Programmare in C
(strutture di controllo)

Maurizio Palesi
Salvatore Serrano

Esempio: Algoritmo del Risveglio

1. Alzarsi dal letto
2. Togliersi il pigiama
3. Fare la doccia
4. Vestirsi
5. Fare colazione
6. Prendere il bus per andare a scuola

Nota

I passi sono eseguiti in sequenza e l'ordine delle istruzioni è essenziale per la correttezza dell'algoritmo

Non basta organizzare i passi in sequenza...

1. Alzarsi dal letto
2. Togliersi il pigiama
3. Fare la doccia
4. Vestirsi
5. Fare colazione
6. Se piove
   Prendere ombrello
7. Prendere il bus per andare a scuola

Ulteriore forma di flusso: Se...Altrimenti

1. Alzarsi dal letto
2. Togliersi il pigiama
3. Fare la doccia
4. Vestirsi
5. Fare colazione
6. **Se** sciopero mezzi pubblici
   Prendere macchina
   altrimenti
   Prendere il bus

Ulteriore forma di controllo: ciclo “mentre”

1. Alzarsi dal letto
2. Togliersi il pigiama
3. Fare la doccia
4. Vestirsi
5. Fare colazione
6. **Mentre** piove
   Restare in casa
7. Prendere il bus per andare a scuola

Linguaggio Naturale

- Se…allora…altrimenti...
- qualora… ma… nel caso in cui…
- ripeti… fino a quando…
- mentre…
- nel primo caso…, nel secondo caso...

Nei linguaggi di programmazione questi costrutti sintattici vengono chiamati

strutture di controllo

Strutture di controllo del flusso

- Istruzione composta
- Istruzione decisionale
- Istruzione ciclica

Istruzione Composta

- Le parentesi graffe { } vengono usate per ragguppare in un'unica istruzione composta, detta **blocco**, dichiarazioni e istruzioni, in modo che, dal punto di vista sintattico esse formino un'entità equivalente ad una sola istruzione
- Esempio

```
{
   a = b + 19;
   b = c * 23;
   c = c + 1;
}
```

Strutture di controllo del flusso

- Istruzione composta
- Istruzione decisionale
  - if
  - if-else
  - switch
- Istruzione ciclica

if

- Sintassi

```
if (espressione)
  istruzione;
```

- Semantica

  Valuta **espressione**. Se l'espressione è vera (cioè assume valore diverso da 0) esegue l'istruzione **istruzione**. In ogni caso continua eseguendo la prima istruzione successiva alla struttura di controllo.

Esempio if

```
#include <stdio.h>

main()
{
   int n;

   printf("Dammi un numero: ");
   scanf("%d", &n);

   if (n < 100)
      printf("minore di 100\n");
}
```

Stampa "minore di 100" se il numero letto è minore di 100

Esempio if + istruzione composta

```
#include <stdio.h>
main()
{
   int n;
   int min100 = 0;

   printf("Dammi un numero: ");
   scanf("%d", &n);
   if (n < 100)
   {
      printf("minore di 100\n");
      min100 = 1;
   }
}
```

Stampa "minore di 100" se il numero letto è minore di 100 e assegna 1 alla variabile min100

Esercizio if

- Scrivere un programma che richieda in ingresso un numero intero e stampa il suo valore assoluto
- Valore assoluto di un numero n
  - Se n<0 → -n
  - Se n ³ 0 → n
- **Esempio**
  - -9 → 9
  - 7 → 7

Soluzione

```
#include <stdio.h>

main()
{
   int n;

   printf("Dammi un numero: ");
   scanf("%d", &n);

   if (n < 0)
      n = -n;

   printf("%d", n);
}
```

If-else

- Sintassi

```
if (espressione)
  istruzione_1;
else
  istruzione_2;
```

- Semantica

  Valuta **espressione**. Se l'espressione è vera (cioè assume valore diverso da 0) esegue l'istruzione **istruzione_1**. Se l'espressione è falsa (cioè assume valore 0) esegue l'istruzione **istruzione_2**.

Esempio if-else

```
#include <stdio.h>

main()
{
   int n;

   printf("Dammi un numero: ");
   scanf("%d", &n);

   if (n < 100)
      printf("minore di 100\n");
   else
      printf("maggiore di 100\n");
}
```

Stampa "minore di 100" se il numero letto è minore di 100. Stampa maggiore di 100 se il numero letto è maggiore o uguale a 100.

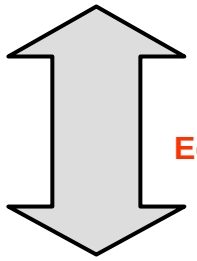
If-else + istruzione composta

```
#include <stdio.h>
main()
{
   int n;
   int min100 = 0, mag100 = 0;

   printf("Dammi un numero: ");
   scanf("%d", &n);
   if (n < 100)
   {
      printf("minore di 100\n");
      min100 = 1;
   } else {
      printf("minore di 100\n");
      mag100 = 1;
   }
}
```


Osservazione

```
if (x != 0)
   printf("pippo");
```



Equivalenti

```
if (x)
   printf("pippo");
```

Esercizio

```
#include <stdio.h>
main()
{
   int a, b, max;

   printf("Dammi un numero: ");
   scanf("%d", &a);
   printf("Dammi un altro numero: ");
   scanf("%d", &b);

   if (a > b)
      max = a;
   else
      max = b;

   printf("il maggiore è %d\n", max);
}
```

Stampa il maggiore di due interi immessi da tastiera.

Esercizio

```
#include <stdio.h>
main()
{
   int n;

   printf("Dammi un numero: ");
   scanf("%d", &n);

   if (n % 2 == 0)
      printf("%d è un numero pari\n", n);
   else
      printf("%d è un numero dispari\n", n);
}
```

Determina se il numero immesso da tastiera è pari o dispari.

Esercizio

```
#include <stdio.h>
main(){
   int a, b, c, max;
   printf("Dammi tre numeri: ");
   scanf("%d %d %d", &a, &b, &c);
   if (a > b)
      if (a > c)
         max = a;
      else
         max = c;
   else
      if (b > c)
         max = b;
      else
         max = c;
   printf("Il maggiore è %d\n", max);
}
```

Stampa il maggiore tra tre numeri immessi da tastiera.

Esercizio

```
#include <stdio.h>
main()
{
   int a, b, c, max;

   printf("Dammi tre numeri: ");
   scanf("%d %d %d", &a, &b, &c);

   if (a > b && a > c)
      max = a;
   else if (b > a && b > c)
      max = b;
   else
      max = c;

   printf("Il maggiore è %d\n", max);
}
```

Stampa il maggiore tra tre numeri immessi da tastiera.

Esercizio

- Scrivere un programma che legga un carattere e lo traduce nel suo corrispondente carattere minuscolo. Se il carattere di input non è maiuscolo, o non è una lettera, allora lo stampi inalterato.

Codice ASCII	
Carattere	Decimale
...	...
A	65
B	66
C	67
...	...
a	97
b	98
c	99
...	...

Soluzione

```
#include <stdio.h>

main()
{
   char car;

   printf("Inserisci un carattere: ");
   scanf("%c" &car);

   if (car >= 'A' && car <= 'Z')
      car = car + 32;

   printf("%c\n", car);
}
```

Esercizio

- Scrivere un programma per il calcolo della radici di un'equazione di secondo grado di coefficienti *a*, *b* e *c* immessi da tastiera.
- $ax^2 + bx + c = 0$
  - delta = b^2 - 4ac
  - a = b = 0 → Degenere
  - a = 0 → I grado (1 radice, x=-c/b)
  - delta >0 → $x_{1,2} = (-b \pm \text{sqrt(delta)})/2a$
  - delta <0 → radici complesse

Soluzione

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

main()
{
   float a, b, c;
   float delta, r1, r2;

   printf("coefficiente di secondo grado: ");
   scanf("%f", &a);
   printf("coefficiente di primo grado: ");
   scanf("%f", &b);
   printf("termine noto: ");
   scanf("%f", &c);
```

continua

Soluzione

continuo

```
if (a == 0 && b == 0)
   printf("degenere!\n");
else if (a == 0)
   printf("Equazione di primo grado. X=%f\n", -c/b);
else {
   delta = b*b - 4*a*c;
   if (delta < 0)
      printf("Discriminante negativo!\n");
   else {
      r1 = (-b + sqrt(delta))/(2*a);
      r2 = (-b - sqrt(delta))/(2*a);
      printf("radici %f %f\n", r1, r2);
   }
}
```


Esercizio

- In accordo con le regole del Calendario Gregoriano un anno è bisestile quando è multiplo di 4 e non è un secolo oppure è un secolo multiplo di 400
- Scrivere un programma che verifichi se l'anno inserito da tastiera è bisestile

Esempio

➔1992	bisestile (multiplo di 4)
➔1800	no (secolo non multiplo di 400)
➔2000	bisestile (secolo multiplo di 400)

Soluzione

```
#include <stdio.h>

main() {
   int anno;
   int ris;
   printf("Anno: ");
   scanf("%d", &anno);
   ris = ((anno % 4 == 0) && (anno % 100 != 0))
             || (anno % 400 == 0);
   if (ris == 1)
       printf("Bisestile\n");
   else
       printf("NON bisestile\n");
}
```


switch

- Sintassi

```
switch (espressione){
    case costante1:sequenza_di_istruzioni_1;break;
    case costante2:sequenza_di_istruzioni_2;break;
    …
    case costanteN:sequenza_di_istruzioni_N;break;
default:sequenza_di_istruzioni_Default;
}
```

- Semantica

  Struttura di scelta multipla. Controlla se un'espressione assume un valore all'interno di un certo insieme di costanti e si comporta di conseguenza.

Esempio switch

Stampa un numero di asterischi pari all'intero inserito da tastiera fino a un massimo di 3. Altrimenti stampa un punto esclamativo.

```
#include <stdio.h>

main() {
      int num;
      printf("Dammi un numero: ");
      scanf("%d", &num);
      switch (num) {
            case 1: printf("*\n"); break;
            case 2: printf("**\n"); break;
            case 3: printf("***\n"); break;
            default: printf("!\n");
      }
}
```


Osservazione 1

- Le possibili scelte devono essere valori costanti

```
#include <stdio.h>

main() {
   int num;
   int uno = 1;
   int due = 2;
   printf("Dammi un numero: ");
   scanf("%d", &num);
   switch (num) {
       case uno: printf("Pippo\n"); break;
       case due: printf("Pluto\n"); break;
       default: printf("Topolandia\n");
   }
}
```

SBAGLIATO
valori non costanti

Osservazione 2

- **break** non è strettamente indispensabile
  - ➔ Se non è presente viene eseguita sequenzialmente ogni istruzione a partire dal **case** che è stato raggiunto

```
#include <stdio.h>

main() {
   int num;
   printf("Dammi un numero: ");
   scanf("%d", &num);
   switch (num) {
       case 1: printf("*\n");
       case 2: printf("**\n");
       case 3: printf("***\n");
       default: printf("!\n");
   }
}
```


Osservazione 3

- Possono esserci più etichette per una stessa sequenza di istruzioni

```
#include <stdio.h>
main() {
   char car;
   scanf("%c", &car);
   switch (car) {
       case 'a':case 'e':
       case 'i':case 'o':
       case 'u':printf("Vocale minuscola\n");break;
       case 'A':case 'E':
       case 'I':case 'O':
       case 'U':printf("Vocale maiuscola\n");break;
       default: printf("Non è una vocale\n");
   }
}
```

Esercizio

- Scrivere un programma che visualizza il seguente menu:

```
Menu di prova
1)  Immettere dati
2)  Visualizzare dati
3)  Modificare dati
Scelta:
```

Quindi aspetta l'immissione di un carattere da parte dell'utente e visualizza una scritta corrispondente alla scelta effettuata, del tipo:

```
In esecuzione l'opzione 1
```

Se la scelta non è tra quelle proposte **(1,2,3)** deve essere visualizzata la scritta

```
Opzione inesistente
```


Soluzione

```
#include <stdio.h>
main() {
    int num;
    printf("Menu di prova\n");
    printf("1) Immettere dati\n");
    printf("2) Visualizzare dati\n");
    printf("3) Modificare dati\n");
    printf("Scelta:");
    scanf("%d", &num);
    switch (num) {
       case 1: printf("In esecuzione l'opzione 1"); break;
       case 2: printf("In esecuzione l'opzione 2"); break;
       case 3: printf("In esecuzione l'opzione 3"); break;
       default: printf("Opzione inesistente\n");
    }
}
```

Esercizio

- Scrivere un programma che riceve in ingresso un mese (numero intero) e stampa quanti giorno ha quel mese

```
4,6,9,11              Þ   30
1,3,4,5,7,8,10,12         Þ   31
2                     Þ   28
mÏ[1..12]             Þ   non esistente
```


Soluzione stupida

```
#include <stdio.h>
main() {
    int num;
    printf("Dammi un mese (numero):");
    scanf("%d", &num);
    switch (num) {
       case 1: printf("31 giorni\n"); break;
       case 2: printf("28 giorni\n"); break;
       case 3: printf("31 giorni\n"); break;
       case 4: printf("30 giorni\n"); break;
       case 5: printf("31 giorni\n"); break;
       case 6: printf("30 giorni\n"); break;
       case 7: printf("31 giorni\n"); break;
       case 8: printf("31 giorni\n"); break;
       case 9: printf("30 giorni\n"); break;
       case 10: printf("31 giorni\n"); break;
       case 11: printf("30 giorni\n"); break;
       case 12: printf("31 giorni\n"); break;
       default: printf("Mese inesistente\n");
    }
}
```

Soluzione furba

```
#include <stdio.h>
main() {
   int num;
   printf("Dammi un mese (numero):");
   scanf("%d", &num);
   switch (num) {
     case 1: case 3: case 5:
     case 7: case 8: case 10:
     case 12: printf("31 giorni\n"); break;
     case 4: case 6: case 9:
     case 11: printf("30 giorni\n"); break;
     case 2: printf("28 giorni\n"); break;
     default: printf("Mese inesistente\n");
   }
}
```


Strutture di controllo del flusso

- Istruzione composta
- Istruzione decisionale
- Istruzione ciclica

Istruzioni cicliche

- Consentono di realizzare cicli di elaborazione, ossia l'esecuzione ripetuta di una sequenza di istruzioni
- Due differenti tipologie di istruzioni cicliche
  - ➔Il numero di volte per il quale viene ripetuta l'esecuzione della sequenza è noto a priori
  - ➔Il numero di volte per il quale la sequenza viene ripetuta non è noto a priori, ma è condizionato dal verificarsi di un evento assegnato

Esempi

- Calcolare le paghe dei dipendenti di una azienda
  - ➔Noto il numero `N` dei dipendenti, ripetere `N` volte la sequenza *calcolo della paga*
- Riempire il piatto della bilancia fino ad ottenere il peso desiderato
  - ➔La sequenza *mettere sul piatto* va eseguita e rieseguita fino a quando non si raggiunge il peso desiderato

Istruzioni cicliche

- **for**
- **while**
- **do…while**

Esempio

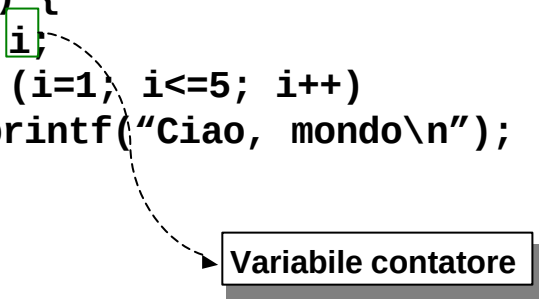
- Vogliamo visualizzare cinque volte la scritta **Ciao, mondo**

Il numero di volte per il quale viene ripetuta l'esecuzione dell'istruzione è noto a priori: 5

```
…
printf("Ciao, mondo\n");
printf("Ciao, mondo\n");
printf("Ciao, mondo\n");
printf("Ciao, mondo\n");
printf("Ciao, mondo\n");
…
```

Usando il for

```
#include <stdio.h>
main() {
  int i;
  for (i=1; i<=5; i++)
     printf("Ciao, mondo\n");
}
```



for

- Sintassi

```
for (<inizializzazione>; <condizione>; <aggiornamento>)
      <istruzione>
```

- Semantica
  - ➔ `<inizializzazione>`
    - ✓ Viene eseguita una sola volta
    - ✓ Serve per impostare le variabili
    - ✓ Può anche non essere presente
  - ➔ `<condizione>`
    - ✓ Viene valutata ogni volta prima di eseguire le istruzioni del ciclo
      - – Se è vera si esegue ancora `<istruzione>`
      - – Se è falsa si esce dal ciclo passando all'istruzione successiva al programma
    - ✓ Può essere non presente per realizzare un ***loop infinito***
  - ➔ `<aggiornamento>`
    - ✓ Viene eseguito alla fine di ogni ciclo
    - ✓ Serve tipicamente ad aggiornare il valore della variabile ***contatore***
    - ✓ Può anche non essere presente

Esempio

```
#include <stdio.h>
main() {
   int x;
   for (x=1; x<=10; x++)
       printf("%d\n", x);
}
```

x	x<=10
1	Si
2	Si
3	Si
4	Si
5	Si
6	Si
7	Si
8	Si
9	Si
10	Si
11	No

Si valuta la condizione
viene eseguita l'istruzione del ciclo
si aggiorna il contatore

Si valuta la condizione
l'istruzione del ciclo non viene eseguita
si esce dal ciclo

Esempio

```
#include <stdio.h>
main() {
   int x;
   for (x=10; x>=0; x-=2)
       printf("%d\n", x);
}
```

x	x>=0
10	Si
8	Si
6	Si
4	Si
2	Si
0	Si
-2	No

Si valuta la condizione
viene eseguita l'istruzione del ciclo
si aggiorna il contatore

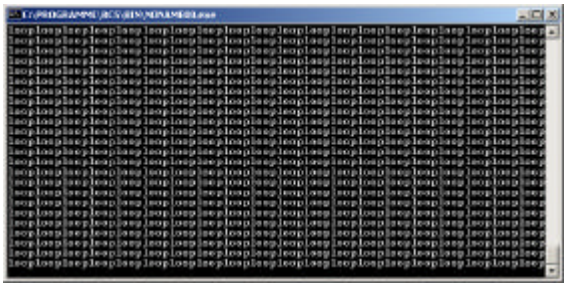
Si valuta la condizione
l'istruzione del ciclo non viene eseguita
si esce dal ciclo

Esempio: loop infinito

```
#include <stdio.h>
main() {
   int x;
   for ( ; ; )
        printf("loop");
}
```

```
#include <stdio.h>
main() {
   int x;
   for (x=0; ; )
        printf("loop");
}
```

```
#include <stdio.h>
main() {
   int x;
   for (x=0; x==0; )
        printf("loop");
}
```



Osservazione 1

- Dato che la condizione viene valutata prima di ogni ciclo il **for** permette anche di non eseguire nemmeno una volta le istruzioni che fanno parte del corpo del ciclo

- Esempio

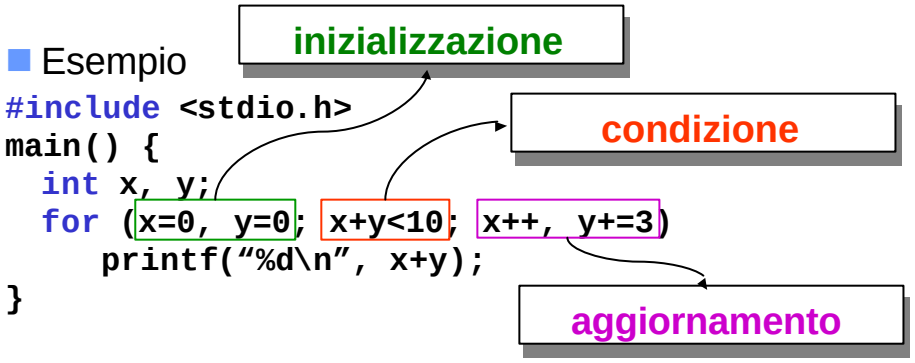
```
#include <stdio.h>
main() {
  int x;
  for (x=10; x<5; x++)
     printf("%d\n", x);
}
```

NON STAMPA NULLA

Osservazione 2

- **<inizializzazione>** ed **<aggiornamento>**, nella sintassi del **for**, possono contenere più istruzioni, che dovranno essere separate da virgola

- Esempio




```
#include <stdio.h>
main() {
  int x, y;
  for (x=0, y=0; x+y<10; x++, y+=3)
     printf("%d\n", x+y);
}
```


Esercizio

- Scrivere un programma che richiede all'utente un numero naturale n e calcola la somma dei primi n numeri naturali

Soluzione

```
#include <stdio.h>
main() {
   int n, somma, i;
   printf("Inserisci n:");
   scanf("%d", &n);
   somma=0;
   for (i=1; i<=n; i++)
       somma+=i;
   printf("La somma dei primi n num = %d", somma);
}
```


Esercizio

- Scrivere un programma che richiede all'utente un numero naturale n e calcola la somma di n numeri inseriti dall'utente

Soluzione

```
#include <stdio.h>
main() {
   int n, somma=0, num, i;
   printf("Inserisci n:");
   scanf("%d", &n);
   printf("Inserimento di %d numeri\n", n);
   for (i=1; i<=n; i++) {
       printf("Inserisci il %d^ numero:", i);
       scanf("%d", &num);
       somma+=num;
   }
   printf("Somma = %d", somma);
}
```


Esercizio

- Scrivere un programma che richiede all'utente un numero naturale n e calcola il massimo degli n numeri inseriti dall'utente.

Soluzione

```
#include <stdio.h>
main() {
   int n, max, num, i;
   printf("Inserisci n:");
   scanf("%d", &n);
   printf("Inserimento di %d numeri\n", n);
   i=1;
   printf("Inserisci il %d^ numero:", i);
   scanf("%d", &max);
   for (i=2; i<=n; i++) {
       printf("Inserisci il %d^ numero:", i);
       scanf("%d", &num);
       if (num > max)
             max = num;
   }
   printf("Massimo = %d", max);
}
```


Esercizio

- Scrivere un programma che richiede all'utente un naturale n e calcola il fattoriale di n, indicato con n!

  n!:
  0! = 1
  n! = n • (n-1) • (n-2) • … • 2 • 1

Soluzione

```
#include <stdio.h>
main() {
   int n, fatt, i;
   printf("Inserisci n:");
   scanf("%d", &n);
   fatt=1;
   for (i=n; i>=1; i--)
       fatt*=i;
   printf("n! = %d", fatt);
}
```


Istruzioni cicliche

- for
- while
- do…while

while

- Permette di ottenere la ripetizione ciclica di una istruzione (o di un blocco di istruzioni) sotto il controllo di una condizione di terminazione
- Sintassi

```
while (<espressione>)
      <istruzione>
```

- Semantica
  - **<espressione> deve essere valutata ogni volta prima di eseguire <istruzione>**
  - **se la valutazione di <espressione> è vera allora <istruzione> viene eseguita, altrimenti no e si esce dal ciclo**
  - **La valutazione dell'espressione è effettuata all'inizio, il ciclo quindi può essere eseguito:**
    - ✓ **Zero volte**
    - ✓ **Un numero finito di volte**
    - ✓ **Infinite volte**

Convergenza del ciclo

- Assicurarsi che nel corpo del **while** venga alterato il valore di una delle informazioni coinvolte in **<espressione>**
- Assicurarsi che in un numero finito di iterazioni **<espressione>** diventi falsa
- Assicurarsi che tutte le variabili utilizzate in **<espressione>** siano state inizializzate prima del **while**

Esempio while

- Somma numeri interi in ingresso finchè non viene dato lo zero

Alla variabile num si è assegnato il valore 1 per far in modo che il ciclo venga eseguito almeno una volta

```
#include <stdio.h>
main() {
   int num = 1;
   int somma = 0;
   while (num != 0) {
       printf("Inserisci un numero (0 per finire):");
       scanf("%d", &num);
       somma+=num;
   }
   printf("Somma = %d", somma);
}
```


Esempio while

- Somma numeri interi in ingresso finchè non viene dato lo zero (soluzione alternativa)

Alla variabile num si è assegnato il primo numero inserito dall'utente fuori dal ciclo

```
#include <stdio.h>
main() {
   int num;
   int somma = 0;
   printf("Inserisci un numero (0 per finire):");
   scanf("%d", &num);
   while (num != 0) {
       somma+=num;
       printf("Inserisci un numero (0 per finire):");
       scanf("%d", &num);
   }
   printf("Somma = %d", somma);
}
```

Relazione tra for e while

```
for (<inizializzazione>; <condizione>; <aggiornamento>)
      <istruzione>
```

… equivale a:

```
<inizializzazione>
while (<condizione>) {
  <istruzione>;
  <aggiornamento>;
}
```


Esercizio

- Calcolare il massimo comune divisore fra due numeri x e y utilizzando l'algoritmo di Euclide:
  - ➔algoritmo di Euclide
    - ✓ Se x==y ⇒ MCD(x, y) = x = y
    - ✓ Se x>y ⇒ MCD(x, y) = MCD(x-y, y)
    - ✓ Se x<y ⇒ MCD(x, y) = MCD(x, y-x)
- Esempio

  x = 15, y = 9

  MCD(15, 9) = MCD(6, 9) = MCD(6,3) = MCD(3, 3) = 3

Soluzione

```
#include <stdio.h>
main() {
   int x, y;
   printf("Inserisci il 1^ numero:");
   scanf("%d", &x);
   printf("Inserisci il 2^ numero:");
   scanf("%d", &y);
   while (x != y) {
       if (x > y)
             x=x-y;
       else
             y=y-x;
   }
   printf("MCD = %d", x);
}
```


Esercizio

- Dato il programma:

```
#include <stdio.h>
main() {
   int i, x=10;
   for (i=1; i<=x; i++)
       printf("%d\n", i*x);
}
```

Scrivere un programma equivalente usando il **`while`**

Soluzione

```
#include <stdio.h>
main() {
   int i=1, x=10;
   while (i<=x) {
       printf("%d\n", i*x);
       i++;
   }
}
```


Istruzioni cicliche

- **for**
- **while**
- **do…while**

do-while

- **for** e **while** controllano la condizione di terminazione all'inizio del ciclo
- **do-while** controlla la condizione al termine di ogni iterazione
- Consente di eseguire un ciclo da 1 ad infinite volte

do-while

- **Sintassi**

```
do
    <istruzione>
while (<condizione>);
```

- **Semantica**

  <condizione> viene valutata ogni volta dopo aver eseguito **<istruzione>**

  se la valutazione di **<condizione>** è vera allora il ciclo viene ripetuto, cioè si riesegue **<istruzione>,** altrimenti si esce dal ciclo

Osservazione

- In altri linguaggi (es. *Pascal*, dove tale struttura prende il nome di `repeat…until`)
- Si esce dal ciclo quando **`<condizione>`** diventa vera
- In **C** mentre **`<condizione>`** è vera si continua ad eseguire il ciclo

```
#include <stdio.h>
main() {
   int n=12, k=5;
   do {
        n = n / 2;
        k = k – 1;
   } while (n >= k);
   printf("%d %d\n", n, k);
}
```

n	k
12	5
6	4
3	3
1	2

Esempio

- Legge numeri finché non viene inserito un numero minore o uguale a 10

```
#include <stdio.h>
main() {
   int num;
   do
        scanf("%d", &num);
   while (n > 10);
}
```

Esempio

- Somma numeri interi in ingresso finché non viene dato lo zero

```
#include <stdio.h>
main() {
   int num;
   int somma = 0;
   do {
       printf("Inserisci un numero (0 per finire):");
       scanf("%d", &num);
       somma+=num;
   } while (num != 0);
   printf("Somma = %d", somma);
}
```


Relazione tra do-while e while

```
do {
      <istruzione>;
} while (<condizione>);
```

…è equivalente a

```
<istruzione>;
while (<condizione>) {
      <istruzione>;
}
```

Esercizio

- Scrivere un programma che legge continuamente un carattere finché questo sia s oppure n. In questo caso stampa OK.

Soluzione

```
#include <stdio.h>
main() {
  char car;
  do {
      printf("Rispondi s oppure n:\n");
      scanf("%c", &car);
  } while ((car != 's') && (car != 'n'));
  printf("OK");
}
```

Anche il tasto [INVIO] corrisponde a un carattere accettato da `scanf("%c", &car)`
È necessaria quindi una ulteriore lettura di carattere in una variabile ausiliaria.

Soluzione

```
#include <stdio.h>
main() {
   char car, pausa;
   do {
       printf(“Rispondi s oppure n:\n”);
       scanf(“%c”, &car);
       scanf(“%c”, &pausa);
   } while ((car != ‘s’) && (car != ‘n’));
   printf(“OK”);
}
```


Stile di programmazione

- Evitare programmi prolissi
- Non usare variabili più del necessario
- Memorizzare i risultati di computazioni intermedie quando tali risultati sono riusati

Esempio

- Prolisso

```
#include <stdio.h>
main() {
   int somma, diff;
   int ris;
   int a1, a2;

   printf("Introduci dato:");
   scanf("%d", &a1);
   printf("Introduci dato:");
   scanf("%d", &a2);

   somma = a1 + a2;
   diff = a1 – a2;
   ris = somma / diff;

    printf("Ris %d\n", ris);
}
```

- Semplice

```
#include <stdio.h>
main() {
   int a1, a2;

   printf("Introduci dato:");
   scanf("%d", &a1);
   printf("Introduci dato:");
   scanf("%d", &a2);

    printf("Ris %d\n",
           (a1+a2)/(a1-a2));
}
```


Esempio

- Peggiore

```
s = (m+n) + 5 * (m+n) * (m+n);

t = (m+n) / (m-n);

.
.
.

u = 3 * (m+n);
```

- Migliore

```
sum = (m+n);

s = sum + 5 * sum * sum;

t = sum / (m-n);

.
.
.

u = 3 * sum;
```

Evito di ricalcolare ogni volta l'espressione m+n

Esempio

- Peggiore

```
while ((i<m-n+1) && (ris<max))
   ris = ris * i;
```

- Migliore

```
sup = m-n+1;
while ((i<sup) && (ris<max))
   ris = ris * i;
```

Evito di ricalcolare ogni volta l'espressione `m-n+1`