $i = 75 \rightarrow c = K$
- $i = 76 \rightarrow c = L$
- $i = 77 \rightarrow c = M$
- $i = 78 \rightarrow c = N$
- $i = 79 \rightarrow c = O$
- $i = 80 \rightarrow c = P$
- $i = 81 \rightarrow c = Q$
- $i = 82 \rightarrow c = R$
- $i = 83 \rightarrow c = S$
- $i = 84 \rightarrow c = T$
- $i = 85 \rightarrow c = U$





Esempio di ciclo for: calcolo del fattoriale

Fattoriale (n) = n(n-1)(n-2) 1

fattoriale (0) = 1

fattoriale (n) = n * fattoriale (n-1)

□ *Esempio:* fattoriale (4) = 4 * 3 * 2 * 1

Esempio: calcolo del fattoriale



```
#include <stdio.h>
```

```
int main () {  
    int n, i ;  
    unsigned long int p = 1;  
  
    printf("Inserire numero intero maggiore di zero: ");  
    scanf("%d", &n);  
    for ( i = 1 ; i <= n ; i++)  
        p = p* i;  
    printf("\n%d ! = %lu\n", n, p);  
    return 0;  
}
```



Esempio di ciclo for :: stampa dei pari

- Leggere due interi n ed m e stampare tutti i numeri pari compresi tra n ed m



Esempio di ciclo for :: stampa dei pari

```
leggi (n,m);  
int i;  
for (i = n; i <= m; i++)  
    if (i%2==0) printf(“%d ”, i);
```

Esercizio da fare: piramide di asterischi



- ❑ Leggere un intero compreso fra 1 e N e stampare una piramide di asterischi di altezza pari h al numero letto

❑ **Asterischi sulla riga:** $\text{numero_riga} * 2 - 1$

❑ 1 nella prima riga

❑ 3 nella seconda riga

❑ 5 nella terza riga

❑ 7 nella quarta

❑ 9 nella quinta

❑ 11 nella sesta

❑ **Spazi bianchi:** $h - \text{numero_riga}$

❑ 5 nella prima riga

❑ 4 nella seconda riga

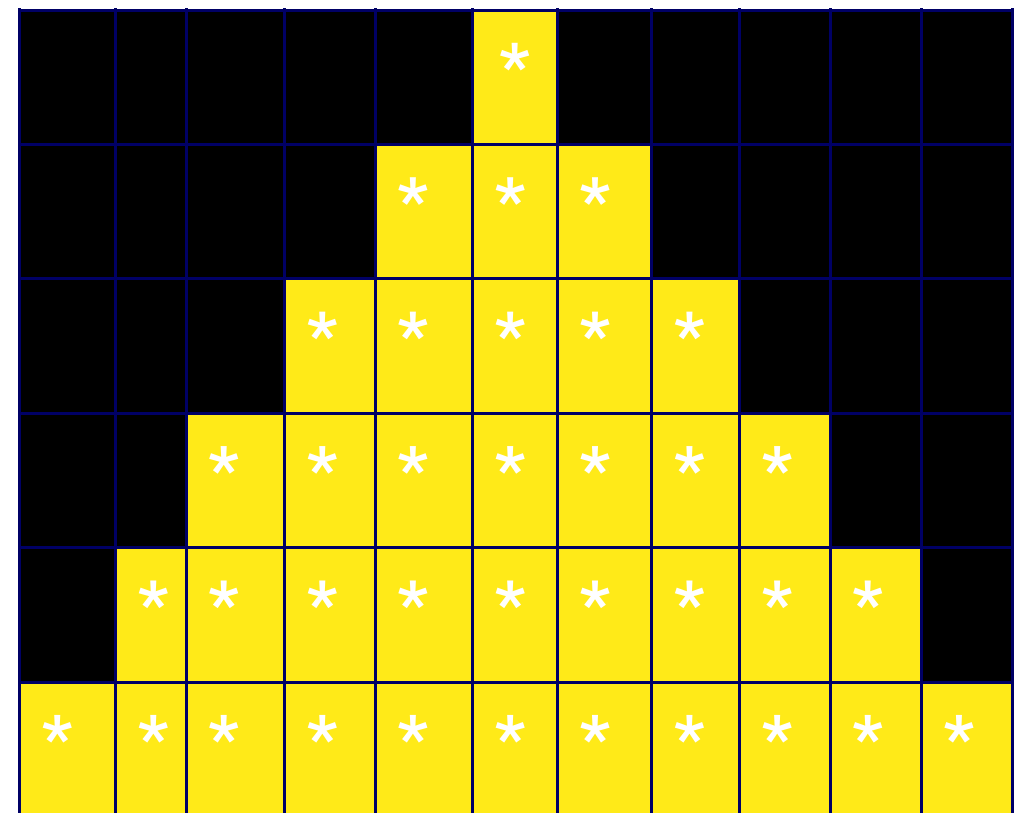
❑ 3 nella terza riga

❑ 2 nella quarta

❑ 1 nella quinta

❑ 0 nella sesta

N=6





Finché il numero delle righe non è pari all'altezza data

1. Stampa gli spazi
2. Stampa gli asterischi
3. Vai a capo

1. Stampa gli spazi

$i=1$

Finché i è minore o uguale a $(\text{altezza}-\text{riga})$

Stampa uno spazio

$i=i+1$

2. Stampa gli asterischi

$j=1$

Finché j è minore o uguale a $(\text{riga}*2-1)$

Stampa asterisco

$j=j+1$

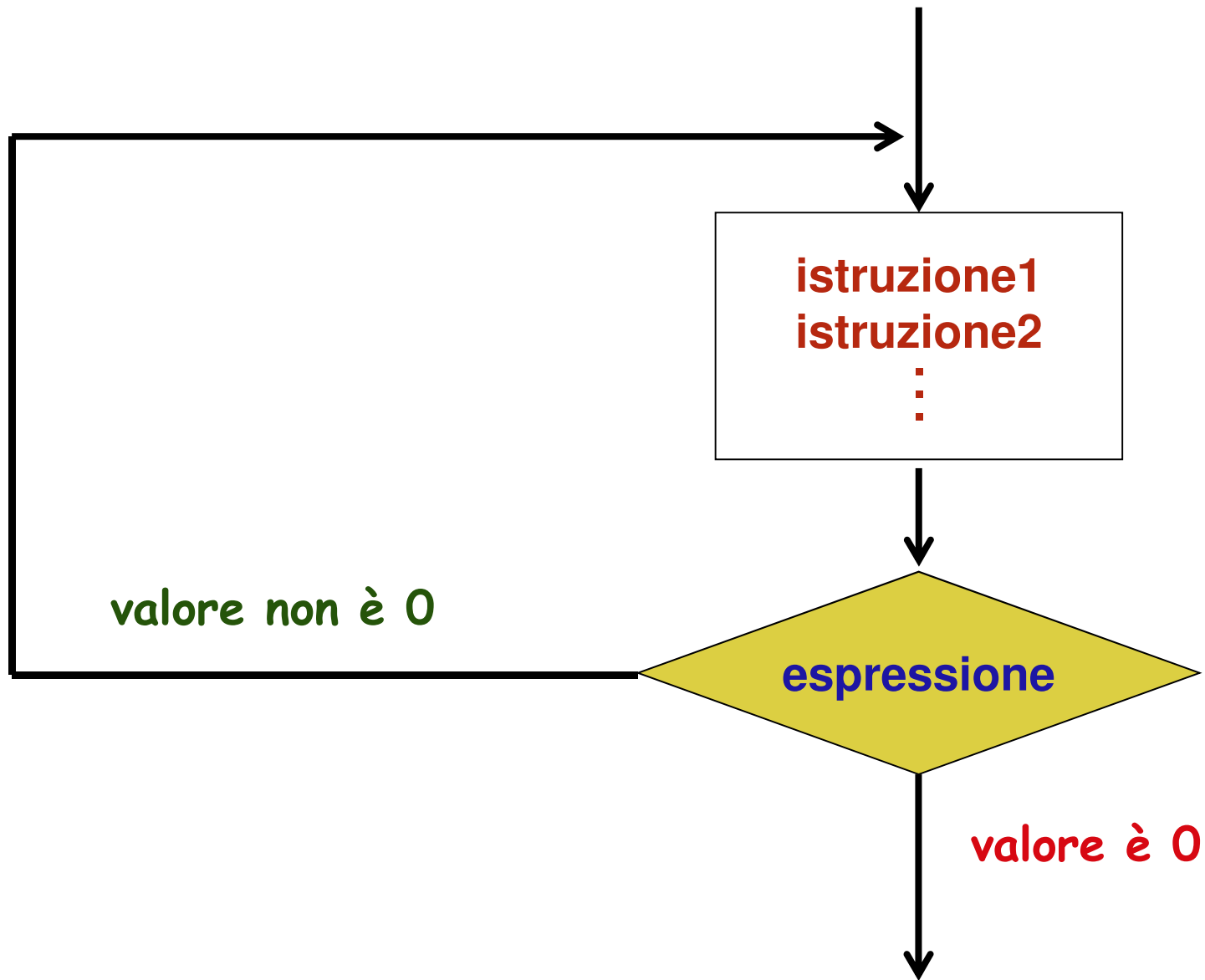
C: Ciclo do while



```
do { istruzione1 ; istruzione2 ; ... ; }  
while ( espressione )
```

- Il corpo del ciclo **istruzione1 ; istruzione2 ; ... ;** è eseguito una prima volta
- Se **espressione** ha valore **diverso da 0**, il corpo viene eseguito di nuovo, altrimenti si esce dal ciclo

C: Ciclo do while



C: Operatori ++ --



- Operatori unari: **incremento** (++) e **decremento** (--)
- ++ e -- sono applicabili solo a variabili e provocano incremento o decremento di 1
- Possono apparire come **suffisso** (p.es. x++) o come **prefisso** (p.es. --k)
- **Suffisso** definisce **postincremento** (++) o **postdecremento** (--) della variabile: si usa il valore corrente della variabile e poi lo si incrementa o decrementa di 1
- **Prefisso** definisce **preincremento** (++) o **predecremento** (--) della variabile: si incrementa o decrementa di 1 il valore corrente della variabile e poi si usa il valore



Operatori di assegnazione composta

- Si osservi che il seguente frammento di programma:

somma = somma + addendo;

salario = salario * aumento;

- si può abbreviare in

somma += addendo;

salario *= aumento;

- In generale l'assegnazione:

x = x <operatore> espressione;

si può abbreviare in

x <operatore>= espressione;

- Per ciascun operatore +, -, *, /, % esiste il corrispondente +=, -=, *=, /=, %=



Operatori di incremento e decremento

- Per incrementare di 1 una variabile intera si può usare una qualunque delle seguenti tre istruzioni equivalenti:

`x = x + 1;`

`x += 1;`

`x++;`

- Si noti che la forma più concisa è quella che usa l'operatore di **post-incremento: ++**

- In modo simile, per decrementare di 1 una variabile intera si può usare una qualunque delle seguenti tre istruzioni equivalenti:

`x = x - 1;`

`x -= 1;`

`x--;`

- Anche in questo caso la forma più concisa è quella che usa l'operatore di **post-decremento: --**



Espressioni con side-effect ed istruzioni (1)

- Con il termine *espressione* si indicano due diverse nozioni:
 - le espressioni che hanno come effetto solamente il calcolo di un valore, come ad esempio le espressioni del tipo `int`, che si possono comporre secondo le regole delle espressioni matematiche;
 - le espressioni che, oltre a calcolare un valore, corrispondono ad una operazione sulla memoria, come ad esempio un'assegnazione (semplice o composta) oppure un post-incremento.
 - Per queste espressioni useremo il termine *espressioni-con-side-effect*
 - Comportano una modifica della memoria



Espressioni con side-effect ed istruzioni (2)

□ *Esempio:*

- $23*x+5$ è una espressione matematica;
 - $x = 7$ è una espressione-con-side-effect ammessa il cui valore è 7 (il lato destro dell'assegnazione). Terminandola con ";" si ottiene l'istruzione valida $x = 7$;
 - $y = x = 7$ è anch'essa una espressione valida in C avente due side-effect: il primo assegna 7 a x , mentre il secondo assegna il valore dell'espressione $x = 7$ (che come detto vale 7) a y .
- Sebbene il linguaggio C consenta un uso indifferenziato dei due tipi di espressioni, noi useremo le **espressioni-con-side-effect** per formare delle istruzioni, evitando sempre l'uso di espressioni-con-side-effect all'interno di espressioni matematiche.



Espressioni con side-effect ed istruzioni (3)

□ *Esempio:*

- $x = 5 * (y = 7)$; dovrebbe essere sempre scritto
 - $y = 7$;
 - $x = 5*y$;
- Questa distinzione è motivata dal fatto che le espressioni sono *un'astrazione del concetto matematico di funzione e di applicazione di funzione*, mentre le espressioni-con-side-effect (istruzioni) sono *un'astrazione del concetto di assegnazione* cioè di modifica della memoria del programma.

Esempio



```
int main ( ) {  
    int c = 2 ;                               /* stampati */  
    printf( "%d\n", c ) ;                     /*      2      */  
    printf( "%d\n", c++ ) ;                   /*      2      */  
    printf( "%d\n", c ) ;                     /*      3      */  
    c = 2 ;  
    printf( "%d\n", c ) ;                     /*      2      */  
    printf( "%d\n", ++c ) ;                   /*      3      */  
    printf( "%d\n", c ) ;                     /*      3      */  
    return 0 ;  
}
```

Esempio: iterazione definita



Stampa dei numeri da 1 a 100

Il numero di ripetizioni del corpo del while è determinabile a priori

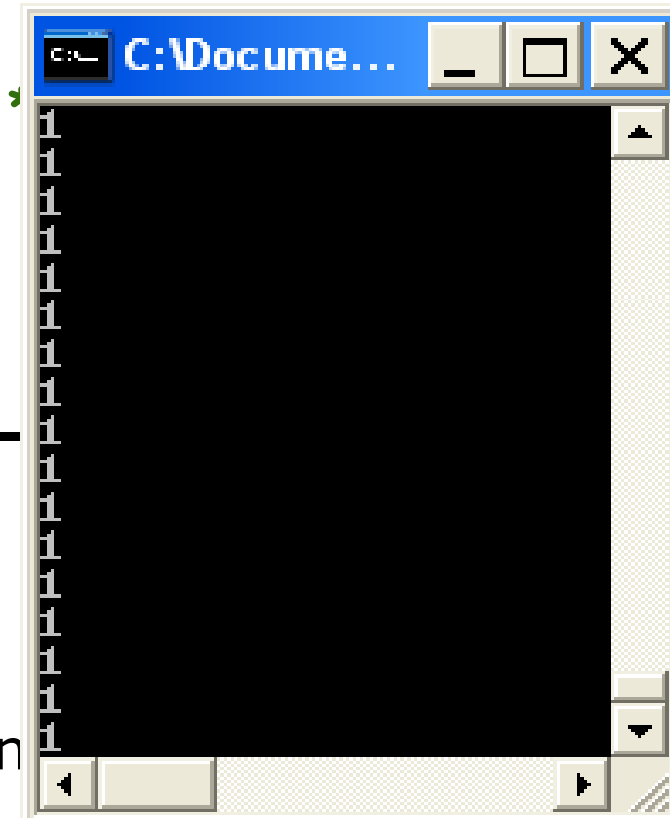
```
int main( )
{
    int num = 1 ;          /* inizializzazione contatore */
    while ( num <= 100 ) /* controllo valore massimo */
    {
        printf( "%d\n" , num ) ;
        num = num + 1 ;    /* incremento contatore */
    }
    return 0;
}
```

Errore nelle iterazioni

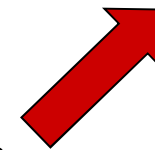


- Se il valore sentinella o il contatore non vengono opportunamente modificati all'interno del blocco di istruzioni del while, il programma può entrare in un ciclo infinito.

```
int main( )  
{  
    int num = 1 ;           /* inizializzazione contatore */  
    while ( num <= 100 )   /* controllo valore massimo */  
        printf( "%d\n" , num ) ;  
    num = num + 1 ;       /* incremento contatore */  
    return 0;  
}
```



Il programma entra in un ciclo infinito



Esempio: iterazione indefinita



Conta numeri interi positivi letti da input, la lettura termina quando viene letto il valore -1

Il numero di ripetizioni del corpo del while non è determinabile a priori

```
int main( )
{
    int num, cont = 0 ;
    printf ("Inserire sequenza di interi positivi terminata con -1 \n");
    scanf ("%d" , &num);
    while ( num != -1 )    /* controllo valore sentinella */
    {
        cont= cont + 1;
        scanf( "%d" , &num ); /* modifica valore sentinella */
    }
    printf ("Numero di interi letti = %d\n", cont);
    return 0;
}
```

Esercizio 1: Stampa max e min



Leggere da standard input una sequenza di 12 voti (tra 0 e 30) e stampare il massimo ed il minimo dei voti

Analisi dei dati (variabili da usare!)

Quattro **variabili** di tipo **unsigned short int**: contatore (**i**), voto corrente (**voto**), minimo (**min**) e massimo (**max**)

Algoritmo

Inizializza **min** e **max** (attenzione!)

Per ogni voto letto

se **min** è maggiore di **voto**, allora

min = voto

se **max** è minore di **voto**, allora

max = voto

Stampa **min** e **max**

Soluzione 1



```
#include <stdio.h>
#include <stdlib.h>
#define NUM_VOTI 12

int main () {
    int voto=0, min, max, i = 1;

    printf("\nInserire voto %d: ", i);
    scanf("%d", &voto);
    printf("Voto inserito: %d\n", voto);

    while ( voto > 30 || voto < 0 ) {
        printf("Voto non valido, inserire voto %d: ", i);
        scanf("%d", &voto);
        printf("Voto inserito: %d\n", voto);
    }
}
```

Soluzione 1



```
i = i + 1;
min = max = voto;
while ( i <= NUM_VOTI ) {
    printf("\nInserire voto %d: ", i);
    scanf("%d", &voto);
    printf("Voto inserito: %d\n", voto);
    while ( voto > 30 || voto < 0 ) {
        printf("Voto non valido, inserire voto %d: ", i);
        scanf("%d", &voto);
        printf("Voto inserito: %d\n", voto);
    }
    if ( voto > max )
        max = voto;
    if ( voto < min )
        min = voto;
    i = i + 1;
}

printf("Massimo = %d, Minimo = %d\n", max, min);
return 0;
}
```



Esercizio 2

Leggere da standard input una sequenza di caratteri terminata da invio e determinare il numero di occorrenze delle vocali all'interno della sequenza

```
Inserire una sequenza di caratteri terminata da invio:
```

```
Prova di esecuzione dell'esercizio numero 2
```

```
Numero occorrenze lettera a: 1
```

```
Numero occorrenze lettera e: 7
```

```
Numero occorrenze lettera i: 4
```

```
Numero occorrenze lettera o: 4
```

```
Numero occorrenze lettera u: 2
```

```
Premere un tasto per continuare . . .
```


Soluzione 2



```
#include <stdio.h>
#include <stdlib.h>
int main () {
    int cont_a=0, cont_e=0, cont_i=0, cont_o=0, cont_u=0;
    char carattere;

    printf("\nInserire una sequenza di caratteri terminata da invio: \n");
    scanf("%c", &carattere);

    while ( carattere != '\n' ) {
        switch (carattere) {
            case 'a': cont_a = cont_a +1; break;
            case 'e': cont_e = cont_e +1; break;
            case 'i': cont_i = cont_i +1; break;
            case 'o': cont_o = cont_o +1; break;
            case 'u': cont_u = cont_u +1; break;
        }
        scanf("%c", &carattere);
    }
}
```



Soluzione 2

```
printf("Numero occorrenze lettera a: %d\n", cont_a);  
printf("Numero occorrenze lettera e: %d\n", cont_e);  
printf("Numero occorrenze lettera i: %d\n", cont_i);  
printf("Numero occorrenze lettera o: %d\n", cont_o);  
printf("Numero occorrenze lettera u: %d\n", cont_u);  
  
system("pause");  
return 0;  
}
```



Esercizio 3

Leggere da standard input un numero intero $N \geq 1$ e stampare la somma dei numeri da 1 a N .

```
Inserire n(>=1): 100
Risultato = 5050
Premere un tasto per continuare . . .
```



Soluzione 3

```
int n, i = 1, somma = 0;
printf("\nInserire n(>=1): ");
scanf("%d", &n);

while ( n <= 0 ) {
    printf ("Inserire n (>=1): ");
    scanf("%d", &n);
}

while ( i <= n ) {
    somma = somma + i;
    i = i + 1;
}

printf("Risultato = %d\n", somma);
```

Esercizio 4



Leggere da input due numeri interi n ed m e stampare tutti i numeri pari compresi tra n ed m se n è minore di m , oppure compresi tra m ed n se m è minore di n

```
Inserire n ed m: 4 71
6  8  10  12  14  16  18  20  22  24  26  28  30  32
34  36  38  40  42  44  46  48  50  52  54  56  58  60
62  64  66  68  70
Premere un tasto per continuare . . .
```



Soluzione 4

```
int n, m, i;
```

```
printf("\nInserire n ed m: ");
```

```
scanf("%d%d", &n, &m);
```

```
if ( n < m ) {  
    i = n + 1;  
    while ( i < m ) {  
        if ( i%2 == 0 )  
            printf ("%d ", i);  
        i = i + 1;  
    }  
}  
else {  
    i = m + 1;  
    while ( i < n ) {  
        if ( i%2 == 0 )  
            printf ("%d ", i);  
        i = i + 1;  
    }  
}
```

```
if ( n < m ) {  
    i = n + 1;  
    while ( i < m ) {  
        if ( i%2 == 0 )  
            printf ("%d ", i);  
        i = i + 1;  
    }  
}
```

```
    i = m + 1;  
    while ( i < n ) {  
        if ( i%2 == 0 )  
            printf ("%d ", i);  
        i = i + 1;  
    }
```

Esercizio 4 bis



Leggere da input due numeri interi n ed m e stampare tutti i numeri dispari compresi tra n ed m se n è minore di m , oppure compresi tra m ed n se m è minore di n

Soluzione 4 bis



```
int n, m, i, max;
```

```
printf("\nInserire n ed m: ");
```

```
scanf("%d%d", &n, &m);
```

```
if ( n < m ) {
```

```
    max = m;
```

```
    i = n + 1;
```

```
}
```

```
else {
```

```
    max = n;
```

```
    i = m + 1;
```

```
}
```

```
while ( i < max ) {
```

```
    if ( i%2 != 0 )
```

```
        printf ("%d ", i);
```

```
    i = i + 1;
```

```
}
```